aws

開發人員指南

Amazon GameLift

Amazon GameLift: 開發人員指南

Table of Contents

什麼是 Amazon GameLift？

您可以使用 Amazon GameLift 在雲端部署、操作和擴展專屬、低成本伺服器，以進行工作階段型多人遊戲。Amazon 以AWS全球運算基礎設施為基礎，可GameLift協助交付高效能、高可靠性的遊戲伺服器，同時動態擴展資源使用量以滿足全球玩家的需求

亞馬遜的用途 GameLift

Amazon GameLift 支援以下使用案例及其他使用案例：

- 使用您自己的自定義多人遊戲服務器，或使用ready-to-go實時服務器託管您的遊戲。
- 使用亞馬遜彈性運算雲端 (Amazon EC2) 競價型執行個體執行低成本託管資源。
- 根據使用情況自動調整遊戲所需的託管資源量。
- 使用亞馬遜軟體集中管理您的 Amazon EC2 運算資源GameLift。
- 在多人遊戲中與亞馬遜匹配玩家GameLiftFlexMatch。
- 使用 Amazon 反覆測試您的遊戲伺服器和用戶端建置。GameLift Anywhere
- 使用您自己的硬體，同時透過 Amazon 集中管理所有硬體GameLiftAnywhere。

> **Tip**
>
> 若要試用 Amazon GameLift 遊戲伺服器託管，請參閱開始使用亞馬遜 GameLift。

開始使用亞馬遜GameLift解決方案

適用於GameLift遊戲開發人員的 Amazon

- 亞馬遜GameLift託管自定義服務器
- 亞馬遜GameLift託管與實時服務器
- 亞馬遜GameLift飛行在亞馬遜 EC2 上託管
- 亞馬遜GameLiftFlexMatch的配對
- 亞馬遜GameLiftAnywhere硬件託管

亞馬遜GameLift託管自定義服務器

亞馬遜GameLift取代了託管自己的自定義遊戲服務器所需的工作。自動擴展功能可協助您避免支付超出所需資源的費用。自動縮放功能還有助於確保您始終擁有可供新玩家加入的遊戲，而且只需最少的等待時間。

如需 Amazon GameLift 託管的詳細資訊，請參閱亞馬遜如何GameLift工作。

主要功能

- 使用 Amazon GameLift 管理功能，包括自動擴展、多位置佇列和遊戲工作階段放置。
- 部署要在 Amazon Linux 或 Windows Server 作業系統上執行的遊戲伺服器。
- 管理遊戲工作階段和玩家工作階段。
- 針對伺服器處理序設定自訂的健全狀況追蹤，以偵測問題並解決效能不佳的程序。
- 使用適用於 Amazon 的AWS CloudFormation範本管理您的遊戲資源GameLift。

亞馬遜GameLift託管與實時服務器

使用實時服務器站起來不需要定制遊戲服務器的遊戲。這個輕量級的伺服器解決方案提供遊戲伺服器，您可以根據遊戲進行配置。

如需 Amazon 使用即時伺服器GameLift託管的詳細資訊，請參閱將遊戲與 Amazon GameLift 即時伺服器整合。

主要功能

- 使用 Amazon GameLift 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 託管資源並為您的車隊選擇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。
- 對實時配置和服務器邏輯進行實時更新。

亞馬遜GameLift飛行在亞馬遜 EC2 上託管

使用亞馬遜 GameLift FleetIQ 直接使用您在亞馬遜 EC2 和亞馬遜 EC2 自動擴展中的託管資源。這為亞馬遜GameLift優化提供了價格低廉，具有彈性的遊戲託管的好處。此解決方案適用於需要比全受管 Amazon 解決GameLift方案提供更大彈性的遊戲開發人員。

如需 Amazon GameLift FleetIQ 如何搭配 Amazon EC2 和 EC2 自動擴展進行遊戲託管的相關資訊，請參閱 Amazon GameLift FleetI Q 開發人員指南。

主要功能

- 使用 FleetIQ 演算法取得最佳化競價型執行個體平衡。
- 使用玩家路由功能有效地管理您的遊戲伺服器資源，並為加入遊戲提供更好的玩家體驗。
- 根據玩家使用情況自動擴展託管容量。
- 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，包括視窗、Linux、容器和 Kubernetes。

亞馬遜GameLiftFlexMatch的配對

用FlexMatch於建立自訂規則集，為您的遊戲定義多人遊戲比賽。FlexMatch使用規則集來比較每場比賽的兼容玩家，並為玩家提供理想的多人遊戲體驗。

如需詳細資訊FlexMatch，請參閱什麼是 Amazon GameLiftFlexMatch？

主要功能

- 平衡匹配創建速度和匹配質量。
- 根據定義的特徵匹配球員或團隊。
- 定義規則以根據延遲將玩家置於比賽中。

亞馬遜GameLiftAnywhere硬件託管

使用 Amazon GameLift Anywhere 將環境中任何位置的硬體整合到 Amazon GameLift 遊戲託管中。您可以將Anywhere艦隊和 EC2 叢集整合到分房系統和遊戲工作階段佇列中，以管理硬體上的配對和遊戲配置。

如需使用進行測試的詳細資訊Anywhere，請參閱使用亞馬遜機GameLiftAnywhere隊測試您的整合。如需設定Anywhere叢集的詳細資訊，請參閱建立亞馬遜GameLift車隊。

主要功能

- 對遊戲伺服器和用戶端組建執行快速的反覆測試。
- 使用設定的 Amazon GameLift 工具將遊戲部署到您自己的硬體。

- 隨時隨地使用離您玩家最近的硬體。

訪問亞馬遜 GameLift

使用這些工具與亞馬遜合作GameLift。

亞馬遜GameLift開發套件

Amazon GameLift SDK 包含GameLift從遊戲用戶端、遊戲伺服器和遊戲服務與 Amazon 通訊所需的程式庫。如需詳細資訊，請參閱Amazon 的開發支持 GameLift。

亞馬遜GameLift實時客戶端 SDK

實時客戶端 SDK 使遊戲客戶端能夠連接到實時服務器，加入遊戲會話並與其他玩家保持同步。下載 SDK 並了解有關使用實時服務器客戶端 API（C＃）進行 API 調用的更多信息。

亞馬遜GameLift遊戲

使AWS Management Console用 GameLift to Amazon 管理您的遊戲部署、設定資源，以及追蹤玩家使用情況和效能指標。Amazon 主GameLift控台提供 GUI 替代方案，可使用 AWS Command Line Interface (AWS CLI) 以程式設計方式管理資源。

AWS CLI

使用此命令列工具對 AWS SDK 進行呼叫，包括 Amazon GameLift API。若要取得有關使用的資訊AWS CLI，請參閱《使用指南》AWS CLI中的〈AWS Command Line Interface使用〉。

亞馬遜定價 GameLift

Amazon 會依使用期間對執行個體GameLift收費，以及依傳輸的資料量收取頻寬費用。有關亞馬遜的費用和價格的完整列表GameLift，請參閱亞馬遜GameLift定價。

有關計算託管遊戲或與 Amazon 配對的費用的資訊GameLift，請參閱生成亞馬遜GameLift定價估算，其中說明如何使用 AWS Pricing Calculator.

亞馬遜如何GameLift工作

本主題涵蓋遊戲託管的核心元件，並說明 Amazon 如何GameLift讓玩家可以使用您的多人遊戲伺服器。

準備好在亞馬遜上託管遊戲做好準備了GameLift嗎？ 退房亞馬遜GameLift託管路線圖。

關鍵元件

設置亞馬遜GameLift託管您的遊戲涉及使用以下組件。中的圖表Amazon 託管的遊戲架構 GameLift可視化這些組件之間的關係。

- 遊戲伺服器是指在叢集中執行的遊戲伺服器軟體。您將遊戲服務器構建或腳本上傳到亞馬遜GameLift並告訴亞馬遜GameLift。當您使用亞馬遜GameLiftAnywhere或亞馬遜 GameLift FleetIQ時，您可以將遊戲伺服器組建直接上傳到運算資源。
- 遊戲會話是一個正在進行的遊戲與玩家。您會定義遊戲工作階段的基本特性，例如，生命週期和玩家數量。然後，玩家連接到遊戲服務器以加入遊戲會話。
- 遊戲用戶端是您遊戲在玩家裝置上執行的軟體。遊戲用戶端會根據從 Amazon 接收到的連線資訊，透過後端服務連線到遊戲伺服器以加入遊戲工作階段GameLift。
- 後端服務是額外的自訂服務，可處理與 Amazon 相關的任務GameLift。最佳做法是，您的後端服務應該處理與 Amazon 的所有遊戲用戶端通訊GameLift。

託管遊戲伺服器

使用亞馬遜GameLift，您可以通過三種不同的方式託管遊戲服務器：亞馬遜託管GameLift，亞馬遜GameLift FleetIQ 和亞馬遜。GameLift Anywhere有關亞馬遜 GameLift FleetIQ 的更多信息，請參閱什麼是亞馬遜 GameLift FleetIQ？

您可以依照自己的遊戲需求設計機群。如需設計叢集的詳細資訊，請參閱亞馬遜GameLift車隊設計指南。

亞馬遜託管 GameLift

使用受管 AmazonGameLift，您可以在 Amazon GameLift 虛擬運算資源 (稱為執行個體) 上託管遊戲伺服器。建立一組執行個體並部署執行您的遊戲伺服器，藉此設定您的主機資源。

亞馬遜 GameLift Anywhere

使用 Amazon GameLiftAnywhere，您可以在您管理的運算上託管遊戲伺服器。透過建立參考您的運算的Anywhere叢集來設定您的主機資源。

機群別名

別名是您可以在艦隊之間轉移的指定，使其成為擁有一般叢集位置的便捷方式。您可以使用別名將遊戲客戶端從使用一個艦隊切換到另一個艦隊，而無需更改遊戲客戶端。您也可以建立指向內容的端子別名。

執行遊戲階段

在您將遊戲伺服器組建部署到叢集並且 Amazon 在每個執行個體上GameLift啟動遊戲伺服器程序之後，叢集就可以託管遊戲工作階段。當您的遊戲用戶端服務向後端服務或 Amazon 傳送放置請求時，Amazon 就會GameLift啟動新的遊戲工作階段GameLift。

遊戲工作階段配置和 FleetIQ 演算法

佇列使用 FleetIQ 演算法來選取可用的遊戲伺服器來主持新的遊戲工作階段。遊戲工作階段放置的關鍵元件是 Amazon GameLift 遊戲工作階段佇列。您可以為遊戲工作階段佇列指派一份艦隊清單，以決定佇列可以放置遊戲工作階段的位置。如需有關遊戲工作階段佇列以及如何為您的遊戲設計的詳細資訊，請參閱設計遊戲工作階段佇列。

玩家與遊戲的連線

作為遊戲工作階段放置過程的一部分，佇列或遊戲工作階段會提示選取的遊戲伺服器開始新的遊戲工作階段。遊戲服務器響應提示，並在準備好接受玩家連接GameLift時向亞馬遜報告。GameLift然後，Amazon 會將連線資訊傳送至後端服務或遊戲用戶端服務。您的遊戲用戶端會使用這些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

當艦隊在作用中並準備好主持遊戲工作階段時，您可以調整艦隊容量以滿足玩家的需求。我們建議您在所有快速找到遊戲和閒置資源的超支之間找到平衡。

Amazon GameLift 提供高效的自動擴展工具，或者您也可以手動設定叢集容量。如需詳細資訊，請參閱擴展亞馬遜GameLift託管容量。

Auto Scaling

亞馬遜GameLift提供了兩種自動擴展的方法：

- 基於目標的自動縮放
- 使用以規則為基礎的原則自動調整

其他擴展功能

- 遊戲工作階段保護 — 防GameLift止 Amazon 在縮小活動期間結束主持活躍玩家的遊戲工作階段。
- 擴展限制 — 透過設定叢集中執行個體數量的最小和最大限制來控制整體執行個體使用量。

- 暫停自動擴展 — 在叢集位置層級暫停自動調整規模，而無需變更或刪除自動擴展政策。
- 擴展指標 — 追蹤叢集的容量和擴展事件歷史記錄。

監控亞馬遜 GameLift

當您啟動並執行叢集時，Amazon GameLift 會收集各種資訊，以協助您監控已部署遊戲伺服器的效能。您可以使用這些資訊來最佳化資源使用、疑難排解問題，以及深入瞭解玩家在遊戲中的活躍狀態。亞馬遜GameLift收集以下內容：

- 艦隊、位置、遊戲工作階段和玩家工作階段詳細資訊
- 用量指標
- 伺服器程序健康
- 遊戲工作階段記

如需 Amazon 中監控的詳細資訊GameLift，請參閱監控 Amazon GameLift。

使用其他AWS資源

您的遊戲伺服器和應用程式可以與其他AWS資源通訊。例如，您可以使用一組 Web 服務進行玩家身份驗證或社交網絡。為了讓您的遊戲伺服器存取您AWS 帳戶管理的AWS資源，請明確允許 Amazon GameLift 存取您的AWS資源。

亞馬遜GameLift提供了幾個用於管理這種類型的訪問的選項。如需詳細資訊，請參閱 與您車隊的其他AWS資源進行溝通。

玩家如何連接到遊戲

遊戲工作階段是您在 Amazon 上執行的遊戲執行個體GameLift。若要進行遊戲，玩家可以尋找並加入現有的遊戲工作階段，或是建立新的遊戲工作階段並加入遊戲階段。玩家通過為遊戲會話創建一個玩家會話加入。如果遊戲工作階段對玩家開放，那麼 Amazon 會為玩家GameLift保留一個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，請參閱添加 Amazon GameLift 到您的遊戲客戶端。有關將播放器連接到實時服務器的信息，請參閱為即時伺服器整合遊戲用戶端。

亞馬遜GameLift提供了一些與遊戲和玩家會話相關的功能。

在多個位置的最佳可用資源上主持遊戲工作階段

設定 Amazon 如何選GameLift取資源來託管新遊戲工作階段時，請從多個選項中進行選擇。如果您在多個位置執行叢集，則可以設計遊戲工作階段佇列，將新的遊戲工作階段放置在任何叢集上，不論位置為何。

控制玩家對遊戲會話的訪問

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。

使用自訂的遊戲和玩家資料

添加自定義數據到遊戲會話對象和玩家會話對象。Amazon GameLift 會在開始新的遊戲工作階段時，將遊戲工作階段資料傳送至遊戲伺服器。當玩家連接到遊戲會話時，Amazon 會將玩家會話數據GameLift傳遞到遊戲服務器。

篩選和排序可用遊戲工作階段

使用工作階段搜尋和排序來尋找最適合潛在玩家的比賽，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。

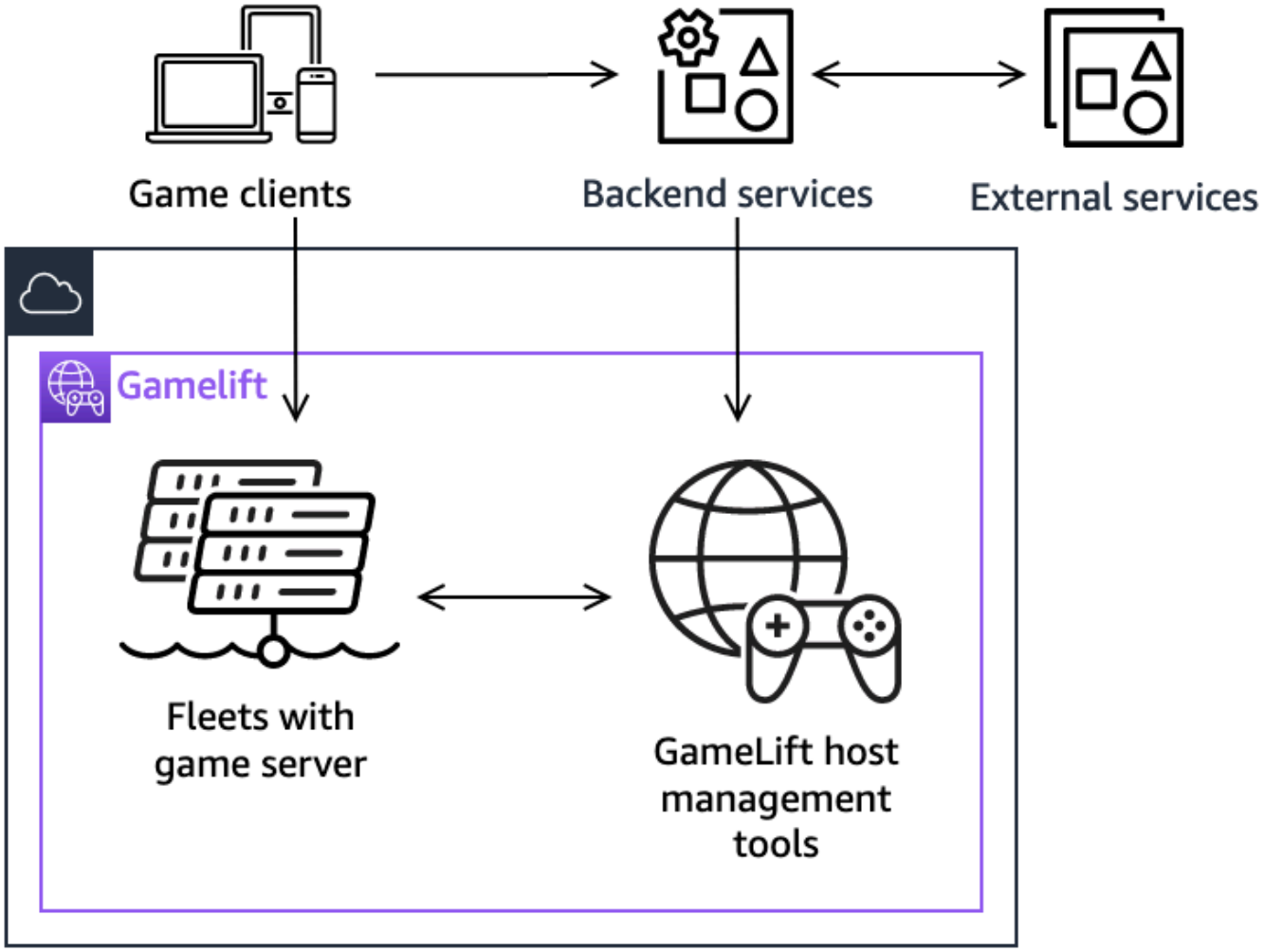
跟踪遊戲和玩家使用數據

自動儲存已完成遊戲工作階段的記錄。GameLift將 Amazon 整合到遊戲伺服器時，您可以設定日誌儲存。如需詳細資訊，請參閱在亞馬遜記錄服務器消息 GameLift。

使用 Amazon 主GameLift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。如需更多詳細資訊，請參閱 查看有關遊戲和玩家會話的數據 及 指標。

Amazon 託管的遊戲架構 GameLift

下圖說明使用受管 Amazon GameLift 解決方案託管之遊戲架構的關鍵元件。



該架構的關鍵組件包括以下內容：

遊戲客戶端

若要加入 Amazon 上託管的遊戲 GameLift，您的遊戲用戶端必須先找到可用的遊戲工作階段。遊戲用戶端會搜尋現有的遊戲工作階段、請求配對，或透過後端服務與 Amazon GameLift 通訊來開始新的遊戲工作階段。後端服務向 Amazon 發出請求 GameLift，服務會接收遊戲工作階段資訊，並將其轉送回遊戲用戶端。然後遊戲客戶端連接到遊戲服務器。如需詳細資訊，請參閱 為 Amazon 準備遊戲 GameLift。

後端服務

後端服務會呼叫 AWS SDK 中的 Amazon GameLift 服 GameLift 務 API 作業，以處理遊戲用戶端與 Amazon 之間的通訊。您還可以將後端服務用於其他遊戲特定任務，例如玩家身份驗證和授權，庫存或貨幣控制。如需詳細資訊，請參閱 設計您的遊戲用戶端服務。

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。外部服務可以通過後端服務和 Amazon 將信息傳遞到您的遊戲服務器 GameLift。

遊戲伺服器

您將遊戲伺服器軟體上傳到 Amazon GameLift， GameLift 然後 Amazon 將其部署到託管機器上，以主持遊戲工作階段並接受玩家連線。遊戲伺服器會與 Amazon 通訊， GameLift 以開始遊戲工作階段、驗證新連線的玩家，以及報告遊戲工作階段、玩家連線和可用資源的狀態。

自訂遊戲伺服器使用 Amazon 伺 GameLift 服 GameLift 器開發套件與 Amazon 通訊。 GameLift 透過後端服務從 Amazon 收到連線詳細資訊後，遊戲用戶端會直接連線到遊戲伺服器。如需詳細資訊，請參閱 將遊戲與自訂遊戲伺服器整合。

實時服務器是運行自定義腳本的遊戲服務器。加入遊戲時，遊戲客戶端使用實時客戶端 SDK 直接連接到實時服務器。如需詳細資訊，請參閱 將遊戲與 Amazon GameLift 即時伺服器整合。

主機管理工具

設定和管理主機資源時，遊戲擁有者會使用主機管理工具來管理遊戲伺服器組建或指令碼、叢集、配對和佇列。 AWS SDK 和主控台中的 Amazon GameLift 工具集提供多種管理託管資源的方式。您可以遠程訪問任何單個遊戲服務器進行故障排除。

設定

取得設定使用 Amazon GameLift 託管多人遊戲的協助。AWS 帳戶

> **Tip**
>
> 若要試用 Amazon GameLift 遊戲伺服器託管，請參閱開始使用亞馬遜 GameLift。

主題

設置一個 AWS 帳戶

要開始使用 Amazon GameLift，請創建並設置您的 AWS 帳戶. 建立一個 AWS 帳戶. 本節將逐步引導您建立帳戶、設定使用者以及設定權限。

主題

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 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，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console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)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 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 身分中心使用者登入的說明，請參閱使用AWS 登入 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 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。
2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

管理 Amazon 的用戶許可 GameLift

根據 Amazon GameLift 資源的需要，建立其他使用者或將存取權限延伸至現有使用者。最佳做法 (IAM 中的安全最佳做法) 是為所有使用者套用最低權限許可。如需有關權限語法的指導，請參閱亞馬遜的 IAM 許可示例 GameLift。

使用下列指示，根據您管理 AWS 帳戶中使用者的方式來設定使用者權限。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 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。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- 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 中的指示進行操作。
  - (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，最佳做法一律將權限附加到角色或使用者群組，而非個別使用者。

設定使用者的程式設計存取

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

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	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參

哪個使用者需要程式設計存取權？	到	By
		考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，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 存取金鑰的最佳做法。

為您的遊戲設定程式化存取

大多數遊戲使用後端服務 GameLift 使用 AWS SDK 與 Amazon 進行通信。例如，您使用後端服務 (代表遊戲用戶端行事) 來要求遊戲工作階段、將玩家置於遊戲中，以及其他工作。這些服務需要程式設計存取和安全登入資料，才能驗證對 Amazon GameLift 服務 API 的呼叫。

對於 Amazon GameLift，您可以在 AWS Identity and Access Management (IAM) 中建立播放器使用者來管理此存取權限。透過下列其中一個選項管理玩家使用者權限：

- 建立具有玩家使用者權限的 IAM 角色，並允許玩家使用者在需要時擔任該角色。在向 Amazon 發出請求之前，後端服務必須包含承擔此角色的程式碼 GameLift。根據安全性最佳做法，角色提供有限的暫時存取權限。您可以將角色用於在 AWS 資源 (IAM 角色) 或外部 (IAM 角色隨處) 上執行的 AWS 工作負載。
- 建立具有玩家使用者權限的 IAM 使用者群組，並將您的玩家使用者新增至群組。此選項為玩家使用者提供長期登入資料，後端服務在與 Amazon 通訊時必須存放和使用這些登入資料 GameLift。

如需使用權限原則語法，請參閱玩家使用者權限範例。

有關管理工作負載使用許可的詳細資訊，請參閱 IAM 身分：IAM 中的臨時登入資料。

亞馬遜的 IAM 許可示例 GameLift

使用這些範例中的語法為需要存取 Amazon GameLift 資源的使用者設定 AWS Identity and Access Management (IAM) 許可。如需管理使用者權限的詳細資訊，請參閱管理 Amazon 的用戶許可 GameLift。為 IAM Identity Center 以外的使用者管理許可時，最佳做法一律將許可附加到 IAM 角色或使用者群組，而非個別使用者。

如果您使用 Amazon GameLift FleetIQ 作為獨立解決方案，請參閱設定您AWS 帳戶的亞馬遜 GameLift FleetIQ。

管理員權限範例

這些範例可讓使用者完整存取管理 Amazon GameLift 遊戲託管資源。

Example 亞馬遜資GameLift源許可的語法

下面的示例擴展了對所有亞馬遜GameLift資源的訪問。

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

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

下列範例將存取權延伸至所有預設未啟用的 Amazon GameLift 資源和AWS區域。如需有關預設未啟用的區域以及如何啟用的詳細資AWS 區域訊，請參閱 AWS 一般參考.

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

Example 亞馬遜資GameLift源和**`PassRole`**許可的語法

下列範例可擴充對所有 Amazon GameLift 資源的存取權，並允許使用者將 IAM 服務角色傳遞給 Amazon GameLift。服務角色可讓 Amazon GameLift 有限的代表您存取其他資源和服務的能力，如中所述為 Amazon 設置 IAM 服務角色 GameLift。例如，在回應`CreateBuild`請求時，Amazon

GameLift 需要存取 Amazon S3 儲存貯體中的組建檔案。如需有關PassRole動作的詳細資訊，請參閱 IAM：將 IAM 角色傳遞給 IAM 使用者指南中的特定AWS服務。

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

玩家使用者權限範例

這些範例可讓後端服務或其他實體對 Amazon API 進行 GameLift 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場景。如需詳細資訊，請參閱 為您的遊戲設定程式化存取。

Example 遊戲工作階段放置權限的語法

下列範例擴充對 Amazon GameLift API 的存取權，這些 API 使用遊戲工作階段放置佇列建立遊戲工作階段和管理玩家工作階段。

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

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

Example 配對權限的語法

下列範例將存取權延伸至管理FlexMatch配對活動之 Amazon GameLift API 的存取權。FlexMatch匹配新的或現有的遊戲會話的玩家，並為亞馬遜GameLift上託管的遊戲啟動遊戲會話位置。如需詳細資訊FlexMatch，請參閱什麼是 Amazon GameLiftFlexMatch？

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

Example 手動遊戲工作階段放置權限的語法

下列範例擴充對 Amazon GameLift API 的存取權，這些 API 可在指定叢集上手動建立遊戲工作階段和玩家工作階段。這個案例支援不使用放置佇列的遊戲，例如讓玩家透過從可用遊戲工作階段清單中選擇加入的遊戲 (list-and-pick「」方法)。

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

```
    "Action": [
      "gamelift:CreateGameSession",
      "gamelift:DescribeGameSessions",
      "gamelift:SearchGameSessions",
      "gamelift:CreatePlayerSession",
      "gamelift:CreatePlayerSessions",
      "gamelift:DescribePlayerSessions"
    ],
    "Resource": "*"
  }
}
```

為 Amazon 設置 IAM 服務角色 GameLift

某些 Amazon GameLift 功能需要您將有限的存取權限擴展到您擁有的AWS資源。您可以透過建立 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，因為同樣是 AWS 身分，也有許可政策可決定該身分在 AWS 中可執行和不可執行的操作。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 受管叢集搭配使用的角色。如果您使用 Amazon GameLift FleetIQ 優化亞馬遜彈性運算雲端 (Amazon EC2) 執行個體上的遊戲託管服務，請參閱為 Amazon FleetIQ 設定您AWS 帳戶的遊戲。 GameLift

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

建立自訂 IAM 角色

步驟 1：建立權限原則。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 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) 在 AWS Management Console 中建立或編輯政策時，您可以產生可在 AWS CloudFormation 範本中使用的 JSON 或 YAML 政策範本。

   若要這麼做，請在 [原則編輯器] 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。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
11. 選擇 Create policy (建立政策) 儲存您的新政策。

第 2 步：創建一個 Amazon GameLift 可以承擔的角色。

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
2. 在 [選取信任的實體] 頁面上，選擇 [自訂信任原則] 選項。此選項會開啟 [自訂信任原則編輯器]。
3. 將預設 JSON 語法取代為下列項目，然後選擇 [下一步] 繼續。

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

```
    ]
}
```

4. 在 [新增權限] 頁面上，找出並選取您在步驟 1 中建立的權限原則。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說明 (選擇性)。檢閱信任實體和新增權限。
6. 選擇 [建立角色] 以儲存新角色。

權限原則語法

- Amazon 承擔服務角色的許 GameLift 可

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

- 存取預設未啟用的AWS區域的權限

```
{
  "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"
    }
  ]
}
```

Amazon 的開發支持 GameLift

Amazon GameLift 提供了一組 SDK，您可以將其與受管遊戲託管解決方案搭配使用。使用 Amazon GameLift SDK 將必要的功能新增至需要與 Amazon 主機服務互動的多人遊戲伺服器、遊戲用戶端和遊戲 GameLift 服務。

如需 Amazon GameLift 開發套件版本和開發套件相容性的最新資訊，請參閱Amazon GameLift 版本說明。

自訂遊戲伺服器

使用 Amazon 伺服器開發套件建立和部署 64 位元自訂遊戲伺 GameLift 服器。與伺服器 SDK 整合並部署進行託管的遊戲伺服器可以與 Amazon GameLift 服務通訊，以啟動和管理遊戲工作階段。如需有關整合伺服器 SDK 的資訊，請參閱中的主題為 Amazon 準備遊戲 GameLift。

開發作業系統

- Windows
- Linux

支持的編程語言

Amazon GameLift 提供以下語言的服務器開發套件。 下載伺服器 SDK。 如需版本特定的詳細資訊，請參閱每個套件中隨附的 Readme 檔案。

- C++ 伺服器 SDK
  - SDK 參考資料
  - SDK 整合
- C# 服務器 SDK（版本可能支持 .NET 4 和 .NET 6）
  - SDK 參考資料
  - SDK 整合

- Go
  - SDK 參考資料
  - SDK 整合

支援的遊戲引擎

在任何支援 C++、C# 或 Go 程式庫的引擎上使用特定語言的 SDK。此外，Amazon GameLift 提供了這些遊戲引擎插件：下載 Amazon GameLift 插件

- 团结
  - C# 伺服器 SDK 外掛程式的 Unity 是一個輕量級的外掛程式，您可以使用 Unity 套件管理器安裝預先建置的程式庫。將此外掛程式與下列版本搭配使用：適用於視窗和 Mac 作業系統的 2020.3 LTS、2021.3 LTS 和 2022.3 LTS。它支持統一的 .NET 框架和 .NET 標準配置文件，使用 .NET 標準 2.1 和 .NET 4.x。
    - GameLift 將亞馬遜整合到統一項目中
  - Unity 2021.3 LTS 和 2022.3 LTS 的獨立插件是一個功能齊全的插件，其中包含針對統一和 GUI 元素構建的 C # SDK 庫，用於配置和部署 Amazon GameLift 資源進行託管。
    - Amazon GameLift 插件統一指南服務器 SDK 5.x
    - C# 的亞馬遜GameLift伺服器 SDK 參考
- 虛幻引擎
  - 用於虛幻的 C ++ 服務器 SDK 插件是一個輕量級的插件，由 C ++ 虛幻源代碼組成，您可以將其構建到庫中以與虛幻引擎版本 4，5 和 5.1 一起使用。
    - GameLift 將 Amazon 集成到虛幻引擎項目中
    - 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考
  - 虛幻引擎 5.0，5.1 和 5.2 的獨立插件是一個功能齊全的插件，具有用於虛幻服務器 SDK 庫和 SDK 的 C ++。 AWS 該插件安裝在虛幻編輯器中，其中包含用於配置和部署用於託管的 Amazon GameLift 資源的 UI 元素和支持材料。
    - 將遊戲與虛幻引擎的 Amazon GameLift 插件集成
    - 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考

遊戲伺服器作業系統

使用 Amazon GameLift 伺服器開發套件建置遊戲伺服器，以便在下列平台上執行：

- Windows Server 2016

- Amazon Linux 2023
- Amazon 亞馬遜 2
- 視窗伺服器 2012 (請參閱 Amazon GameLift 常見問題解答視窗 2012)
- Amazon Linux（AL1）（請參閱 Amazon GameLift 常見問題解答 AL 1）

針對自訂用戶端服務

使用 AWS 開發套件搭配 Amazon GameLift API 建立 64 位元自訂用戶端服務。此 SDK 可讓用戶端服務管理遊戲工作階段，並將玩家加入 Amazon 上託管的遊戲 GameLift。若要開始使用，請下載 AWS SDK。如需將開發套件與 Amazon 搭配使用的詳細資訊 GameLift，請參閱 Amazon GameLift API 參考資料。

對於實時服務器

配置和部署實時服務器以託管您的多人遊戲。若要允許您的遊戲用戶端連線到即時伺服器，請使用 Amazon GameLift 即時用戶端 SDK。遊戲客戶端使用此 SDK 與實時服務器以及連接到服務器的其他遊戲客戶端交換消息。若要開始使用，請下載 Amazon GameLift 即時用戶端開發套件。如需組態資訊，請參閱為即時伺服器整合遊戲用戶端。

開發套件支援

實時客戶端 SDK 包含以下語言的源：

- C# (.NET)

開發環境

根據需要從原始碼為下列支援的開發作業系統和遊戲引擎建置 SDK：

- 操作系統 — 視窗, Linux 的, 安卓系統, iOS 版
- 遊戲引擎-Unity，支持 C＃ 庫的引擎

遊戲伺服器作業系統

您可以將實時服務器部署到在以下平台上運行的託管資源上：

- Amazon Linux
- Amazon Linux 2

管理您的遊戲託管成本

您的AWS帳單反映了您的遊戲託管成本。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/billing/。有關幫助您管理AWS成本的工具和資源的更多信息，請參閱用AWS Billing戶指南。本指南可協助您檢閱資源使用量、建立 future 的使用情況，以及判斷擴充需求。

特別是，請考慮這些技巧來幫助您管理 Amazon GameLift 服務的成本。

建立帳單警示以監控使用情況

設定AWS免費方案使用量警示，以便在您的使用量接近或超過 Amazon GameLift 和其他AWS 服務產品的免費方案限制時通知您。您可以設定警示，以根據您的使用量層級採取行動。例如，當達到免費方案限制時，您可以自動將預算設為零。

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，請參閱使AWS Billing用指南中的下列主題：

- 追蹤您的AWS免費方案使用量
- 帳單提示偏好設

跟踪每個 Amazon GameLift 車隊的成

使用AWS成本分配標籤，根據 Amazon GameLift Amazon EC2 叢集和其他 EC2 資源來組織和追蹤遊戲託管成本。透過個別或按群組標記您的叢集，您可以建立成本分配報告，根據指派的標籤對成本進行分類。您可以使用這種類型的報告來確定艦隊如何為您的託管成本帶來貢獻。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, 使用AWS Billing者指南
- 使用 AWSCost Explorer, AWS Cost Management使用者指南分析您的成本

將未使用的叢集容量設為零

即使沒有使用託管遊戲會話，艦隊也可能會繼續產生成本。為避免產生不必要的費用，請在不使用時將您的機隊縮小到零。如果您使用 auto 動擴展，請暫停此活動並手動設定叢集容量。

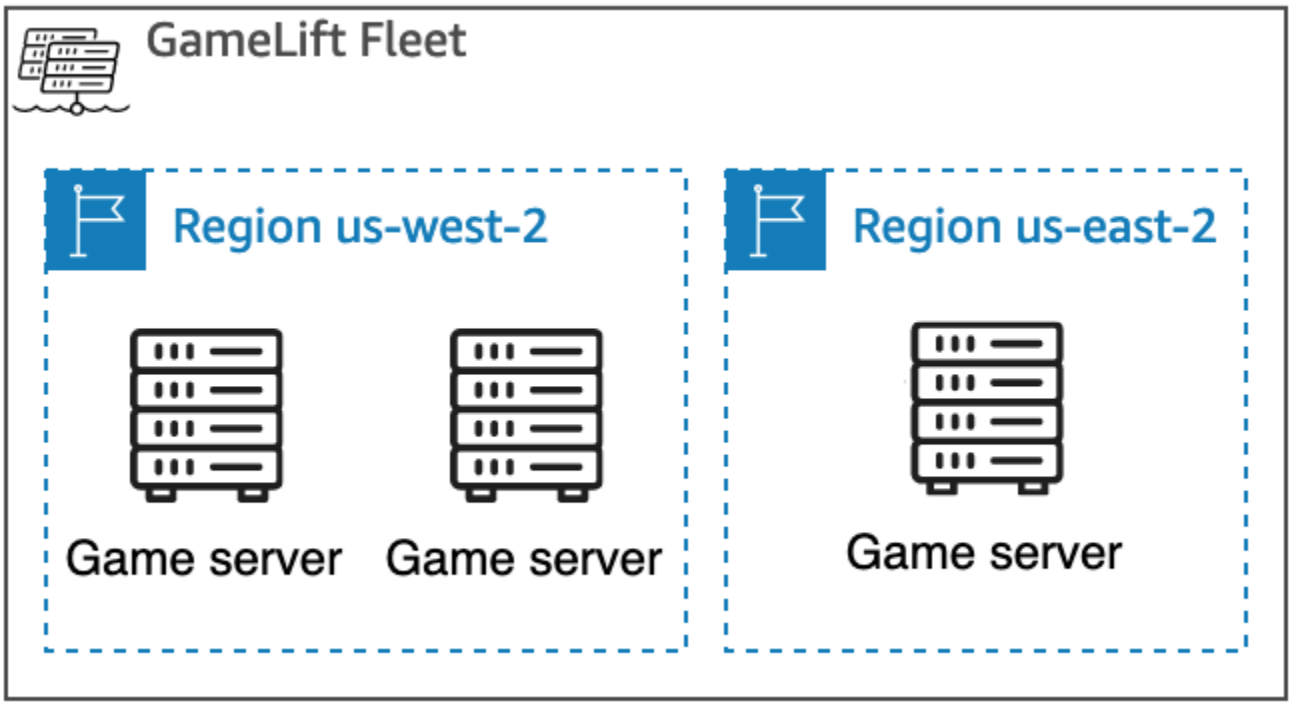
Amazon GameLift 託管地點

Amazon GameLift 在多個區域AWS 區域和 Local Zones 提供。如需位置的完整 Amazon GameLift 單，請參閱 AWS 一般參考.

Amazon GameLift 託管

當您創建 Amazon GameLift 車隊時，Amazon GameLift 會在您當前的車隊中創建該艦隊的資源AWS 區域。Amazon GameLift 稱該區域為艦隊的本地區域。要管理車隊，請從其所在地區訪問它。

除了叢集的本地區域以外，多地點叢集還會將執行個體部署到其他位置。透過多地點叢集，您可以個別管理每個位置的容量，也可以按地點放置遊戲工作階段。 多地點叢集可以在 Amazon GameLift 支援的任何區域或本地區域中擁有遠端位置。下圖說明在兩個區域中具有資源的多地點叢集。在圖中，`us-west-2`區域包括兩個遊戲伺服器，而該`us-east-2`地區有一個遊戲伺服器。



如果您選擇將多位置叢集與區域中的執行個體搭配使用預設未啟用的執行個體，則必須在您AWS 帳戶的. 此外，您的 Amazon GameLift 管理員政策必須允許該`ec2:DescribeRegions`動作。如需有關預設未啟用的區域以及如何啟用的詳細資AWS 區域訊，請參閱 AWS 一般參考. 如需預設未啟用區域的政策範例，請參閱管理員權限範例。

> **Important**
>
> 若要使用預設未啟用的區域，請在您的AWS 帳戶.
>
> - 您在 2022 年 2 月 28 日之前建立的未啟用區域的艦隊將不會受到影響。
> - 若要建立新的多位置叢集或更新現有的多位置叢集，請先啟用您選擇使用的任何區域。

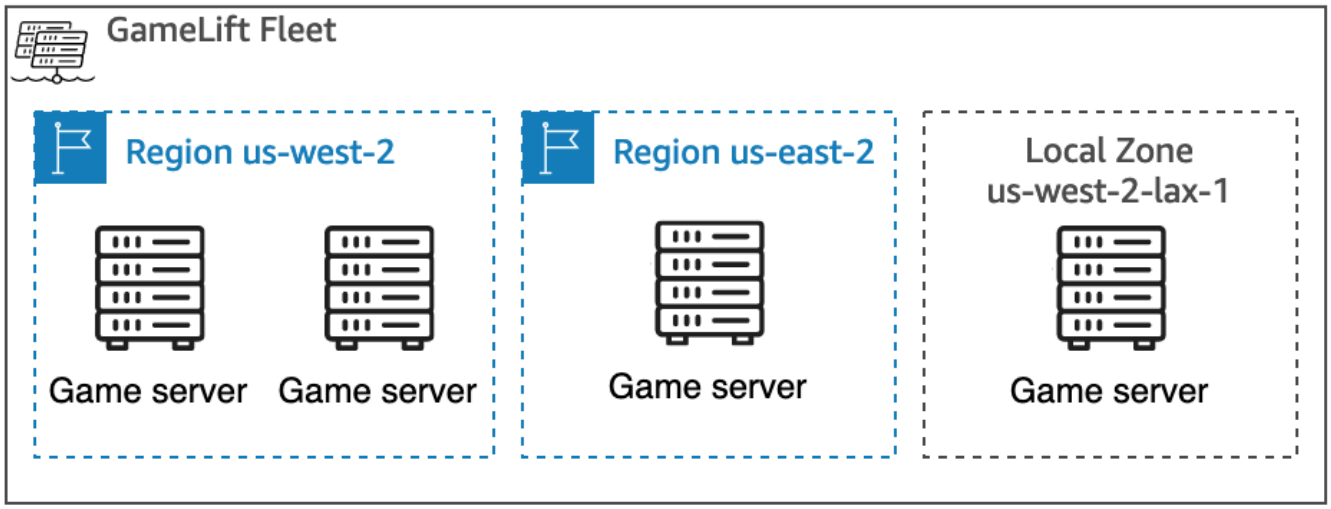
對於遊戲工作階段放置，您可以在 Amazon GameLift 支援的任何位置建立遊戲工作階段佇列。Amazon 會從您建立佇列的位置 GameLift 放置遊戲工作階段。

本機區域

「本機區域」是您使用者所AWS 區域在地理位置上的延伸。Local Zones 有自己的互聯網連接。Local Zones 也支援，AWS Direct Connect因此在本機區域中建立的資源可以透過低延遲通訊為本機使用者提供服務。如需詳細資訊，請參閱 AWS Local Zones。

本地區域的代碼是其區域代碼，後跟一個標識符，表示其物理位置。例如，「本`us-west-2-lax-1`地區域」位於洛杉磯。如需可用 Local Zones 的清單，請參閱可用 Local Zones。

Amazon 會在您為叢集選擇的每個位置 GameLift 託管您的遊戲。下圖描述了在該地區擁有兩個遊戲伺服器的艦隊，`us-west-2`區域中有一個遊戲伺服器，以及一部在`us-west-2-lax-1`本地區`us-east-2`域中的遊戲伺服器。



可用 Local Zones

下表列出了可用的 Local Zones 及其實體位置。

本地區域	位置 (都會區)	
us-east-1-atl-1	亞特蘭大	
us-east-1-chi-1	芝加哥	
us-east-1-dfw-1	達拉斯	
us-east-1-iah-1	休士頓	
us-east-1-mci-1	堪薩斯市	
us-west-2-den-1	丹佛	
us-west-2-lax-1	洛杉磯	
us-west-2-phx-1	Phoenix	

Amazon GameLift Anywhere

您可以使用 Amazon GameLift Anywhere 使用自己的硬體建立車隊，並使用 Amazon GameLift 管理遊戲組建、指令碼、遊戲伺服器和用戶端。Amazon GameLift Anywhere 在 Amazon GameLift 支持的所有區域都可以使用。如需建立Anywhere叢集和測試遊戲伺服器整合的詳細資訊，請參閱創建一個 Amazon GameLift Anywhere 車隊和使用亞馬遜機GameLiftAnywhere隊測試您的整合。

使用 Amazon， GameLift Anywhere您可以建立自訂位置，代表您用來託管 Amazon GameLift 整合式遊戲伺服器的硬體實體位置。

Amazon GameLift FlexMatch

對於 FlexMatch，您可以在 Amazon GameLift 支援的任何位置託管比賽產生的遊戲工作階段。實際的配對活動會AWS 區域在您選擇建立分房系統資源的地方進行。Amazon GameLift 路線會將請求與分房系統相符，並在該位置處理它們。有關 Amazon 的更多信息 GameLift FlexMatch，請參閱 Amazon 是什麼 GameLift FlexMatch？

AWS 區域支援 FlexMatch 資源

Amazon GameLift 在中國

使 GameLift 用 Amazon 購買由 Sinnet 運營的中國（北京）區域或由 NWCD 運營的中國（寧夏）區域的資源時，您必須擁有一個單獨的AWS（中國）帳戶。請注意，部分功能無法在中國地區使用。如需 GameLift 在這些區域中使用 Amazon 的詳細資訊，請參閱下列資源：

- Amazon Web Services 在中國
- Amazon GameLift（在中國開始使用 Amazon Web Services）

開始使用亞馬遜 GameLift

我們建議您在將 Amazon 用GameLift於自己的遊戲之前嘗試以下示例。自訂遊戲伺服器範例提供您在 Amazon GameLift 主控台中託管遊戲的經驗。實時服務器示例向您展示瞭如何準備遊戲以使用實時服務器進行託管。

要開始使用亞馬遜GameLift為您自己的遊戲，請參閱亞馬遜GameLift託管路線圖。

自訂遊戲伺服器範例

這個例子演示了亞馬遜上的實時自定義遊戲GameLift。此範例會引導您完成下列步驟：

- 建立範例遊戲組建。
- 創建一個運行遊戲服務器的艦隊。
- 從示例遊戲客戶端連接到遊戲服務器。
- 查看艦隊和遊戲會話指標。

完成這些步驟後，您可以啟動多個遊戲客戶端並玩遊戲以生成託管數據。然後，您可以瀏覽 Amazon GameLift 主控台以檢視託管資源、追蹤指標，並嘗試擴展託管容量的方法。

要開始使用，請登錄到亞馬遜GameLift控制台。

實時服務器示例遊戲

這個例子是一個名為 Mega Frog Race 的完整多人遊戲，包括源代碼。該示例顯示了如何將您的遊戲客戶端與實時服務器集成。您也可以使用此示例遊戲作為起點，以嘗試其他 Amazon GameLift 功能，例如FlexMatch。

如需實際操作教學課程，請參閱 FOR Games 部落格JavaScript上的僅用幾行AWS為多人遊戲建立伺服器。

有關巨型青蛙競賽的源代碼，請參閱存GitHub儲庫。

源代碼包括以下部分：

- 遊戲用戶端 — C++ 遊戲用戶端的原始程式碼，在 Unity 中建立。遊戲客戶端獲取遊戲會話連接信息，連接到服務器，並與其他玩家交換更新。

- 後端服務 — 管理 Amazon 直接 API 呼叫之AWS Lambda函數的原始程式碼GameLift。
- 實時腳本 — 為遊戲配置實時服務器群的源腳本文件。此指令碼包含即時伺服器與 Amazon 通訊 GameLift和託管遊戲所需的最低組態。

亞馬遜GameLift託管路線圖

本主題可協助您從工作階段型多人遊戲的不同 Amazon GameLift 託管選項中進行選擇。本節的其餘主題將逐步引導您瞭解如何將 Amazon 用GameLift於託管主機。

在您開始準備將遊戲推出生產之前，請填寫上市問卷以開始與 Amazon GameLift 團隊合作。

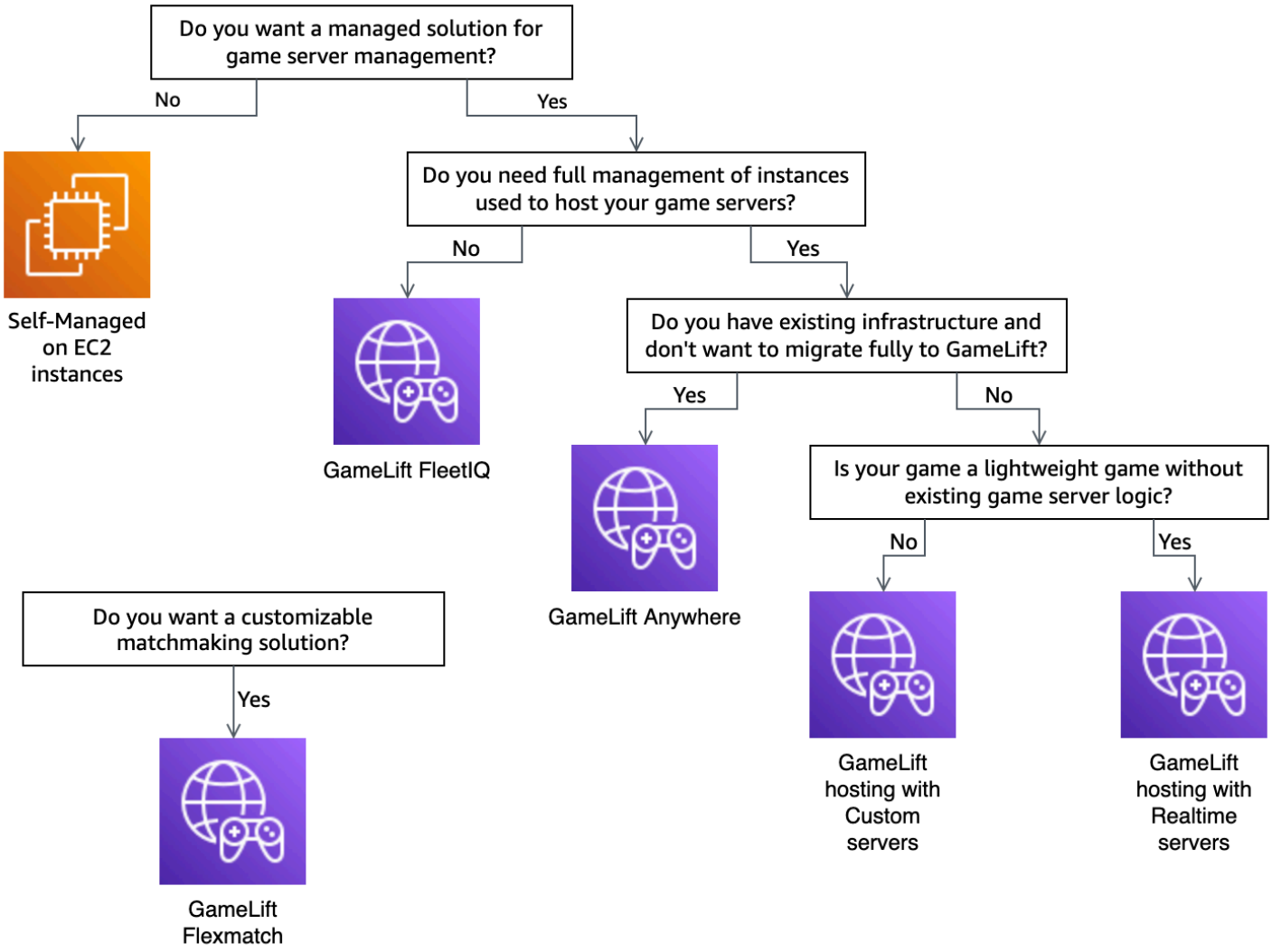
主題

選擇託管選項

以下流程圖提出問題，引導您找到適合您使用案例的正確 Amazon GameLift 解決方案。

1. 您想要遊戲伺服器管理的受管理解決方案嗎？
   - 是 — 繼續執行步驟 2。
   - 否 — 請考慮 Amazon EC2 執行個體上的自我管理遊戲伺服器。
2. 您是否需要完全控制託管遊戲伺服器的執行個體？
   - 是的-考慮亞馬遜 GameLift FleetIQ。
   - 否 — 繼續執行步驟 3。
3. 您是否擁有要與 Amazon 搭配使用的現有基礎設施GameLift？
   - 是的-考慮亞馬遜GameLiftAnywhere。
   - 否 — 繼續執行步驟 4。
4. 您的遊戲輕量級是否沒有現有的遊戲伺服器邏輯
   - 是-考慮實時服務器。

- 否 — 考慮使用自訂伺服器。



以下是有關流程圖中提到的一些 Amazon GameLift 託管選項的更多信息：

亞馬遜 GameLift Anywhere

利GameLiftAnywhere用 Amazon GameLift 管理工具在您自己的硬體上託管您的遊戲。您也可以使用Anywhere叢集反覆測試您的遊戲伺服器。如需詳細資訊，請參閱創建一個 Amazon GameLift Anywhere 車隊。

亞馬遜託管 GameLift

託管亞馬遜GameLift託管有兩種選擇：

自訂伺服器 — Amazon GameLift 託管執行遊戲伺服器二進位檔的自訂伺服器。

即時伺服器 — Amazon GameLift 託管您的輕量級遊戲伺服器。

亞馬遜GameLift花旗

在流程圖中，升降機和班次移轉是指您無法變更遊戲架構時的移轉。使用 Amazon GameLift FleetIQ 只需對現有部署進行較少的變更，並提供 Amazon GameLift 工具進行叢集管理。如需詳細資訊，請參閱 Amazon GameLift FleetIQ 開發人員指南。

如果您決定使用亞馬遜GameLiftAnywhere或亞馬遜託管GameLift，請繼續準備您的亞馬遜遊戲GameLift。

準備您的亞馬遜遊戲 GameLift

本主題說明準備多人遊戲以與受管 Amazon GameLift 託管整合的步驟。為了準備你的遊戲，你必須激活它和亞馬遜之間的通信GameLift。

準備您的自訂遊戲伺服器

若要啟動和停止遊戲工作階段以及執行其他任務，遊戲伺服器必須能夠通GameLift知 Amazon 其狀態。若要啟動與 Amazon 的通訊GameLift，請將程式碼新增至您的遊戲伺服器專案。如需詳細資訊，請參閱將遊戲與自訂遊戲伺服器整合。

1. 準備您的自定義遊戲服務器以在亞馬遜上託管GameLift。
   - 取得 Amazon GameLift 伺服器開發套件，並針對您偏好的程式設計語言和遊戲引擎進行建置。
   - 將代碼添加到您的遊戲服務器項目以激活與亞馬遜的通信GameLift。
2. 準備好您的遊戲用戶端連線到 Amazon GameLift 託管的遊戲工作階段。
   - 將 AWS SDK 添加到後端服務和遊戲客戶端項目中。如需詳細資訊，請參閱下載用戶端服務的 Amazon GameLift 開發套件。
   - 新增功能以擷取有關遊戲工作階段的資訊、放置新的遊戲工作階段，以及在遊戲工作階段中為玩家預留空間。
   - （可選）用FlexMatch於玩家配對。如需詳細資訊，請參閱與 Amazon GameLift 主機FlexMatch整合。

準備您的實時服務器

Amazon GameLift 即時伺服器提供輕量型伺服器解決方案，您可以根據自己的遊戲進行設定。即時伺服器提供 Amazon 為遊戲伺服器提GameLift供的優點相同，但遊戲伺服器可自訂性降低。

創建用於在亞馬遜上託管的實時腳本GameLift。

實時腳本包含您的服務器配置和可選的自定義遊戲邏輯。即時伺服器旨在啟動和停止遊戲工作階段、接受玩家連線，以及管理與 Amazon 以GameLift及遊戲中玩家之間的通訊。您還可以為遊戲添加自定義服務器邏輯的掛鉤。實時服務器使用 Node.js 和JavaScript. 如需詳細資訊，請參閱 創建實時腳本 及 測試您與亞馬遜的整合 GameLift。

測試您與亞馬遜的整合 GameLift

亞馬遜在測試遊戲服務器時GameLift支持快速迭代。本主題會引導您完成可用的測試類型。

自訂遊戲伺服器

使用 Amazon GameLift 將環境中任何位置的硬體整合到 Amazon GameLift 遊戲託管架構中。Amazon會GameLift在Anywhere叢集中向 Amazon GameLift Anywhere 註冊您的硬體，以便您可以使用自己的本機開發電腦進行測試。如需使用 Amazon 進行測試的詳細資訊 GameLiftAnywhere，請參閱使用亞馬遜機GameLiftAnywhere隊測試您的整合。如需使用 Amazon GameLift Anywhere 透過現場部署解決方案託管遊戲的詳細資訊，請參閱選擇 Amazon GameLift 運算資源。

即時伺服器

使用即時伺服器，您可以隨時更新指令碼。當您更新實時腳本時，Amazon GameLift 會在幾分鐘內將新版本分發到您的託管資源。Amazon GameLift 部署新指令碼後，所有新遊戲工作階段都會使用新的指令碼版本。Amazon GameLift 部署新指令碼後，您可以立即開始測試。有關實時服務器的更多信息，請參閱，將遊戲與 Amazon GameLift 即時伺服器整合

規劃和部署您的亞馬遜GameLift資源

使用下列秘訣來協助您規劃全球 Amazon GameLift 資源部署。如需有關在哪裡可以透過 Amazon 託管遊戲的資訊GameLift，請參閱Amazon GameLift 託管地點。

若要部署 Amazon 資GameLift源，請完成以下任務：

- 打包並上傳您的遊戲服務器到亞馬遜GameLift或您的硬件。將伺服器上傳到 Amazon 時GameLift，您只會將伺服器上傳到艦隊所在地。AWS 區域Amazon GameLift 會自動將伺服器分配到叢集中的其他位置。如需詳細資訊，請參閱將構建和腳本上傳到亞馬遜 GameLift。

- 為您的遊戲設計和部署 Amazon GameLift 叢集。決定要使用的運算資源類型、要部署到哪些位置、是否使用佇列，以及其他選項。如需詳細資訊，請參閱亞馬遜GameLift車隊設計指南。
- 建立佇列以管理新的遊戲工作階段放置位置和 Spot 執行個體策略。如需詳細資訊，請參閱設計遊戲工作階段佇列。
- 使用自動擴展來管理您的車隊的託管容量，以滿足預期的玩家需求。如需詳細資訊，請參閱擴展亞馬遜GameLift託管容量。
- 為您的遊戲使用FlexMatch配對規則。如需詳細資訊，請參閱與 Amazon GameLift 主機FlexMatch整合。

自動部署您的亞馬遜GameLift資源

為了簡化 Amazon GameLift 資源的全球部署，我們建議您使用基礎設施即程式碼 (IaC) 來定義資源。由於 Amazon GameLift 支援AWS CloudFormation範本，因此您可以在範本中為任何部署特定組態設定參數。

若要管理AWS CloudFormation堆疊的部署，我們也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 這些功能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。

以下是您可以使用 CI/CD 工具或服務自動化的新遊戲伺服器版本的 Amazon GameLift 資源部署的一些常見步驟：

- 構建和測試遊戲服務器二進製文件。
- 將二進製文件上傳到亞馬遜GameLift或您的硬件。
- 在新版本中部署新艦隊。
- 部署新叢集後，從 Amazon GameLift 佇列中移除舊版的叢集，並將其取代為新叢集。
- 之前版本的艦隊成功結束所有遊戲會話後，刪除這些艦隊的AWS CloudFormation堆棧。

您也可以使用AWS Cloud Development Kit (AWS CDK)來定義您的亞馬遜GameLift資源。如需關於 AWS CDK 的詳細資訊，請參閱《AWS Cloud Development Kit (AWS CDK) 開發人員指南》。

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，以驗證玩家並與 Amazon API 進行通訊。GameLift透過實作自訂遊戲用戶端服務，您可以：

- 為您的玩家自訂驗證。

- 控制 Amazon 如何GameLift配對並開始遊戲工作階段。
- 使用您的玩家數據庫獲得玩家屬性，例如配對技能評級，而不是信任客戶端。

使用遊戲用戶端服務也可降低直接與 Amazon GameLift API 互動的遊戲用戶端所帶來的安全風險。

驗證您的玩家

您可以使用 Amazon Cognito 和玩家工作階段 ID 來驗證您的遊戲用戶端。若要管理播放器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，建置自訂身分識別解決方案並將其託管在上面AWS。您也可以透過 API 閘道使用 Lambda 授權器進行自訂授權邏輯。

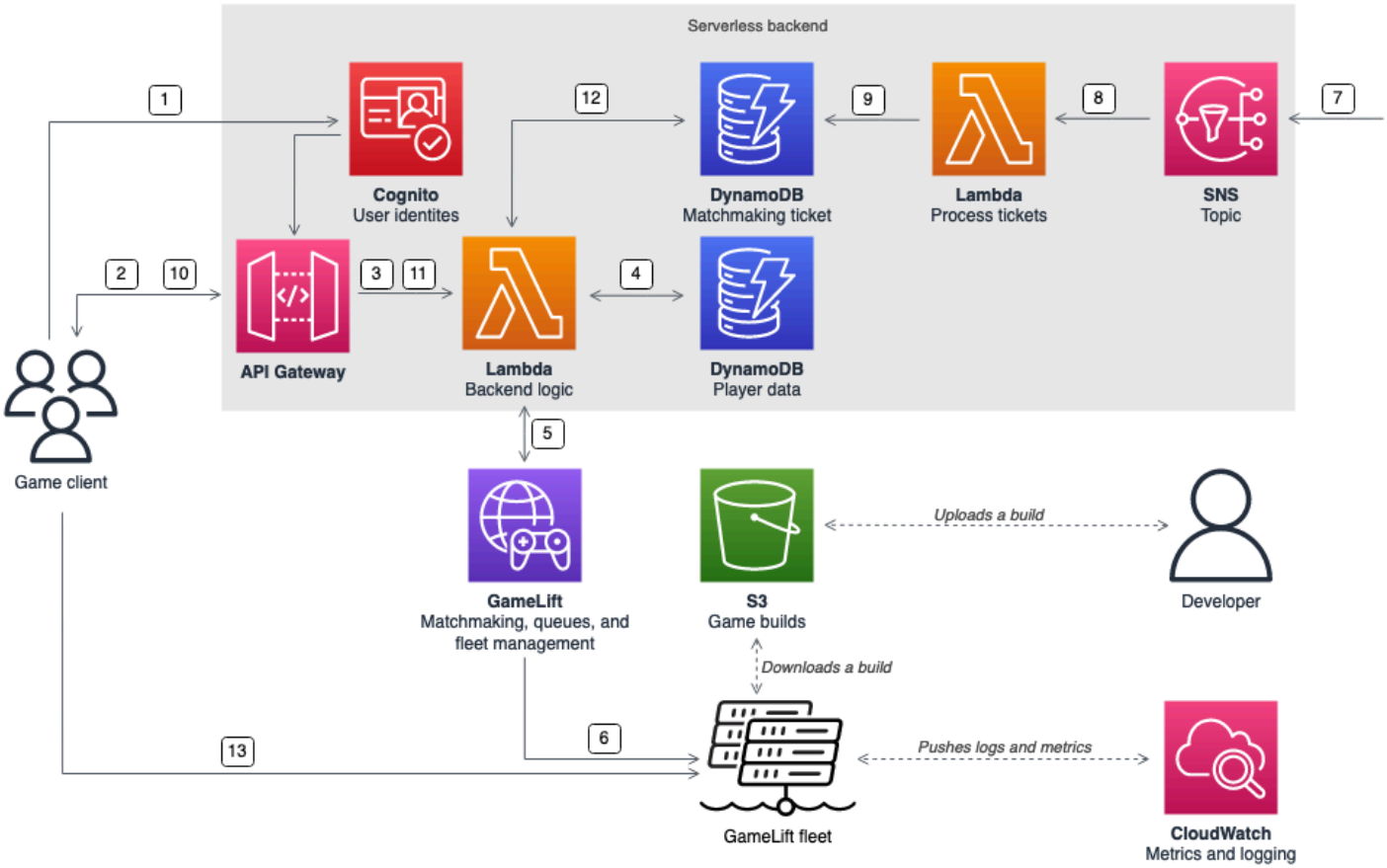
其他資源：

- 使用身分集區 (聯合身分) (Amazon Cognito 開發人員指南)
- 開始使用使用者集區 (Amazon Cognito 開發人員指南)
- 如何使用亞馬遜 Cognito 設置玩家身份驗證（AWS適用於遊戲博客）

具有無伺服器後端功能的獨立遊戲工作階段

使用無伺服器用戶端服務架構，後端可以從可高度擴展的資料庫檢視配對票證的狀態，而不是直接存取 Amazon GameLift API。

下圖顯示使用建置的無伺服器後端，AWS 服務可將玩家與 Amazon GameLift 叢集上執行的遊戲相符。下列清單提供圖表中每個編號圖說明的描述。若要試用此範例，請參閱開啟AWS的多人工作階段型遊戲託管。GitHub



1. 遊戲用戶端從 Amazon Cognito 身分識別集區請求 Amazon Cognito 使用者身分。
2. 遊戲用戶端會接收臨時存取登入資料，並透過 Amazon API 閘道 API 要求遊戲工作階段。
3. API 閘道會叫用AWS Lambda函式。
4. Lambda 函數會從亞馬遜動態 B NoSQL 表請求播放器資料。此函數會在請求內容資料中提供 Amazon Cognito 身分識別。
5. Lambda 函數透過亞馬遜配GameLiftFlexMatch對要求進行比對。
6. FlexMatch匹配一組具有適當延遲時間的玩家，然後通過 Amazon GameLift 隊列請求遊戲會話放置。佇列中有一或多個AWS 區域位置的車隊。
7. Amazon 將工作階段GameLift置於叢集的其中一個位置後，Amazon 會GameLift向 Amazon 簡單通知服務 (Amazon SNS) 主題傳送事件通知。
8. Lambda 函數會接收亞馬遜 SNS 事件並進行處理。
9. 如果配對票證是一個`MatchmakingSucceeded`事件，則 Lambda 函數會將結果連同遊戲伺服器的連接埠和 IP 位址寫入 DynamoDB 表格。
10. 遊戲用戶端向 API Gateway 發出已簽署的要求，以便在特定時間間隔內檢視配對票證的狀態。
11. API 閘道使用 Lambda 函數來檢查配對票證狀態。

12Lambda 函數會檢查 DynamoDB 表格，以查看工單是否成功。如果成功，該功能會將遊戲伺服器的連接埠和 IP 位址以及玩家工作階段 ID 傳回給用戶端。如果票證尚未成功，該函數將發送一個響應，驗證匹配尚未準備好。

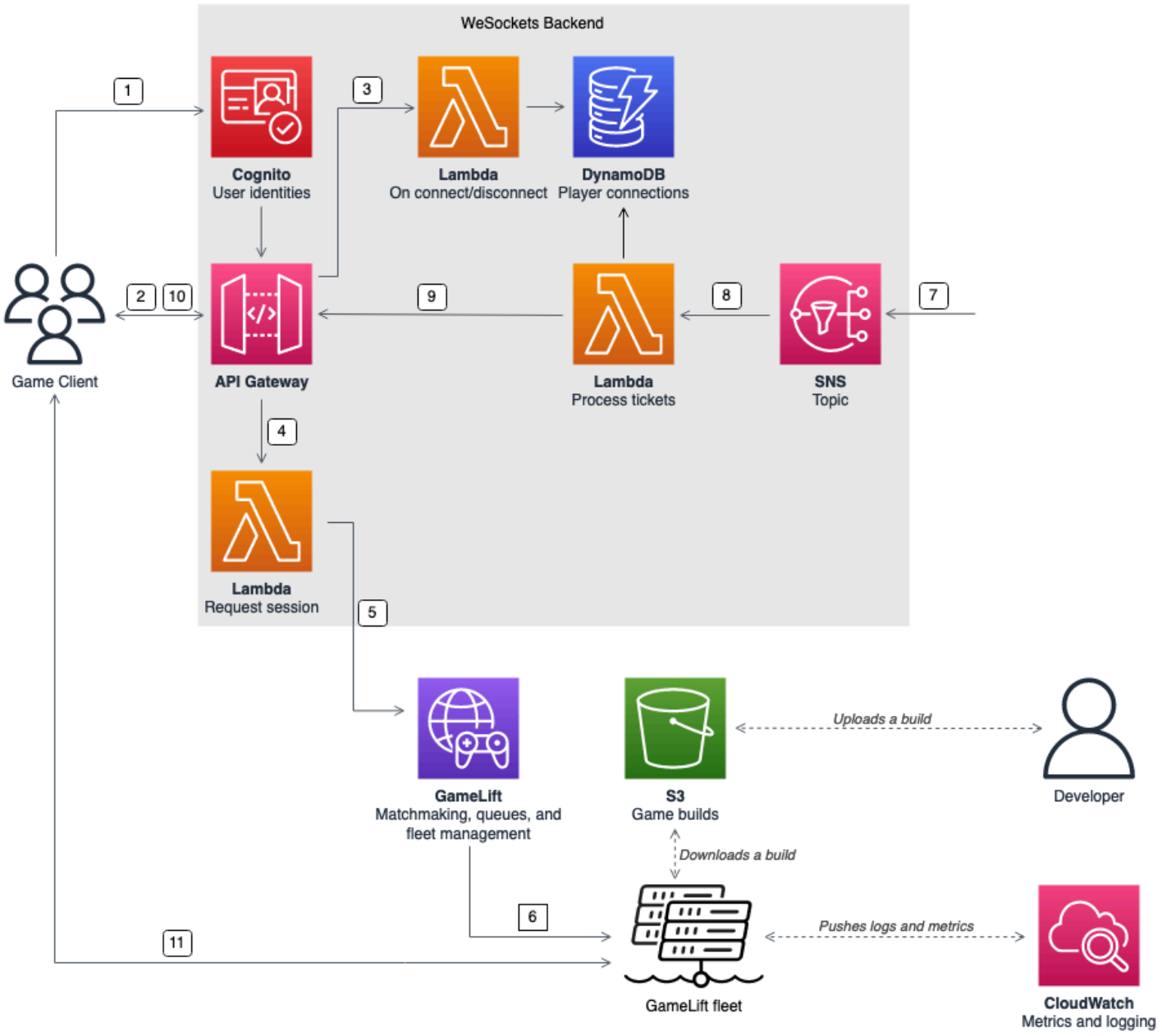
13遊戲用戶端會使用後端服務提供的連接埠和 IP 位址，使用 TCP 或 UDP 連線到遊戲伺服器。然後遊戲用戶端會將玩家工作階段 ID 傳送到遊戲伺服器，然後再使用 Amazon GameLift Server SDK 驗證 ID。

具有WebSocket基於後端的獨立遊戲會話服務器

使用 Amazon API 閘道架WebSocket構，您可以使用伺服器啟動的訊息發出配對請求，WebSockets並傳送推播通知以完成配對。這種架構通過在客戶端和服務器之間具有雙向通信來提高性能。

如需使用 API 閘道 WebSock API 的詳細資訊，請參閱使用 WebSocket API。

下圖顯示WebSocket基於後端架構，該架構使用 API Gateway 和其他AWS 服務方式將玩家與在 Amazon GameLift 叢集上執行的遊戲進行比對。下列清單提供圖表中每個編號圖說明的描述。



1. 遊戲用戶端從 Amazon Cognito 身分識別集區請求 Amazon Cognito 使用者身分。
2. 遊戲用戶端會使用亞馬遜認證簽署 API 閘道 API 的WebSocket連線。
3. API 閘道會在連線上呼叫AWS Lambda函式。此函數會將連線資訊儲存在 Amazon DynamoDB 表格中。
4. 遊戲用戶端透過 API 閘道 API 透過WebSocket連線傳送訊息至 Lambda 函數，以要求工作階段。
5. Lambda 函數會接收訊息，然後透過亞馬遜配GameLiftFlexMatch對請求比對。
6. FlexMatch匹配一組玩家後，FlexMatch請通過 Amazon GameLift 隊列請求遊戲會話放置。

7. Amazon 將工作階段GameLift置於叢集的其中一個位置後，Amazon 會GameLift向 Amazon 簡單通知服務 (Amazon SNS) 主題傳送事件通知。
8. Lambda 函數會接收亞馬遜 SNS 事件並進行處理。
9. 如果配對票證是一個`MatchmakingSucceeded`事件，則 Lambda 函數會向 DynamoDB 要求正確的玩家連線。然後，函數會透過 API 閘道 API 透過WebSocket連線傳送訊息給遊戲用戶端。在這種架構中，遊戲客戶端不會主動輪詢配對的狀態。
10. 遊戲客戶端通過WebSocket連接接收遊戲服務器的端口和 IP 地址以及玩家會話 ID。
11. 遊戲用戶端會使用後端服務提供的連接埠和 IP 位址，使用 TCP 或 UDP 連線到遊戲伺服器。遊戲用戶端也會將玩家工作階段 ID 傳送到遊戲伺服器，然後再使用 Amazon GameLift Server SDK 驗證 ID。

為亞馬遜設置指標和日誌記錄 GameLift

您可以使用從 Amazon GameLift 遊戲伺服器和資源收集的資料來協助識別異常情況。您也可以使用指標來協助改善效能。

亞馬遜要觀察的關鍵領域GameLift包括：

- Amazon GameLift 服務指標 — Amazon 在您的資源上GameLift提供 Amazon CloudWatch 指標，包括遊戲伺服器、艦隊、佇列和FlexMatch. 您可以在 Amazon GameLift 主控台和主控台中找到這些指標。CloudWatch如需中的 Amazon GameLift 指標的詳細資訊CloudWatch，請參閱監控亞馬遜 GameLift與亞馬遜 CloudWatch。
- 遊戲伺服器指標 — Amazon GameLift 無法存取您的遊戲伺服器指標。不過，您可以使用CloudWatch代理程式CloudWatch直接從遊戲伺服器傳送自訂指標到。您也可以使用叢集 AWS Identity and Access Management (IAM) 角色和 AWS SDK 將指標直接傳送至CloudWatch。如需如何設定指標的範例，請參閱開啟AWS的多人工作階段型遊戲託管。GitHub此儲存庫包含 C# StatSD 用戶端的範例CloudWatch代理程式設定和程式碼。
- 遊戲伺服器記錄 — 若要在遊戲伺服器上設定遊戲伺服器記錄檔，請使用 Amazon GameLift Server SDK 組態。您也可以使用 Amazon CloudWatch Logs 做為即時日誌管理解決方案，並且可以使用CloudWatch代理程式設定日誌。如需詳細資訊，請參閱 在亞馬遜記錄服務器消息 GameLift。

遊戲啟動清單

您可以使用這些檢查清單來驗證遊戲部署的階段。在檢查清單中，標記為 [嚴重] 的項目對於您的生產啟動至關重要。

主題

入職

使用以下檢查清單來追蹤您的遊戲上線以進行 Amazon GameLift 託管的項目。標記為 [重要] 的項目對於您的生產啟動至關重要。

- [關鍵] 在亞馬遜GameLift控制台中填寫亞馬GameLift遜入職問卷。
- [關鍵] 為遊戲客戶端設計並實施後端服務，以便與您的遊戲服務器進行交互。
- [關鍵] 建立您提供給 Amazon GameLift 伺服器執行個體以存取其他AWS資源的 AWS Identity and Access Management (IAM) 角色。
- [關鍵] 設計和實作容錯移轉至其他AWS 區域用於FlexMatch和佇列。
- 考慮到遊戲的隊列和艦隊結構，計劃將艦隊推出到您的目標位置。
- 使用基礎架構即AWS Cloud Development Kit (AWS CDK)程式碼 (IaC) AWS CloudFormation 和.
- 使用亞馬遜CloudWatch和亞馬遜簡單儲存服務 (Amazon S3) 收集日誌和分析。

測試

使用下列檢查清單追蹤測試項目，同時使用 Amazon GameLift 主機開發遊戲。標記為 [重要] 的項目對於您的生產啟動至關重要。

- [關鍵] 完成發布問卷，並將完成的問卷提交給 Amazon 上GameLift市團隊。您可以在 Amazon GameLift 控制台中找到啟動問卷。
- [嚴重] Amazon GameLift 服務配額和其他AWS 服務配額的要求增加，以便您的即時環境可以擴展到生產需求。
- [嚴重] 確認即時叢集上開啟的連接埠是否符合伺服器可使用的連接埠範圍。
- [嚴重] 關閉 RDP 連接埠 3389 和安全殼層連接埠 22。
- 制定管DevOps理遊戲的計劃。如果您使用的是 Amazon CloudWatch Logs 或 Amazon CloudWatch 自訂指標，請針對伺服器叢集上的嚴重或嚴重問題定義警示。模擬故障並測試手冊。
- 確認執行個體上執行的伺服器數目是否在伺服器執行個體類型的功能範圍內。

- 首先調整您的擴展政策，使其更加保守，並提供比您想像的更多閒置容量。您可以稍後針對成本進行最佳化。請考慮使用具有 20% 閒置容量的目標型擴展政策。
- 使用FlexMatch延遲規則來匹配地理位置接近相同AWS 區域的玩家。使用來自負載測試用戶端的綜合延遲資料，測試其在負載下的行為方式。
- 負載測試您的玩家身份驗證和遊戲會話基礎結構，以查看它是否有效擴展以滿足需求。
- 確認伺服器持續執行數天仍然可以接受連線。
- 將您的AWS Support計劃等級提高到商業或企業級，AWS以便在發生問題或中斷時回應您。

啟動

使用下列檢查清單追蹤您在 Amazon 上託管遊戲的啟動項目GameLift。標記為 [重要] 的項目對於您的生產啟動至關重要。

- [關鍵] 將艦隊保護政策設定為全面保護所有現場艦隊，以便縮小規模不會停止使用中的遊戲工作階段。
- [關鍵] 將車隊最大尺寸設定為足以滿足尖峰預期需求的最低限度。我們建議您將最大尺寸加倍以應付意想不到的需求。
- 鼓勵您的整個開發團隊參與發布活動，並在啟動室監控您的遊戲發布。
- 監控玩家延遲和玩家體驗。

推出後

使用下列檢查清單追蹤您在 Amazon GameLift 上託管的遊戲上市後項目。

- 調整擴展規則以最小化閒置容量。
- 根據您的延遲需求修改FlexMatch規則或新增其他位置。
- 優化服務器可執行文件，因為其性能效率直接影響車隊成本。若要使用相同的基礎結構執行更多遊戲工作階段，請增加每個執行個體的伺服器處理序數
- 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。

為 Amazon 準備遊戲 GameLift

若要準備好在 Amazon 上託管的多人遊戲 GameLift，請設定遊戲與 Amazon 之間的通訊 GameLift。本節中的主題提供了將您的遊戲與 Amazon GameLift、自訂遊戲伺服器和即時伺服器整合以及新增配對功能的詳細說明。 FlexMatch

主題

- 將遊戲與自訂遊戲伺服器整合
- 將遊戲與 Amazon GameLift 即時伺服器整合
- 用於 Unity 的 Amazon GameLift 插件集成遊戲
- 將遊戲與虛幻引擎的 Amazon GameLift 插件集成
- 取得 Amazon GameLift 執行個體的叢集資料
- 新增FlexMatch配對

將遊戲與自訂遊戲伺服器整合

亞馬遜GameLift提供了一套完整的工具集，用於準備要在 Amazon 上運行的多人遊戲和自定義遊戲服務器GameLift。Amazon 開GameLift發套件包含遊戲用戶端和伺服器與 Amazon GameLift 通訊所需的程式庫。如需 SDK 及其取得位置的詳細資訊，請參閱Amazon 的開發支持 GameLift。

本節中的主題包含有關如何在 Amazon 上部署之前將 Amazon GameLift 功能新增至遊戲用戶端和遊戲伺服器的詳細說明GameLift。如需在 Amazon 上啟動並執行遊戲的完整藍圖GameLift，請參閱亞馬遜 GameLift託管路線圖。

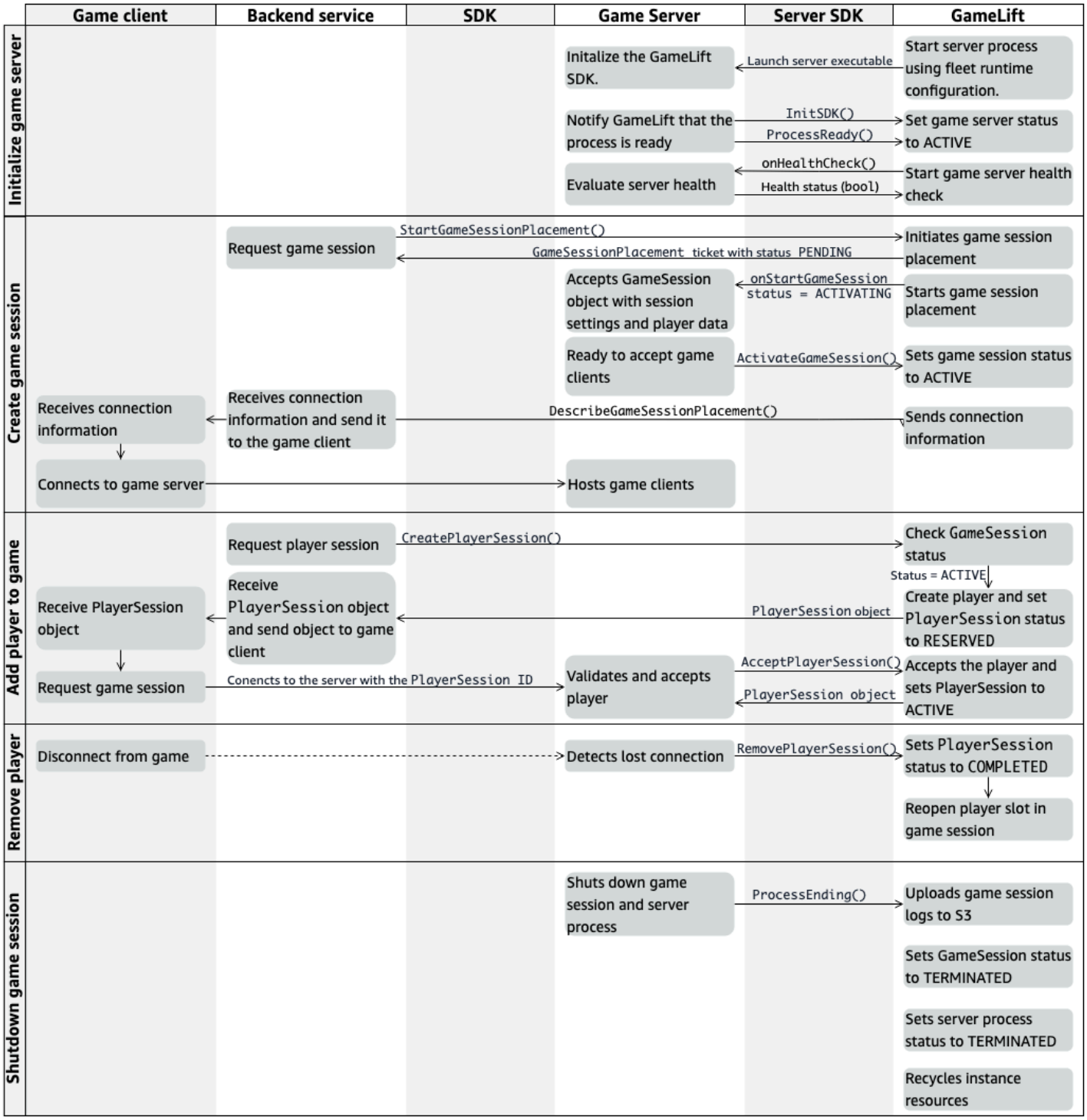
主題

- Amazon GameLift 與遊戲用戶端伺服器互動
- 整合您的遊戲伺服器與亞馬遜 GameLift
- 將您的遊戲用戶端與 Amazon 整合 GameLift
- 遊戲引擎和亞馬遜 GameLift
- 使用亞馬遜機GameLiftAnywhere隊測試您的整合
- 使用亞馬遜GameLift本地測試您的整合

Amazon GameLift 與遊戲用戶端伺服器互動

本主題說明遊戲用戶端、後端服務、遊戲伺服器和 Amazon 之間的互動GameLift。

下圖說明遊戲用戶端、後端服務、Amazon GameLift SDK、受管 EC2 遊戲伺服器、Amazon GameLift 伺服器 SDK 和 Amazon 之間的互動GameLift。如需所示互動的詳細說明，請參閱本頁的下列章節。



初始化遊戲伺服器

下列步驟說明當您準備好遊戲伺服器託管遊戲工作階段時所發生的互動情況。

1. 亞馬遜在亞馬遜彈性計算雲（Amazon EC2）實例上GameLift啟動服務器可執行文件。
2. 遊戲伺服器呼叫：

   a. `InitSDK()` 初始化伺服器開發套件。

   b. `ProcessReady()`傳達遊戲會話準備情況，連接信息和遊戲會話日誌文件的位置。

   然後服務器進程等待來自亞馬遜GameLift的回調。
3. Amazon 會GameLift更新伺服器程序的狀態，以啟`ACTIVE`用遊戲工作階段放置。
4. 亞馬遜GameLift開始調用`onHealthCheck`回調，並在服務器進程處於活動狀態時繼續定期調用它。伺服器處理序可在一分鐘內報告狀況良好或狀況不良。

建立遊戲工作階段

在您初始化遊戲伺服器之後，當您建立遊戲工作階段來接待玩家時，會發生下列互動。

1. 後端服務會呼叫 SDK 作業`StartGameSessionPlacement()`。
2. 亞馬遜GameLift創建一個帶有狀態的新`GameSessionPlacement`票證，`PENDING`並將其返回到後端服務。
3. 後端服務從佇列取得放置票證狀態。如需詳細資訊，請參閱設定遊戲工作階段位置的事件通知。
4. Amazon 透過選取適當的叢集並在具有遊戲工作階段的叢集中搜尋作用中的伺服器程序，以GameLift啟動0遊戲工作階段配置。當亞馬遜GameLift找到一個服務器進程時，亞馬遜GameLift會執行以下操作：

   a. 使用具有`GameSession`狀態的位置請求中的遊戲會話設置和玩家數據創建對`ACTIVATING`象。

   b. 叫用伺服器處理序上的`onStartGameSession`回呼。亞馬遜將信息GameLift傳遞給`GameSession`對象，表明服務器進程可能會設置遊戲會話。

   c. 將伺服器處理序的遊戲工作階段數變更為1。
5. 服務器進程運行`onStartGameSession`回調函數。當服務器進程準備好接受播放器連接時，它會調用`ActivateGameSession()`並等待播放器連接。

6. Amazon GameLift 會使用伺服器處理序的連線資訊來更新GameSession物件。(此資訊包括使用報告的連接埠設定ProcessReady()。) 亞馬遜GameLift也將狀態更改為ACTIVE。
7. 後端服務呼叫DescribeGameSessionPlacement()以偵測更新的票證狀態。接著，後端服務會使用連線資訊，將遊戲用戶端連線至伺服器程序，並加入遊戲工作階段。

將玩家新增至遊戲

此順序描述了將玩家添加到現有遊戲工作階段的過程。玩家工作階段也可以作為遊戲工作階段放置請求的一部分來申請。

1. 後端服務會CreatePlayerSession()使用遊戲工作階段 ID 呼叫用戶端 API 作業。
2. 亞馬遜GameLift檢查遊戲會話狀態（必須是ACTIVE），並在遊戲會話中查找打開的玩家插槽。如果一個插槽可用，那GameLift麼亞馬遜執行以下操作：
   a. 建立新PlayerSession物件並將狀況設定為RESERVED。
   b. 使用PlayerSession物件回應後端服務要求。
3. 後端服務使用玩家工作階段 ID 將遊戲用戶端直接連接到伺服器進程。
4. 伺服器會呼叫伺服器 API 作業AcceptPlayerSession()以驗證播放器工作階段 ID。如果通過驗證，亞馬遜將對PlayerSession象GameLift傳遞給服務器進程。伺服器程序接受或拒絕連接。
5. 亞馬遜GameLift執行以下操作之一：
   a. 如果連接被接受，則亞馬遜將狀PlayerSession態GameLift設置為ACTIVE。
   b. 如果在後端伺服器原始CreatePlayerSession()呼叫的 60 秒內未收到任何回應，Amazon會將PlayerSession狀態GameLift變更為TIMEDOUT並重新開啟遊戲工作階段中的玩家位置。

移除玩家

從遊戲工作階段中移除玩家以為新玩家建立加入空間時，會發生以下互動。

1. 玩家與遊戲中斷連線。
2. 服務器檢測到丟失的連接並調用服務器 API 操作RemovePlayerSession()。
3. 亞馬遜將狀PlayerSession態GameLift更改為COMPLETED並重新打開遊戲會話中的玩家位置。

關閉遊戲工作階段

當伺服器處理序關閉目前的遊戲工作階段時，就會發生這種互動順序。

1. 伺服器會關閉遊戲工作階段和伺服器。
2. 服務器調`ProcessEnding()`用亞馬遜GameLift。
3. 亞馬遜GameLift執行以下操作：
    a. 將遊戲工作階段日誌上傳到亞馬遜簡單儲存服務 (Amazon S3)。
    b. 將`GameSession`狀態變更為`TERMINATED`。
    c. 將伺服器處理序狀態變更為`TERMINATED`。
    d. 回收實例資源。

整合您的遊戲伺服器與亞馬遜 GameLift

在 Amazon 執行個體上部署並GameLift執行自訂遊戲伺服器之後，它必須能夠與 Amazon GameLift (以及可能的其他資源) 互動。本節說明如何將您的遊戲伺服器軟體與 Amazon 整合GameLift。

> **Note**
> 這些指示假設您已建立，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工作：

- 在 Amazon GameLift 和您的遊戲伺服器之間建立通訊。
- 產生並使用 TLS 憑證，在遊戲用戶端與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資源互動的權限。
- 允許遊戲伺服器程序取得其執行所在叢集的相關資訊。

主題

- 添加亞馬遜GameLift到您的遊戲服務器
- 與您車隊的其他AWS資源進行溝通

添加亞馬遜GameLift到您的遊戲服務器

您的自訂遊戲伺服器必須與 Amazon 通訊GameLift，因為每個遊戲伺服器程序都必須能夠回應 Amazon GameLift 啟動的事件。您的遊戲伺服器還必須讓 Amazon GameLift 知道伺服器處理狀態和玩家連線。如需有關遊戲伺服器、後端服務、遊戲用戶端和 Amazon GameLift 如何共同管理遊戲託管的詳細資訊，請參閱Amazon GameLift 與遊戲用戶端伺服器互動。

若要準備您的遊戲伺服器與 Amazon 互動GameLift，請將 Amazon GameLift Server SDK 新增至您的遊戲伺服器專案，並建置本主題所述的功能。伺服器 SDK 提供多種語言版本。如需 Amazon GameLift 伺服器開發套件的詳細資訊，請參閱Amazon 的開發支持 GameLift。

伺服器 SDK API 參考資料：

- 亞馬遜GameLift服務器 SDK 5.x 參考 C ++
- 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考
- 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考

初始化伺服器處理序

新增程式碼以建立與 Amazon 的通訊，GameLift並報告伺服器處理序已準備好託管遊戲工作階段。此代碼必須在任何 Amazon GameLift 代碼之前運行。

1. 通過調用初始化亞馬遜 GameLift API 客戶端`InitSdk()`。若要初始化 Amazon GameLift Anywhere 運算資源上的伺服器程序，請`InitSdk()`使用下列命令呼叫`ServerParameters`：
   - 用來連接到您的遊戲伺服器的網路通訊端網址。
   - 用來託管遊戲伺服器的程序 ID。
   - 主控遊戲伺服器處理程序的運算 ID。
   - 包含您 Amazon GameLift Anywhere 運算的GameLift叢集識別碼。
   - 亞馬遜GameLift操作生成的授權令牌`GetComputeAuthToken`。

> **Note**
>
> 要在亞馬遜GameLift託管的 Amazon EC2 實例上初始化遊戲服務器，請ServerParameters使用默認 `InitSDK()`（C ++）（C＃）（不真實）（C ++）（C＃）例。亞馬遜GameLift設置了運算環境，並自動GameLift為您連接到亞馬遜。

2. 通GameLift知 Amazon 伺服器處理序已準備好主持遊戲工作階段。使用以下信息調ProcessReady()用（C ++）（C#）（C ++）（虛幻。（請注意，每個服務器進程ProcessReady()只能調用一次）。

   - 伺服器處理序使用的連接埠號碼。後端服務會為遊戲用戶端提供連接埠號碼和 IP 位址，以連線至伺服器程序並加入遊戲工作階段。
   - 您希望 Amazon GameLift 保留的檔案位置，例如遊戲工作階段日誌。服務器進程在遊戲會話期間生成這些文件。它們會暫時儲存在執行伺服器處理序的執行個體上，而且在執行個體關閉時就會遺失。您列出的任何文件都會上傳到亞馬遜GameLift。您可以透過 Amazon GameLift 主控台或呼叫 Amazon GameLift API 操作 GetGameSessionLogUrl() 存取這些檔案。
   - 亞馬遜GameLift可以調用到您的服務器進程的回調函數的名稱。您的遊戲伺服器必須實作這些功能。有關更多信息，請參閱（C ++）（C#）（虛幻））。
     - (選用) onHealthCheck — Amazon 會定期GameLift呼叫此函數，從伺服器要求健康狀態報告。
     - onStartGameSession-亞馬遜GameLift調用此函數以響應客戶請求CreateGameSession（）。
     - onProcessTerminate-亞馬遜GameLift強制服務器進程停止，使其正常關閉。
     - (選用) onUpdateGameSession — Amazon 會將更新的遊戲工作階段物件GameLift傳送至遊戲伺服器，或提供比賽回填請求的狀態更新。FlexMatch回填功能需要此回調。

   您也可以設定遊戲伺服器，以安全地存取您擁有或控制的AWS資源。如需詳細資訊，請參閱與您車隊的其他AWS資源進行溝通。

(選擇性) 報表伺服器處理序健康

將程式碼新增至您的遊戲伺服器以實作回呼函式onHealthCheck()。Amazon 會定期呼GameLift叫此回呼方法來收集運作狀態指標。若要實作此回呼函式，請執行下列動作：

- 評估伺服器處理序的健全狀況狀態。例如，如果有任何外部相依性失敗，您可能會將伺服器處理序報告為健康狀況不佳。
- 完成運作狀態評估和在 60 秒內回應回呼。如果 Amazon 在這段時間內GameLift沒有收到回應，它會自動將伺服器處理序視為健康狀況不佳。
- 返回一個布爾值：true 表示健康，假表示不健康。

如果您沒有實作運作狀態檢查回呼，則 Amazon GameLift 會將伺服器處理序視為健康狀態，除非伺服器沒有回應。

Amazon GameLift 使用伺服器處理序健康狀態來結束健康狀態不良的程序並清除資源。如果伺服器處理序持續報告狀況不良，或連續三次運作狀態檢查沒有回應，Amazon GameLift 可能會關閉該程序並開始新的程序。Amazon GameLift 會在叢集的伺服器程序運作狀態上收集指標。

(選擇性) 取得 TLS 憑證

如果伺服器處理序是在已啟動 TLS 憑證產生的叢集上執行，則您可以擷取 TLS 憑證以與遊戲用戶端建立安全連線，並加密用戶端伺服器的通訊。憑證的副本會存放在執行個體上。要獲取文件位置，請調用 `GetComputeCertificate()`（C ++）（C＃）（虛幻））。

開始遊戲工作階段

新增程式碼以實作回呼函式 `onStartGameSession`。亞馬遜GameLift調用此回調以在服務器上啟動遊戲會話。

該`onStartGameSession`函數需要一個GameSession對象作為輸入參數。此物件包含重要的遊戲工作階段資訊，例如玩家數目上限。它還可以包括遊戲數據和玩家數據。函數實現應該執行以下任務：

- 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，遊戲用戶端在連線到伺服器程序時所參考的 ID。
- 根據需要處理遊戲數據和玩家數據。此資料位於`GameSession`物件中。
- GameLift當新的遊戲工作階段準備好接受玩家時，請通知 Amazon。調用服務器 API 操作 `ActivateGameSession()`（C ++）（C＃）（虛幻））。為了回應成功通話，Amazon 會將遊戲工作階段狀態GameLift變更為`ACTIVE`。

(選擇性) 驗證新玩家

如果您正在追蹤玩家工作階段的狀態，請新增程式碼以在新玩家連線至遊戲伺服器時驗證他們。亞馬遜GameLift跟踪當前玩家和可用的遊戲會話插槽。

為了進行驗證，要求存取遊戲工作階段的遊戲用戶端必須包含玩家工作階段 ID。當玩家要求使用 StartGameSessionPlacement() 或 StartMatchmaking () 加入遊戲時，Amazon 會GameLift自動生成此 ID。然後，玩家會話在遊戲會話中保留一個開放的老虎機。

當遊戲伺服器處理序收到遊戲用戶端連線要求時，會使用玩家工作階段 ID 呼叫 `AcceptPlayerSession()`(C++) (虛幻)。作為回應，Amazon 會GameLift驗證玩家工作階段 ID 對應

於遊戲工作階段中保留的開放位置。亞馬遜GameLift驗證播放器會話 ID 後，服務器進程接受連接。然後，玩家可以加入遊戲會話。如果 Amazon GameLift 未驗證玩家工作階段 ID，則伺服器程序會拒絕連線。

(選擇性) 回報玩家工作階段結束

如果您正在跟踪玩家會話的狀態，請添加代碼以在玩家離開遊戲會話GameLift時通知亞馬遜。伺服器程序偵測捨棄連線時應執行此程式碼。亞馬遜GameLift使用此通知來跟踪當前玩家和遊戲會話中的可用位置。

要處理丟棄的連接，請在代碼中添加調用到服務器 API 操作 `RemovePlayerSession()`（C ++）（C＃）（虛幻）（C ++）與相應的播放器會話 ID。

結束遊戲工作階段

將代碼添加到服務器進程關閉順序中，以在遊戲會話結束GameLift時通知 Amazon。若要回收和重新整理託管資源，Amazon 會在遊戲工作階段完成後GameLift關閉伺服器程序。

在服務器進程關閉代碼的開始，調用服務器 API 操作 `ProcessEnding()`（C ++）（C＃）（虛幻））。此調用通GameLift知亞馬遜服務器進程正在關閉。亞馬遜GameLift將遊戲會話狀態和服務器進程狀態更改為`TERMINATED`。通話後`ProcessEnding()`，該過程可以安全地關閉。

回應伺服器處理序關閉通知

添加代碼以關閉服務器進程以響應來自亞馬遜的通知GameLift。Amazon GameLift 會在伺服器處理序持續報告狀況不佳，或伺服器處理序執行的執行個體終止時傳送此通知。Amazon GameLift 可以在容量縮減事件中停止執行個體，或是回應競價型執行個體中斷。

若要處理關機通知，請對您的遊戲伺服器程式碼進行下列變更：

- 實作回呼函式 `onProcessTerminate()`。這個函數應該調用關閉服務器進程的代碼。Amazon GameLift 呼叫此操作時，Spot 執行個體中斷會提供兩分鐘通知。此通知提供伺服器處理時間，讓玩家正常中斷連線、保留遊戲狀態資料，以及執行其他清理工作。
- 從遊戲伺服器關閉程式碼呼叫伺服器 API 作業 `GetTerminationTime()`(C++) (C#) (虛 (C++) (C#)。如果 Amazon GameLift 已發出停止伺服器處理序的呼叫，則會`GetTerminationTime()`傳回估計的終止時間。
- 在遊戲服務器關閉代碼的開始時，調用服務器 API 操作 `ProcessEnding()`（C ++）（C＃）（虛幻））。此調用通GameLift知亞馬遜服務器進程正在關閉，亞馬遜GameLift然後將服務器進程狀態更改為`TERMINATED`。通話後`ProcessEnding()`，該過程可以安全地關閉。

與您車隊的其他AWS資源進行溝通

當您建立要在 Amazon GameLift 叢集上部署的遊戲伺服器組建時，您可能希望遊戲組建中的應用程式與您擁有的其他AWS資源直接且安全地通訊。由於 Amazon 會 GameLift 管理您的遊戲託管叢集，因此您必須授予 Amazon 對這些資源和服務的 GameLift有限存取權限。

一些範例案例包括：

- 使用 Amazon CloudWatch 代理程式從受管 EC2 叢集和叢集收集指標、日誌和Anywhere追蹤
- 將執行個體日誌資料傳送到 Amazon CloudWatch 日誌。
- 取得存放在亞馬遜簡易儲存服務 (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中的遊戲資料 (例如遊戲模式或庫存)。
- 使用 Amazon Simple Queue Service (Amazon SQS) 將信號直接傳送到執行個體。
- 存取在 Amazon 彈性運算雲端 (Amazon EC2) 上部署和執行的自訂資源。

Amazon GameLift 支援下列建立存取權的方法：

- 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，並設定該存取權限制。受信任的方可以「假定」一個角色，並取得授權他們與資源互動的臨時安全登入資料。當雙方提出與資源相關的 API 請求時，必須包含憑據。

若要設定 IAM 角色控制的存取權，請執行下列工作：

1. 建立 IAM 角色
2. 修改應用程式以取得認證
3. 將叢集與 IAM 角色建立關聯

建立 IAM 角色

在此步驟中，您會建立 IAM 角色，其中包含一組許可以控制對AWS資源的存取，以及授予 Amazon 使用該角色許可 GameLift 權的信任政策。

如需如何設定 IAM 角色的指示，請參閱為 Amazon 設置 IAM 服務角色 GameLift。建立權限原則時，請選擇應用程式需要使用的特定服務、資源和動作。最佳作法是盡可能限制權限的範圍。

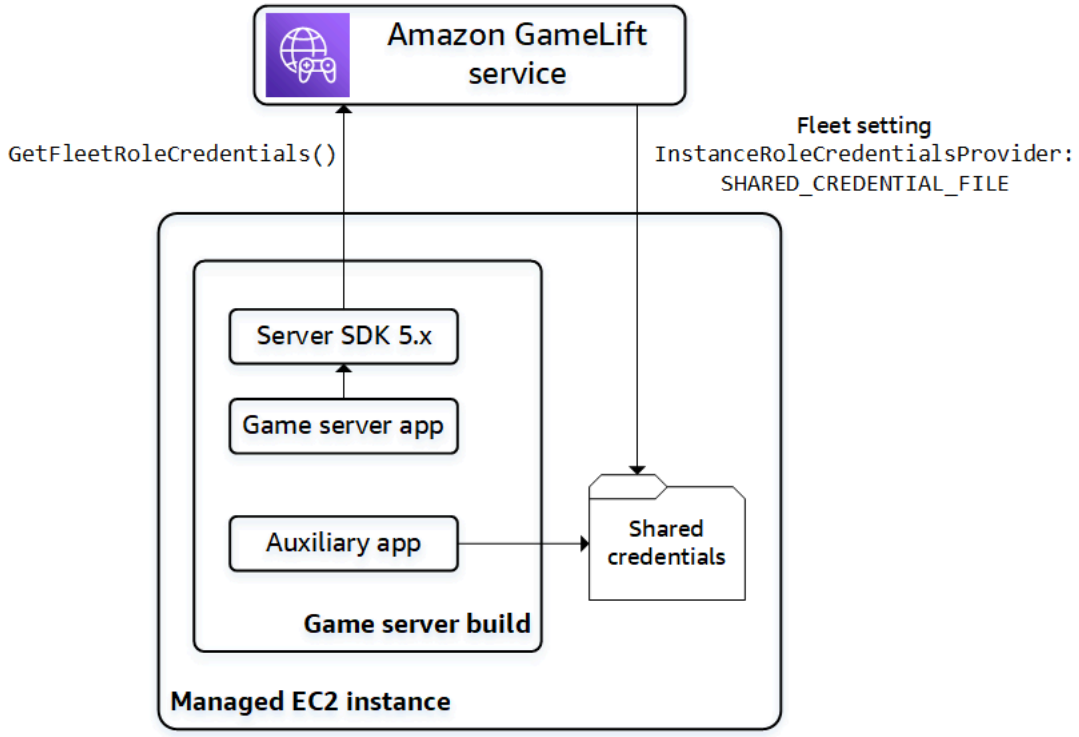
建立角色後，請記下角色的 Amazon 資源名稱 (ARN)。在建立叢集期間，您需要角色 ARN。

修改應用程式以取得認證

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，並在與資源互動時使用這些登入 AWS資料。請參閱下表，判斷如何根據 (1) 應用程式類型，以及 (2) 您的遊戲用來與 Amazon 通訊的伺服器 SDK 版本來修改應用程式 GameLift。

	遊戲伺服器應用	其他應用
使用伺服器 SDK 第 5.x 版	`GetFleetRoleCredentials()` 從您的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，以便從叢集執行個體上的共用檔案提取認證。
使用伺服器 SDK 版本 4 或更早版本	`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 Server SDK 5.x 整合的遊戲，此圖表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)

在您的遊戲服務器代碼中，該代碼應該已經與亞馬遜 GameLift服務器 SDK 5.x 集成，調用`GetFleetRoleCredentials`（C ++）（C＃）（虛幻）以檢索一組臨時憑據。當認證過期時，您可以透過另一次呼叫來重新整理它們`GetFleetRoleCredentials`。

使用共享憑據（服務器 SDK 5.x）

對於使用服務器 SDK 5.x 的遊戲服務器構建部署的非服務器應用程序，請添加代碼以獲取和使用存儲在共享文件中的憑據。Amazon GameLift 會為每個叢集執行個體產生登入資料設定檔。這些認證可供執行個體上的所有應用程式使用。亞馬遜 GameLift不斷刷新臨時登入資料。

您必須設定叢集，才能在建立叢集時產生共用認證檔案。

在需要使用共用認證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 叢集執行個體的指標

如果您想要使用 Amazon CloudWatch 代理程式從 Amazon GameLift 叢集收集指標、日誌和追蹤，請使用此方法授權代理程式將資料發送到您的帳戶。在這個案例中，請執行下列步驟：

1. 擷取或寫入 CloudWatch 代理程式`config.json`檔案。
2. 如上所述，更新代理程式的`common-config.toml`檔案以識別身份證明檔案名稱和設定檔名稱。
3. 設定您的遊戲伺服器組建安裝指令碼以安裝並啟動 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，並取得登入資料以與您的AWS資源互動。在具有伺服器 SDK 4 或更早版本的 Amazon GameLift 叢集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 作業並指定角色 ARN。此作業會傳回一組暫時認證，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。

將叢集與 IAM 角色建立關聯

建立 IAM 角色並更新遊戲伺服器組建中的應用程式以取得和使用存取登入資料之後，您就可以部署叢集。當您設定新叢集時，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 Amazon GameLift 為每個叢集執行個體產生共用登入資料檔案，請將此參數設定為`SHARED_CREDENTIAL_FILE`。

您必須在建立叢集時設定這些值。它們無法稍後更新。

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 執行個體上 GameLift 執行的應用程式與其他AWS資源之間進行通訊。VPC 是您定義的虛擬私人網路，其中包含一

組透過您AWS 帳戶管理的資源。每個亞馬遜 GameLift車隊都有自己的 VPC。透過 VPC 對等互連，您可以在叢集的 VPC 與其他資源之間建立直接的網路連線。AWS

Amazon GameLift 簡化了為您的遊戲伺服器設定 VPC 對等連線的程序。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 對等互連的指示，請參閱。適用於亞馬遜的 VPC 對等互連 GameLift

將您的遊戲用戶端與 Amazon 整合 GameLift

本節中的主題說明您可以新增至後端服務的受管 Amazon GameLift 功能。後端服務會處理下列工作：

- 向 Amazon 索取有關使用中遊戲工作階段的資訊GameLift。
- 將玩家加入現有的遊戲工作階段。
- 創建一個新的遊戲會話，並加入玩家。
- 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 Amazon 以GameLift及在 Amazon 上執行的遊戲伺服器互動的詳細資訊 GameLift，請參閱Amazon GameLift 與遊戲用戶端伺服器互動。

先決條件

- AWS 帳戶。
- 上傳到亞馬遜的遊戲服務器構建GameLift。
- 一個用於託管遊戲的艦隊。

主題

- 添加 Amazon GameLift 到您的遊戲客戶端
- 產生玩家 ID

添加 Amazon GameLift 到您的遊戲客戶端

GameLift 將 Amazon 整合到需要遊戲工作階段資訊的遊戲元件中、建立新遊戲工作階段，以及將玩家新增至遊戲。視您的遊戲架構而定，此功能位於後端服務中，可處理玩家驗證、配對或遊戲工作階段放置等工作。

> **Note**
>
> 如需有關如何為您的 Amazon GameLift 託管遊戲設定配對的詳細資訊，請參閱 Amazon GameLift FlexMatch 開發人員指南。

在後端服務 GameLift 上設置 Amazon

新增程式碼以初始化 Amazon GameLift 用戶端和存放金鑰設定。此代碼必須在依賴 Amazon 的任何代碼之前運行 GameLift。

1. 設定用戶端組態。使用預設的用戶端設定或建立自訂用戶端配置物件。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 Amazon 時要使用的目標區域和端點 GameLift。區域可識別要使用的已部署資源集 (叢集、佇列和配對系統)。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 客戶端。使用 A ws:GameLift:: GameLiftClient () (C++) 或 AmazonGameLiftClient(C#) 搭配預設用戶端組態或自訂用戶端組態。
3. 添加一個機制來為每個玩家生成一個唯一的標識符。如需詳細資訊，請參閱 產生玩家 ID。
4. 收集並儲存下列資訊：

   - 目標叢集 — 許多 Amazon GameLift API 請求必須指定叢集。若要這麼做，請使用指向目標叢集的叢集 ID 或別名 ID。最佳做法是使用艦隊別名，以便您可以將玩家從一個艦隊切換到另一個艦隊，而無需更新後端服務。
   - 目標佇列 — 對於使用多叢集佇列來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 — 所有對 Amazon 的呼叫都 GameLift 必須提供託管遊戲 AWS 帳戶 的登入資料。您可以透過建立播放程式使用者來取得這些認證，如中所述為您的遊戲設定程式化存取。根據您管理播放程式使用者存取權的方式，執行下列動作：
     - 如果您使用角色來管理播放器使用者許可，請在呼叫 Amazon GameLift API 之前新增程式碼以擔任該角色。假設角色的要求會傳回一組暫時的安全性登入資料。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)。
     - 如果您有長期的安全登入資料，請設定您的程式碼以尋找並使用預存的認證。請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。有關存儲憑據的信息，請參閱（C++）和（.NET）的AWS API 引用。

- 如果您有臨時安全登入資料，請使用 AWS Security Token Service (AWS STS) 新增程式碼以定期重新整理登入資料，如 IAM 使用者指南中的 AWS SDK 使用臨時安全登入資料中所述。代碼必須在舊憑據過期之前請求新憑據。

取得遊戲階段

新增程式碼以探索可用的遊戲工作階段，以及管理遊戲工作階段設定和

搜尋目前的遊戲工作階段

用於取SearchGameSessions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準則的工作階段的相關資訊。此呼叫會針GameSession對符合您的搜尋要求的每個作用中遊戲工作階段傳回一個物件。

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() — 除了遊戲工作階段資訊外，還可取得遊戲工作階段的防護狀態。
- UpdateGameSession() — 視需要變更遊戲工作階段的中繼資料和設定。
- GetGameSessionLogUrl— 訪問存儲的遊戲會話日誌.

建立遊戲階段

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個選項，這取決於您是在多個區域 AWS 區域 還是在單一區域中部署遊戲。

在多位置佇列中建立遊戲工作階段

用StartGameSessionPlacement於在佇列中提出新遊戲工作階段的要求。若要使用此作業，請建立佇列。這決定了 Amazon 放 GameLift 置新遊戲會話的位置。如需佇列及其使用方式的詳細資訊，請參閱為遊戲會話放置設置亞馬遜GameLift隊列。

建立遊戲工作階段位置時，請指定要使用的佇列名稱、遊戲工作階段名稱、同時玩家數目上限，以及一組選擇性的遊戲屬性。您也可以選擇性地提供自動加入遊戲階段的玩家清單。如果您包含相關區域的玩家延遲資料，Amazon 會 GameLift 使用此資訊將新的遊戲工作階段置於叢集中，為玩家提供理想的遊戲體驗。

遊戲工作階段放置為非同步程序。提出請求後，您可以讓它成功或逾時。您也可以隨時使用取消請求StopGameSessionPlacement。若要查看放置要求的狀態，請致電DescribeGameSessionPlacement。

在特定艦隊上建立遊戲工作階段

用CreateGameSession於在指定的叢集上建立新的工作階段。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。Amazon GameLift 創建新的遊戲會話並返回一個GameSession對象後，您可以加入玩家。

當您使用此操作時，請提供艦隊 ID 或別名 ID、工作階段名稱，以及遊戲的同時玩家數目上限。您可以選擇包括一組遊戲屬性。遊戲屬性在鍵值對的陣列中定義。

如果您使用 Amazon GameLift 資源保護功能來限制一位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。

加入玩家參加遊戲階段

新增程式碼以保留使用中遊戲工作階段中的玩家位置，並將遊戲用戶端連線至遊戲工作階段。

1. 在遊戲工作階段中保留一個玩家角位

   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。

   建立新玩家工作階段的方式有兩種：

   - 用StartGameSessionPlacement於為新遊戲階段中的一個或多個玩家保留欄位。
   - 使用或CreatePlayerSessions具有遊戲工作階段 ID 的玩家預留位置給一個CreatePlayerSession或多個玩家。

Amazon GameLift 首先驗證遊戲工作階段是否接受新玩家，並且有可用的玩家位置。如果成功，Amazon 為玩家 GameLift 保留一個插槽，創建新的玩家會話，並返回一個PlayerSession對象。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 位址和連接埠。

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 ID。

玩家工作階段可以包含一組自訂玩家資料。此數據存儲在新創建的播放器會話對象中，您可以通過調用 DescribePlayerSessions（ ）來檢索該對象。當玩家直接連接到遊戲會話時，Amazon GameLift 也將此對象傳遞給遊戲服務器。請求多個玩家工作階段時，請為對應至要求中的玩家 ID 的每個玩家提供一串玩家資料。

2. Connect 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。使用此資訊可建立與伺服器的直接連線。

   - 您可以使用指定的連接埠以及指派給伺服器處理序的 DNS 名稱或 IP 位址進行連線。
   - 如果您的叢集已啟用 TLS 憑證產生功能，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的玩家連線，請參考玩家工作階段 ID。

   進行連接後，遊戲客戶端和服務器進程直接進行通信，而不涉及 Amazon GameLift。服務器與 Amazon 保持通信， GameLift以報告玩家連接狀態，健康狀態等。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保留位置，並接受或拒絕玩家連線。當播放程式中斷連線時，伺服器處理序會報告中斷的連線。

使用遊戲工作階段屬

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。遊戲屬性是您的遊戲伺服器可以新增、讀取、列出和變更的鍵值配對。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啟用時傳入遊戲屬性。一個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：`NoviceEasyIntermediate`、、和`Expert`。玩家選擇`Easy`，然後開始遊戲。您的遊戲客戶端使用`StartGameSessionPlacement`或按照前面章節中 GameLift 所述的方式向 Amazon 請求新的遊戲工`CreateGameSession`作階段。在請求中，客戶端通過：`{"Key": "Difficulty", "Value":"Easy"}`。

為了響應請求，Amazon GameLift 創建了一個包含指定遊戲屬性的GameSession對象。Amazon GameLift 然後指示可用的遊戲服務器啟動新的遊戲會話並傳遞對GameSession象。遊戲伺服器會使用Difficulty的開始遊戲工作階段Easy。

進一步了解

- GameProperty 資料類型
- SearchGameSessions() 範例
- UpdateGameSession() GameProperties 參數

產生玩家 ID

Amazon GameLift 使用玩家工作階段來代表連線到遊戲工作階段的玩家。每當玩家使用與 Amazon 整合的遊戲用戶端連線到遊戲工作階段時，Amazon 都會GameLift建立一個玩家工作階段GameLift。當玩家離開遊戲時，玩家工作階段就會結束。亞馬遜GameLift不會重複使用玩家會話。

下列程式碼範例會隨機產生唯一的玩家 ID：

```
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，請參閱查看有關遊戲和玩家會話的數據。

遊戲引擎和亞馬遜 GameLift

您可以將受管亞馬遜GameLift服務與支援 C++ 或 C# 程式庫 (包括 O3DE、虛幻引擎和 Unity) 的大多數主要遊戲引擎搭配使用。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。如需有關可用 Amazon GameLift SDK、支援的開發平台和作業系統的詳細資訊，請參閱遊戲伺服器Amazon 的開發支持 GameLift的相關資訊。

除了本主題中提供的引擎特定資訊之外，您還可以在下列主題中尋找GameLift將 Amazon 整合到遊戲伺服器、用戶端和服務的其他說明：

- 亞馬遜GameLift託管路線圖— 成功GameLift將 Amazon 整合到您的遊戲中並設定託管資源的六個步驟工作流程。
- 添加亞馬遜GameLift到您的遊戲服務器— 有關GameLift將亞馬遜集成到遊戲服務器中的詳細說明。

- 添加 Amazon GameLift 到您的遊戲客戶端 – 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

GameLift使用適用於 C++ 的亞馬遜伺服器開發套件，準備您的遊戲伺GameLift服器以便在亞馬遜 請參閱 添加亞馬遜GameLift到您的遊戲服務器，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

使您的遊戲用戶端和/或遊戲服務能夠與 Amazon GameLift 服務互動，例如尋找可用的遊戲工作階段或建立新的遊戲工作階段，以及將玩家新增至遊戲。核心用戶端功能在適用於 C++ 的 AWS SDK 中提供。要GameLift將亞馬遜集成到您的 O3DE 遊戲項目中，請參閱將亞馬遜添加GameLift到 O3DE 遊戲客戶端和服務器和。添加 Amazon GameLift 到您的遊戲客戶端

虛擬引擎

遊戲伺服器

將虛幻引擎的 Amazon GameLift 伺服器開發套件新增至您的專案，並實作所需的伺GameLift服器功能，準備好要在 Amazon 上託管的遊戲伺服器。如需設定虛幻引擎外掛程式和新增 Amazon 程式GameLift碼的說明，請參閱 GameLift 將 Amazon 集成到虛幻引擎項目中。

遊戲用戶端與服務

使您的遊戲用戶端和/或遊戲服務能夠與 Amazon GameLift 服務互動，例如尋找可用的遊戲工作階段或建立新的遊戲工作階段，以及將玩家新增至遊戲。核心用戶端功能在適用於 C++ 的 AWS SDK 中提供。要GameLift將亞馬遜集成到您的虛幻引擎遊戲項目中，請參閱添加 Amazon GameLift 到您的遊戲客戶端。

Unity

遊戲伺服器

將適用於 C# 的 Amazon GameLift Server SDK 新增至您GameLift的專案，並實作必要的伺服器功能，準備好要在 Amazon 上託管的遊戲伺服器。如需使用 Unity 設定和新增 Amazon GameLift 程式碼的說明，請參閱 GameLift 將亞馬遜整合到統一項目中。

遊戲用戶端與服務

使您的遊戲用戶端和/或遊戲服務能夠與 Amazon GameLift 服務互動，例如尋找可用的遊戲工作階段或建立新的遊戲工作階段，以及將玩家新增至遊戲。核心用戶端功能在 AWS SDK for .NET 中提供。要GameLift將亞馬遜集成到您的統一遊戲項目中，請參閱添加 Amazon GameLift 到您的遊戲客戶端。

其他引擎

如需遊戲伺服器和用戶端可用的 Amazon GameLift 開發套件完整清單，請參閱the section called “開發支援”。

將亞馬遜添加GameLift到 O3DE 遊戲客戶端和服務器

您可以使用 O3DE，這是一種開放原始碼、跨平台的即時 3D 引擎，建立高效能的互動體驗，包括遊戲和模擬。O3DE 渲染器和工具包裝在模塊化框架中，您可以使用首選的開發工具進行修改和擴展。

模塊化框架使用包含具有標準接口和資產的庫的寶石。選擇您自己的寶石，根據您的需求選擇要添加的功能。

亞馬遜GameLift寶石提供以下功能：

亞馬遜GameLift整合

一個擴展 O3DE 網絡層並讓多人 Gem 與亞馬遜GameLift專用服務器解決方案一起工作的框架。寶石提供了與亞馬遜服務GameLift器 SDK 和 AWS SDK 客戶端的集成（調用亞馬遜GameLift服務本身）。

構建和軟件包管理

封裝和選擇性上傳專用伺服器組建和 AWS Cloud Development Kit (AWS CDK) (AWS CDK) 應用程式以設定和更新資源的指示。

亞馬遜GameLift寶石設置

請按照本節中的步驟在 O3DE 中設置亞馬遜GameLift寶石。

先決條件

- 為亞馬遜設置您的AWS帳戶GameLift。如需詳細資訊，請參閱設置一個 AWS 帳戶。
- 設定 O3 AWS DE 的認證。如需詳細資訊，請參閱設定AWS認證。
- 設定AWS CLI和AWS CDK。如需詳細資訊，AWS Command Line Interface和 AWS Cloud Development Kit (AWS CDK).

打開亞馬遜GameLift寶石及其依賴關係

1. 開啟「專案管理員」。
2. 打開項目下的菜單，然後選擇編輯項目設置... 。
3. 選擇配置寶石。
4. 打開亞馬遜GameLift寶石和以下依賴的寶石：
   - AWS核心寶石-提供AWS 服務在 O3DE 中使用的框架。
   - 多人寶石 — 通過擴展網絡框架提供多人遊戲功能。

包括亞馬遜GameLift寶石靜態庫

1. 包含專案伺BUILD_DEPENDENCIES服器目標的Gem::AWSGameLift.Server.Static作為。

```
ly_add_target(
    NAME YourProject.Server.Static STATIC
    ...
    BUILD DEPENDCIES
        PUBLIC
            ...
        PRIVATE
            ...
            Gem::AWSGameLift.Server.Static
)
```

2. AWSGameLiftService將專案伺服器系統元件設定為必要項目。

```
void
 YourProjectServerSystemComponent::GetRequiredServices(AZ::ComponentDescriptor::DependencyA
 required)
{
    ...
    required.push_back(AZ_CRC_CE("AWSGameLiftServerService"));
    ...
}
```

3. (選擇性) 若要以 C++ BUILD_DEPENDENCIES 提出 Amazon GameLift 服務請求，請Gem::AWSGameLift.Client.Static在用戶端目標的中包含。

```
ly_add_target(
```

```
    NAME YourProject.Client.Static STATIC
    ...
    BUILD_DEPENDENCIES
    PUBLIC
        ...
    PRIVATE
        ...
        Gem::AWSCore.Static
        Gem::AWSGameLift.Client.Static
}
```

整合您的遊戲與專屬伺服器

透過工作階段管理整合管理遊戲和專用遊戲伺服器中的遊戲工作階段。若要支援FlexMatch，請參閱FlexMatch整合。

GameLift 將 Amazon 集成到虛幻引擎項目中

本主題說明如何為虛幻引擎設定 Amazon GameLift C++ 伺服器 SDK 外掛程式，並將其整合到您的遊戲專案中。

其他資源：

- 虛幻下載網站的服務器 SDK 插件
- 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考
- the section called “開發支援”

必要條件

在處理之前，請確定您已檢閱下列先決條件：

必要條件

- 能夠運行虛幻引擎的計算機。如需虛幻引擎需求的詳細資訊，請參閱虛幻引擎的硬體與軟體規格說明文件。
- Microsoft 視覺工作室 2019 或更新版本。
- C 製作 3.1 版或更高版本。
- Python 版本 3.6 或更高版本。

- 在路徑上可用的 Git 客戶端。
- 一個史詩遊戲帳號。在虛幻引擎官方網站上註冊一個帳戶。
- 與虛幻引擎 GitHub 帳戶相關聯的帳戶。如需詳細資訊，請參閱虛幻引擎網站 GitHub上的存取虛幻引擎原始程式碼。

> **Note**
>
> Amazon GameLift 目前支持以下版本的虛幻引擎：
>
> - 4.22
> - 4.23
> - 4.24
> - 4.25
> - 4.26
> - 4.27
> - 5.1.0
> - 5.1.1
> - 5.2
> - 5.3

從源代碼構建虛幻引擎

通過 Epic 啟動器下載的虛幻引擎編輯器的標準版本僅允許虛幻客戶端應用程序構建。為了構建虛幻服務器應用程序，您需要使用虛幻引擎 Github 存儲庫從源下載和構建虛幻引擎。如需詳細資訊，請參閱虛幻引擎文件網站上的從原始碼建構虛幻引擎教學課程。

> **Note**
>
> 如果您尚未這麼做，請依照上的存取虛幻引擎原始碼中的指示，將您的 GitHub 帳號連結 GitHub至您的 Epic Games 帳戶。

將虛幻引擎來源複製到您的開發環境

1. 將虛幻引擎原始碼複製到您選擇的分支中的開發環境。

```
git clone https://github.com/EpicGames/UnrealEngine.git
```

2. 查看您用來開發遊戲的版本標籤。例如，下列範例會檢出虛幻引擎版本 5.1.1：

```
git checkout tags/5.1.1-release -b 5.1.1-release
```

3. 導覽至本機存放庫的根資料夾。當您位於根資料夾時，請執行下列檔案：Setup.bat。
4. 在根文件夾中，同時運行文件：GenerateProjectFiles.bat。
5. 執行上述步驟中的檔案之後，會建立虛幻引擎解決方案檔案。UE5.sln打開視覺工作室，並在視覺工作室編輯器中打開該UE5.sln文件。
6. 在 Visual Studio 中，開啟 [檢視] 功能表，然後選擇 [方案總管] 選項。這會開啟虛幻專案節點的右鍵功能表。在 [方案總管] 視窗中，以滑鼠右鍵按一下UE5.sln檔案 (可以只列出UE5)，然後選擇 [建置] 以使用開發編輯器 Win64 目標建置虛幻專案。

> **Note**
>
> 組建可能需要一個多小時才能完成。

構建完成後，您就可以打開虛幻開發編輯器並創建或導入項目。

為外掛程式設定虛幻專案

請按照以下步驟操作，為您的遊戲 GameLift 服務器項目準備好虛幻引擎的 Amazon 服務器 SDK 插件。

若要設定外掛程式的專案

1. 開啟 Visual Studio 時，瀏覽至 [方案總管] 窗格，然後選擇要開啟虛幻專UE5案的內容功能表的檔案。在內容功能表中，選擇「設定為啟動專案」選項。
2. 在您的 Visual Studio 視窗的頂端，選擇開始偵錯 (綠色箭頭)。

   此動作會啟動虛幻編輯器的新原始碼建置執行個體。如需使用虛幻編輯器的詳細資訊，請參閱虛幻引擎文件網站上的虛幻編輯器介面。
3. 關閉您開啟的 Visual Studio 視窗，因為虛幻編輯器會開啟另一個 Visual Studio 視窗，其中包含虛幻專案和您的遊戲專案。
4. 在「虛幻」編輯器中，執行下列任一項作業：

- 選擇您想要與 Amazon GameLift 整合的現有虛幻專案。
- 建立新專案。要嘗試使用虛幻的 Amazon GameLift 插件，請嘗試使用虛幻引擎的第三人稱模板。如需此範本的詳細資訊，請參閱虛幻引擎文件網站上的第三人稱視角範本。

  或者，使用下列設定來設定新專案：

  - C++
  - 隨著入門內容
  - 桌面
  - 專案名稱。在本主題的範例中，我們將專案命名為GameLiftUnrealApp。

5. 在 Visual Studio 的解決方案總管中，瀏覽至虛幻專案的位置。在「虛幻」Source 資料夾中，找到名為*Your-application-name*.Target.cs的檔案。

   例如：GameLiftUnrealApp.Target.cs。

6. 複製此檔案並命名副本：*Your-application-name*Server.Target.cs。
7. 開啟新檔案並進行下列變更：

   - 變更class和constructor以符合檔案名稱。
   - TargetType.Game將Type從變更為TargetType.Server。
   - 最後一個檔案看起來像下列範例：

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

您的專案現在已設定為接受 Amazon GameLift 伺服器 SDK 外掛程式。

下一個任務是為虛幻構建 C ++ 服務器 SDK 庫，以便您可以將它們導入到項目中。

為虛幻構建 C ++ 服務器 SDK 庫

1. 下載虛幻的 Amazon GameLift C ++ 服務器 SDK 插件。

   > **Note**
   >
   > 將 SDK 置於預設下載目錄中，可能會因為路徑超過 260 個字元限制而導致建置失敗。例如：`C:\Users\Administrator\Downloads\GameLift-SDK-Release-06_15_2023\GameLift-Cpp-ServerSDK-5.0.4`
   > 例如，我們建議您將 SDK 移至其他目錄`C:\GameLift-Cpp-ServerSDK-5.0.4`。

2. 下載 OpenSSL 裝 如需下載 OpenSSL 的詳細資訊，請參閱建置與安裝說明文件。

   如需詳細資訊，請參閱 Windows 平台適用的 OpenSSL 注意事項說明文件。

   > **Note**
   >
   > 您用來建立 Amazon GameLift 伺服器開發套件的 OpenSSL 版本應與虛幻用來封裝遊戲伺服器的 OpenSSL 版本相符。您可以在虛幻安裝目錄`...Engine\Source\ThirdParty\OpenSSL`中找到版本資訊。

3. 下載庫後，為虛幻引擎構建 C ++ 服務器 SDK 庫。

   在下載 SDK 的`GameLift-Cpp-ServerSDK-<version>`目錄中，使用`-DBUILD_FOR_UNREAL=1`參數進行編譯並構建服務器 SDK。下面的例子演示了如何使用編譯`cmake`.

   在終端機中執行下列命令：

   ```
   mkdir cmake-build
   cmake.exe -G "Visual Studio 17 2022" -DCMAKE_BUILD_TYPE=Release -S . -B ./cmake-build -DBUILD_FOR_UNREAL=1 -A x64
   cmake.exe --build ./cmake-build --target ALL_BUILD --config Release
   ```

   Windows 組建會在`out\gamelift-server-sdk\Release`資料夾中建立下列二進位檔案：

   - `cmake-build\prefix\bin\aws-cpp-sdk-gamelift-server.dll`
   - `cmake-build\prefix\bin\aws-cpp-sdk-gamelift-server.lib`

將這兩個程式庫檔案複製到 Amazon GameLift 虛幻引擎外掛程式套件中的ThirdParty\GameLiftServerSDK\Win64資料夾。

使用下列程序將 Amazon GameLift 外掛程式匯入範例專案。

導入 Amazon GameLift 插件

1. 在先前的程序中，找出您從外掛程式解壓縮的GameLiftServerSDK資料夾。
2. Plugins在您的遊戲項目根文件夾中找到。（如果該文件夾不存在，則在那裡創建它。）
3. 將GameLiftServerSDK資料夾複製到Plugins.

   這將允許虛幻項目看到插件。
4. 將 Amazon GameLift 服務器 SDK 插件添加到遊戲的.uproject文件中。

   在這個例子中，應用程序被調用GameLiftUnrealApp，所以該文件將是GameLiftUnrealApp.uproject。
5. 編輯.uproject檔案以將外掛程式新增至您的遊戲專案。

```
"Plugins": [
    {
        "Name": "GameLiftServerSDK",
        "Enabled": true
    }
]
```

6. 確保遊戲依賴於插件。 ModuleRules 開啟.Build.cs檔案並新增 Amazon GameLiftServer SDK 相依性。此檔案位於下*Your-application-name*/Source/*/Your-application-name/*。

   例如，教學課程檔案路徑為../GameLiftUnrealApp/Source/GameLiftUnrealApp/GameLiftUnrealApp.Build.cs。
7. 添加"GameLiftServerSDK"到列表的末尾PublicDependencyModuleNames。

```
using UnrealBuildTool;
using System.Collections.Generic;
public class GameLiftUnrealApp : ModuleRules
  {
    public GameLiftUnrealApp(TargetInfo Target)
    {
```

```
        PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
"Engine", "InputCore", "GameLiftServerSDK" });
        bEnableExceptions = true;
    }
}
```

該插件現在應該可以為您的應用程序工作。繼續下一節，將 Amazon GameLift 功能整合到您的遊戲中。

將 Amazon GameLift 服務器代碼添加到您的虛幻項目

您已經配置和設置了虛幻引擎環境，現在可以將遊戲服務器與 Amazon GameLift 集成。本主題中顯示的程式碼會對 Amazon GameLift 服務進行必要的呼叫。它還實現了一組回調函數，用於響應來自 Amazon GameLift 服務的請求。有關每個函數以及代碼功能的詳細信息，請參閱初始化服務器進程。有關此代碼中的 SDK 操作和數據類型的更多信息，請閱讀。虛幻引擎的亞馬遜GameLift服務器 SDK 參考

若要使用 Amazon 初始化遊戲伺服器 GameLift，請使用下列程序。

> **Note**
>
> 下一節提供的 Amazon GameLift 特定程式碼取決於`WITH_GAMELIFT`預處理器旗標的使用情況。只有在符合下列兩個條件時，此旗標才為真：
>
> - `Target.Type == TargetRules.TargetType.Server`
> - 這些插件找到了 Amazon GameLift 服務器 SDK 二進製文件。
>
> 這確保了只有虛幻服務器構建調用 Amazon GameLift 的後端 API。它還允許您編寫代碼，以便為您的遊戲可能產生的所有不同虛幻目標正確執行。

將遊戲伺服器與 Amazon 整合 GameLift

1. 在您的應用程式中開啟`.sln`檔案。在我們的例子中，該文件`GameLiftUnrealApp.sln`位於根文件夾中。
2. 開啟解決方案後，找出應用程式的*`Your-application-name`*`GameMode.h`檔案。範例：`GameLiftUnrealAppGameMode.h`。

3. 將標頭檔案變更為與下列範例程式碼對齊。請務必將 "GameLiftUnrealApp" 取代為您自己的應用程式名稱。

```
#pragma once

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

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    // Process Parameters needs to remain in scope for the lifetime of the app
    FProcessParameters m_params;

    void InitGameLift();
};
```

4. 開啟相關的來源*Your-application-name*GameMode.cpp檔案。在我們的 Example:GameLiftUnrealAppGameMode.cpp. 中，並將代碼更改為與以下示例代碼對齊。請務必將 "GameLiftUnrealApp" 取代為您自己的應用程式名稱。

   此範例說明如何新增與 Amazon 整合的所有必要元素 GameLift，如[將 Amazon 新增 GameLift至遊戲伺服器](#)中所述。其中包含：

   - 初始化 Amazon GameLift API 客戶端。
   - 實施回調函數以響應來自 Amazon GameLift 服務的請求`OnStartGameSession`，包括`OnProcessTerminate`、和`onHealthCheck`。

- 使用指定的連接埠呼叫 ProcessReady ()，在準備好主持遊戲工作階段 GameLiftservice 時通知 Amazon。

```
#include "GameLiftUnrealAppGameMode.h"
#include "GameLiftUnrealAppCharacter.h"

#include "UObject/ConstructorHelpers.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode()
{
    // 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;
    }
}

void AGameLiftUnrealAppGameMode::BeginPlay()
{
#if WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
    UE_LOG(GameServerLog, Log, TEXT("Initializing the GameLift Server"));

    //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 serverParameters;

    //AuthToken returned from the "aws gamelift get-compute-auth-token" API. Note
 this will expire and require a new call to the API after 15 minutes.
```

```
   if (FParse::Value(FCommandLine::Get(), TEXT("-authtoken="),
serverParameters.m_authToken))
   {
       UE_LOG(GameServerLog, Log, TEXT("AUTH_TOKEN: %s"),
*serverParameters.m_authToken)
   }

   //The Host/compute-name of the GameLift Anywhere instance.
   if (FParse::Value(FCommandLine::Get(), TEXT("-hostid="),
serverParameters.m_hostId))
   {
       UE_LOG(GameServerLog, Log, TEXT("HOST_ID: %s"), *serverParameters.m_hostId)
   }

   //The Anywhere Fleet ID.
   if (FParse::Value(FCommandLine::Get(), TEXT("-fleetid="),
serverParameters.m_fleetId))
   {
       UE_LOG(GameServerLog, Log, TEXT("FLEET_ID: %s"),
*serverParameters.m_fleetId)
   }

   //The WebSocket URL (GameLiftServiceSdkEndpoint).
   if (FParse::Value(FCommandLine::Get(), TEXT("-websocketurl="),
serverParameters.m_webSocketUrl))
   {
       UE_LOG(GameServerLog, Log, TEXT("WEBSOCKET_URL: %s"),
*serverParameters.m_webSocketUrl)
   }

   //The PID of the running process
   serverParameters.m_processId = FString::Printf(TEXT("%d"),
GetCurrentProcessId());
   UE_LOG(GameServerLog, Log, TEXT("PID: %s"), *serverParameters.m_processId);

   //InitSDK establishes a local connection with GameLift's agent to enable
further communication.
   //Use InitSDK(serverParameters) for a GameLift Anywhere fleet.
   //Use InitSDK() for a GameLift managed EC2 fleet.
   gameLiftSdkModule->InitSDK(serverParameters);

   //Implement callback function onStartGameSession
   //GameLift sends a game session activation request to the game server
   //and passes a game session object with 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().
    auto onGameSession = [=](Aws::GameLift::Server::Model::GameSession gameSession)
    {
        FString gameSessionId = FString(gameSession.GetGameSessionId());
        UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*gameSessionId);
        gameLiftSdkModule->ActivateGameSession();
    };

    m_params.OnStartGameSession.BindLambda(onGameSession);

    //Implement callback function OnProcessTerminate
    //GameLift 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 GameLift it is shutting down.
    auto onProcessTerminate = [=]()
    {
        UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
        gameLiftSdkModule->ProcessEnding();
    };

    m_params.OnTerminate.BindLambda(onProcessTerminate);

    //Implement callback function OnHealthCheck
    //GameLift 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 GameLift records 'false'.
    //In this example, the game server always reports healthy.
    auto onHealthCheck = []()
    {
        UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
        return true;
    };

    m_params.OnHealthCheck.BindLambda(onHealthCheck);

    //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.
    m_params.port = 7777;

    //Here, the game server tells GameLift where to find game session log files.
    //At the end of a game session, GameLift uploads everything in the specified
    //location and stores it in the cloud for access later.
    TArray<FString> logfiles;
    logfiles.Add(TEXT("GameLift426Test/Saved/Logs/GameLift426Test.log"));
    m_params.logParameters = logfiles;

    //The game server calls ProcessReady() to tell GameLift it's ready to host game
 sessions.
    UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
    gameLiftSdkModule->ProcessReady(m_params);
}
```

5. 為下列兩種目標類型建置遊戲專案：開發編輯器和開發伺服器。

> **Note**
>
> 您不需要重建解決方案。相反，只在與您的應用程序名稱匹配的Games文件夾下構建項目。否則，視覺工作室重建整個 UE5 專案，這可能需要長達一個小時。

6. 兩個組建都完成後，請關閉 Visual Studio 並開啟專.uproject案的檔案，以便在虛幻編輯器中開啟。
7. 在虛幻編輯器中，封裝遊戲的伺服器組建。若要選擇目標，請移至「平台」、「視窗」，然後選取「*Your-application-name* ###」。
8. 要開始構建服務器應用程序的過程，請轉到平台，Windows，然後選擇 P ackage 項目。當構建完成時，您應該有一個可執行文件。在我們的例子中，文件名是GameLiftUnrealAppServer.exe。
9. 在虛幻編輯器中構建服務器應用程序會產生兩個可執行文件。其中一個位於遊戲組建資料夾的根目錄中，並做為實際伺服器可執行檔的包裝函式。

   使用伺服器組建建建立 Amazon GameLift 叢集時，建議您傳入實際的伺服器可執行檔做為執行階段組態啟動路徑。例如，在您的遊戲構建文件夾中，您可能在根目錄中有一個GameLiftFPS.exe文件，另一個文件位於\GameLiftFPS\Binaries\Win64\GameLiftFPSServer.exe。建立叢集時，建議您使用C:\GameLiftFPS\Binaries\Win64\GameLiftFPSServer.exe做為執行階段設定的啟動路徑。

10. 請務必在 Amazon GameLift 叢集上開啟必要的 UDP 連接埠，以便遊戲伺服器可以與遊戲用戶端進行通訊。默認情況下，虛幻引擎使用端口7777。如需詳細資訊，請參閱 Amazon GameLift 服務 API 參考指南UpdateFleetPortSettings中的。

11. 為您的遊戲構建創建一個install.bat文件。只要將遊戲組建部署到 Amazon GameLift 叢集，就會執行此安裝指令碼。這是一個示例install.bat文件：

    ```
    VC_redist.x64.exe /q
    UE5PrereqSetup_x64.exe /q
    ```

    對於某些版本的虛幻引擎，install.bat應該是

    ```
    VC_redist.x64.exe /q
    UEPrereqSetup_x64.exe /q
    ```

    > **Note**
    >
    > 檔案的檔<>PrereqSetup_x64.exe案路徑為Engine\Extras\Redist\en-us。

12. 現在，您可以打包並上傳您的遊戲構建到 Amazon GameLift。

    您在遊戲組建中封裝的 OpenSSL 版本必須符合遊戲引擎在建置遊戲伺服器時所使用的版本。請務必將正確的 OpenSSL 版本與您的遊戲伺服器組建一起封裝。對於視窗作業系統，OpenSSL 格式為.dll.

    > **Note**
    >
    > 在您的遊戲伺服器組建中 Package OpenSSL DLL 檔案。請務必封裝您在建置遊戲伺服器時所使用的 OpenSSL 版本相同。
    >
    > - libssl-1_1-x64.dll
    >
    >   libcrypto-1_1-x64.dll
    >
    > 將您的依賴項與遊戲服務器可執行文件一起打 Package 到 zip 文件的根目錄中。例如，openssl-libdll 應該位於與.exe文件相同的目錄中。

後續步驟

您已經配置並設置了虛幻引擎環境，現在可以開始 GameLift 將 Amazon 集成到您的遊戲中。

如需將 Amazon 新增 GameLift 至遊戲的詳細資訊，請參閱下列內容：

- 添加亞馬遜GameLift到您的遊戲服務器
- 虛幻引擎的亞馬遜GameLift服務器 SDK 參考

如需測試遊戲的指示，請參閱使用亞馬遜機GameLiftAnywhere隊測試您的整合。

GameLift 將亞馬遜整合到統一項目中

本主題說明如何為 Unity 設定 Amazon GameLift C# 伺服器 SDK 外掛程式，並將其整合到您的遊戲專案中。

其他資源：

- 亞馬遜 GameLift 服務器 SDK 下載網站
- 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考
- the section called “開發支援”

先決條件

若要使用亞馬遜 GameLift C# 伺服器 SDK 外掛程式的統一，您需要下列元件：

- 外掛程式支援的開發環境和 Unity 編輯器版本 (請參閱Amazon 的開發支持 GameLift)。如需有關 Unity 版本的資訊，請參閱 Unity 文件中的 Unity 的系統需求。
- 亞馬遜 GameLift 服務器 SDK 插件統一包。此套件包含適用於 C# 的伺服器 SDK 5+。您可以從此網站下載軟件包：亞馬遜入門 GameLift。
- 第三方範圍的登錄。 UnityNuGet此工具管理協力廠商 DLL。如需詳細資訊，請參閱 UnityNuGetGithub 儲存庫。

設定 UnityNuGet

如果您尚未為您的遊戲專案 UnityNuGet 設定，請使用下列步驟使用 Unity 套件管理員安裝工具。或者，您可以使用 NuGet CLI 手動下載 DLL。如需詳細資訊，請參閱適用於統一的 Amazon GameLift C# 伺服器開發套件README。

整合 UnityNuGet 到您的遊戲專案

1. 在 Unity 編輯器中打開項目後，轉到主菜單並選擇編輯，項目設置。從選項中，選擇「Package 件管理員」區段，然後開啟「範圍登錄」群組。
2. 選擇 + 按鈕，並為 UnityNuGet 範圍登錄輸入下列值：

   ```
   Name: Unity NuGet
   URL: https://unitynuget-registry.azurewebsites.net
   Scope(s): org.nuget
   ```

3. 對於統一 2021 版本的用戶：

   設置後 UnityNuGet，檢查 Unity 控制台中顯示的`Assembly Version Validation`錯誤。如果NuGet封裝中強式名稱組件的繫結重新導向未正確解析為 Unity 專案中的路徑，就會發生這些錯誤。若要解決此問題，請設定 Unity 的組件版本驗證：

   a. 在 Unity 編輯器中，轉到主菜單，然後選擇編輯，項目設置，然後打開播放器部分。
   b. 取消選取「組件版本驗證」選項。

安裝插件

使用下列程序安裝適用於 Unity 的 Amazon GameLift C# 伺服器 SDK 外掛程式，並設定 log4net 記錄。

安裝外掛程式

1. 在 Unity 編輯器中打開項目後，轉到主菜單並選擇窗口，P ackage 管理器。
2. 選擇「+」按鈕以新增套件。選擇選項從壓縮包中添加包裹。
3. 在選取磁碟上的套件中，找出 Unity 下載檔案的 Amazon GameLift C# 伺服器 SDK 外掛程式，然後選擇 Amazon GameLift 伺服器 SDK `.tgz` 檔案。選擇「開啟」以安裝外掛程式。

亞馬遜 GameLift 服務器開發套件使用 log4net 框架來輸出日誌消息。默認情況下，它被配置為將消息輸出到服務器構建的終端，但 Unity 需要配置以添加文件日誌記錄支持。您可以在 Amazon GameLift Server SDK 套件中匯入提供的範例，將此支援新增至您的專案。請使用下列程序來新增範例並設定 log4net：

若要為檔案輸出設定 log4net

1. 在 Unity 編輯器中打開項目後，轉到主菜單並選擇窗口，P ackage 管理器。

2. 從下拉式功能表中，選取套件：在專案中，然後從套件清單中選取 Amazon GameLift Server SDK。這會開啟套件詳細資訊。
3. 在套件詳細資訊中，選取 [範例] 群組選項，然後按 [匯入]。
4. 該`log4net.config`文件和隨附的`LoggingConfiguration.cs`腳本會自動執行配置，該配置現在已在項目的文件`Assets/Samples`夾中進行設置。

> **Note**
>
> 如果您需要將`log4net.config`檔案移至專案中的其他資料夾，您也必須`LoggingConfiguration.cs`使用新路徑更新指令碼中的組態檔案路徑。如需詳細資訊，請參閱有關配置 log4net 的 log4net 手冊。

有關更詳細的說明和測試指導，請參閱插件下載中的。`README`

建立亞馬遜 GameLift Anywhere車隊進行測試

您可以將開發工作站設定為 Amazon GameLift Anywhere 託管叢集，以反覆測試您的 Amazon GameLift 整合。使用此設定，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GameLift至 Amazon 以開始遊戲工作階段，以及將用戶端連接到新的遊戲工作階段。將您自己的工作站設置為託管服務器後，您可以監控與 Amazon 的遊戲整合的各個方面 GameLift。

如需設定工作站的指示，請參閱使用亞馬遜機GameLiftAnywhere隊測試您的整合以完成下列步驟：

1. 為您的工作站建立自訂位置。
2. 使用您的新自訂位置建立 Amazon GameLift Anywhere 叢集。如果成功，此要求會傳回叢集 ID。請記下此值，因為您稍後會需要它。
3. 將您的工作站註冊為新Anywhere叢集中的運算。提供唯一的運算名稱，並指定工作站的 IP 位址。如果成功，此要求會以 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. 為您的工作站計算產生驗證 Token。這種短期身份驗證包括令牌和到期日期。您的遊戲伺服器會使用它來驗證與 Amazon GameLift 服務的通訊。將驗證儲存在您的工作站計算機上，以便執行中的遊戲伺服器程序可以存取它。

添加亞馬遜 GameLift 服務器代碼到您的 Unity 項目

您的遊戲伺服器會與 Amazon GameLift 服務進行通訊，以接收指示並報告進行中的狀態。若要完成此操作，您可以新增使用 Amazon 伺服器 SDK 的遊戲 GameLift伺服器程式碼。

提供的程式碼範例說明基本必要的整合元素。它使用 a 來說明MonoBehavior使用亞馬遜進行簡單的遊戲服務器初始化 GameLift。此範例假設遊戲伺服器在 Amazon GameLift Anywhere 叢集上執行以進行測試。它包括以下代碼：

- 初始化亞馬遜 GameLift API 客戶端。此範例使用的版本InitSDK( )與伺服器參數適用於您的 Anywhere叢集和運算。使用 WebSocket URL、叢集 ID、計算名稱 (主機 ID) 和驗證 Token，如上一個主題所定義建立亞馬遜 GameLift Anywhere車隊進行測試。
- 實作回呼函數以回應 Amazon GameLift 服務的請求OnStartGameSession，包括OnProcessTerminate、和onHealthCheck。
- 使用指定的連接埠呼叫 ProcessReady ()，以在程序準備好主持遊戲工作階段時通知 Amazon GameLift 服務。

本主題中介紹的程式碼建立與 Amazon GameLift 服務和的通訊。它還實現了一組回調函數，以響應來自. 有關每個函數以及代碼功能的詳細信息，請參閱初始化服務器進程。有關此代碼中使用的 SDK 操作和數據類型的更多信息，請閱讀C# 的亞馬遜GameLift伺服器 SDK 參考。

此範例顯示如何新增所有必要元素，如將 Amazon 新增 GameLift 至遊戲伺服器中所述。它包括：

如需新增 Amazon GameLift 功能的詳細資訊，請參閱下列主題：

- 添加亞馬遜GameLift到您的遊戲服務器
- C# 的亞馬遜GameLift伺服器 SDK 參考

```
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 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 a GameLift Anywhere fleet.
        //They are not required for a GameLift managed EC2 fleet.
        ServerParameters serverParameters = new ServerParameters(
            webSocketUrl,
            processId,
            hostId,
            fleetId,
            authToken);

        //InitSDK establishes a local connection with an Amazon GameLift agent
        //to enable further communication.
        var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
        if (initSDKOutcome.Success)
        {
            //Implement callback functions
            ProcessParameters processParameters = new ProcessParameters(
            //Implement OnStartGameSession callback
                (gameSession) => {
                    //GameLift 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) => {
```

```
                    //GameLift 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
                    //GameLift 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 GameLift it is shutting
down.
                    GameLiftServerAPI.ProcessEnding();
                },
                () => {
                    //Implement callback function OnHealthCheck
                    //GameLift 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 GameLift
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 GameLift where to find game session
log files.
                    //At the end of a game session, GameLift 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 GameLift 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 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);
        }
    }
}
```

其他資源

使用下列資源來測試您的遊戲伺服器並擴充功能：

- 將您的開發機器設定為 Amazon GameLift Anywhere 叢集，並將其用於本機測試。請參閱測試您的自訂伺服器整合。
- 構建您的遊戲服務器並將構建上傳到亞馬遜 GameLift。請參閱將自訂伺服器組建上傳到 Amazon GameLift。
- 將您的遊戲伺服器組建部署到 Amazon GameLift 受管 EC2 叢集。請參閱建立新的 Amazon GameLift 叢集。

使用亞馬遜機GameLiftAnywhere隊測試您的整合

您可以使用 Amazon GameLift Anywhere 叢集反覆建置和測試您與 Amazon 的遊戲整合。GameLift將您自己的硬體設定為連線至 Amazon GameLift 服務的Anywhere叢集，然後在其上安裝並執行遊戲伺服器。使用測試應用程序運行諸如開始/停止遊戲會話，跟踪玩家連接以及處理配對補充等情況。透過Anywhere叢集，您可以視需要更新遊戲伺服器組建，並完全掌握主機活動。

您可以將亞馬遜GameLiftAnywhere車隊與亞馬遜GameLift服務器 SDK 版本 5 或更高版本集成的遊戲。

主題

- 初步發展
- 在遊戲伺服器上進行迭代

初步發展

您已經開發了自己的遊戲，並將其與 Amazon GameLift 伺服器開發套件整合。為了測試您的整合，您可以將遊戲伺服器組建的每個新版序上傳到 Amazon GameLift 並建立叢集。或者，搭配開發筆記型電腦使用Anywhere機隊，可讓您以更有效率的方式進行反覆式開發和測試。

使用下列程序建立Anywhere叢集，並使用 Amazon GameLift 主控台或 AWS Command Line Interface (AWS CLI) 在筆記型電腦上啟動遊戲工作階段。

Console

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格的 [主機] 下，選擇 [位置]。
3. 選擇 [建立地點]。
4. 在「建立位置」對話方塊中，執行下列操作：

a. 輸入「地點」名稱。這會標示 Amazon GameLift 用來在Anywhere叢集中執行遊戲的運算資源位置。自訂位置名稱必須以自訂- 開頭。

b. 選擇 建立 。

5. 若要建立Anywhere叢集，請執行下列動作：

a. 在功能窗格的 [主機] 底下，選擇 [叢集]。

b. 在 Fleets (機群) 頁面上，選擇 Create fleet (建立機群)。

c. 在 [選擇運算類型] 步驟上，選擇 [任何地方]，然後選擇 [下一步]

d. 在「定義叢集詳細資訊」步驟中，定義新叢集。如需詳細資訊，請參閱建立新的亞馬遜GameLift車隊。

e. 在 「選取位置」步驟中，選取您建立的自訂位置。

f. 完成剩餘的叢集建立步驟以建立您的Anywhere叢集。

6. 在您建立的叢集中，將筆記型電腦註冊為運算資源。使用register-compute命令（或RegisterComputeAPI 操作）。包括在上一步中fleet-id創建的，並添加compute-name和您的筆記本電腦ip-address。

```
aws gamelift register-compute \
    --compute-name DevLaptop \
    --fleet-id fleet-1234 \
    --ip-address 10.1.2.3 \
    --location custom-location-1
```

輸出範例：

```
Compute {
    FleetId = fleet-1234,
    ComputeName = DevLaptop,
    Status = ACTIVE,
    IpAddress = 10.1.2.3,
    GameLiftServiceSdkEndpoint = wss://12345678.execute-api.amazonaws.com/,
    Location = custom-location-1
}
```

7. 啟動遊戲伺服器的除錯工作階段。

a. 在您創建的機群中獲取筆記本電腦的授權令牌。使用get-compute-auth-token命令（或 GetComputeAuthTokenAPI 操作）。

```
aws gamelift get-compute-auth-token \
    --fleet-id fleet-1234 \
    --compute-name DevLaptop
```

輸出範例：

```
ComputeAuthToken {
  FleetId = fleet-1234,
  ComputeName = DevLaptop,
  AuthToken = abcdefg123,
  ExpirationTime = 1897492857.11
}
```

b. 執行遊戲伺服器執行檔的除錯執行個體。若要執行除錯執行個體，您的遊戲伺服器必須呼叫InitSDK()。在該過程準備好主持遊戲會話之後，遊戲服務器會調用ProcessReady()。

8. 建立遊戲工作階段以測試您與 Amazon 的首次整合GameLiftAnywhere。使用create-game-session命令（或 CreateGameSessionAPI 操作）。指定叢集的自訂位置。

```
aws gamelift create-game-session \
    --fleet-id fleet-1234 \
    --name DebugSession \
    --maximum-player-session-count 2 \
    --location custom-location-1
```

輸出範例：

```
GameSession {
    FleetId = fleet-1234,
    GameSessionId = 1111-1111,
    Name = DebugSession,
    IpAddress = 10.1.2.3,
    Port = 1024,
    ...
}
```

Amazon GameLift 會將onStartGameSession()訊息傳送到您註冊的伺服器程序。訊息包含上一步的GameSession物件，其中包含遊戲屬性、遊戲工作階段資料、分房系統資料，以及有關遊戲工作階段的更多資訊。

9. 將邏輯添加到您的遊戲服務器，以便您的服務器進程響應onStartGameSession()消息ActivateGameSession()。該操作向亞馬遜發送確認，表明您GameLift的服務器收到並接受了創建遊戲會話消息。如需詳細資訊，請參閱 亞馬遜GameLift服務器 SDK 參考。

您的遊戲伺服器現在正在執行遊戲工作階段，供您測試並用於反覆運算。若要瞭解如何在遊戲伺服器上重複執行，請繼續下一節。

AWS CLI

1. 使用create-location指令 (或 CreateLocationAPI 作業) 建立自訂位置。自訂位置會標示 Amazon GameLift 用來在Anywhere叢集中執行遊戲的硬體位置。

```
aws gamelift create-location \
    --location-name custom-location-1
```

輸出範例：

```
{
    Location {
        LocationName = custom-location-1
    }
}
```

2. 使用create-fleet指令 (或 CreateFleetAPI 作業) 建立具有您自訂位置的Anywhere叢集。Amazon GameLift 會在您的本地區域和您提供的自訂位置建立叢集。

```
aws gamelift create-fleet \
    --name LaptopFleet \
    --compute-type ANYWHERE \
    --locations "location=custom-location-1"
```

輸出範例：

```
Fleet {
    Name = LaptopFleet,
    ComputeType = ANYWHERE,
    FleetId = fleet-1234,
    Status = ACTIVE
    ...
}
```

3. 在您建立的叢集中，將筆記型電腦註冊為運算資源。使用register-compute命令（或RegisterComputeAPI 操作）。包括在上一步中fleet-id創建的，並添加一個compute-name和您的筆記本電腦的公共ip-address。

```
aws gamelift register-compute \
    --compute-name DevLaptop \
    --fleet-id fleet-1234 \
    --ip-address 10.1.2.3 \
    --location custom-location-1
```

輸出範例：

```
Compute {
    FleetId = fleet-1234,
    ComputeName = DevLaptop,
    Status = ACTIVE,
    IpAddress = 10.1.2.3,
    GameLiftServiceSdkEndpoint = wss://12345678.execute-api.amazonaws.com/,
    Location = custom-location-1
}
```

4. 啟動遊戲伺服器的除錯工作階段。

   a. 在您創建的機群中獲取筆記本電腦的授權令牌。使用get-compute-auth-token命令（或GetComputeAuthTokenAPI 操作）。

```
aws gamelift get-compute-auth-token \
    --fleet-id fleet-1234 \
    --compute-name DevLaptop
```

輸出範例：

```
ComputeAuthToken {
  FleetId = fleet-1234,
  ComputeName = DevLaptop,
  AuthToken = abcdefg123,
  ExpirationTime = 1897492857.11
}
```

b. 執行遊戲伺服器執行檔的除錯執行個體。若要執行除錯執行個體，您的遊戲伺服器必須呼叫InitSDK()。在該過程準備好主持遊戲會話之後，遊戲服務器會調用ProcessReady()。

5. 建立遊戲工作階段以測試您與 Amazon 的首次整合GameLiftAnywhere。使用create-game-session命令（或 CreateGameSessionAPI 操作）。

```
aws gamelift create-game-session \
    --fleet-id fleet-1234 \
    --name DebugSession \
    --maximum-player-session-count 2
```

輸出範例：

```
GameSession {
    FleetId = fleet-1234,
    GameSessionId = 1111-1111,
    Name = DebugSession,
    IpAddress = 10.1.2.3,
    Port = 1024,
    ...
}
```

Amazon GameLift 會將onStartGameSession()訊息傳送到您註冊的伺服器程序。訊息包含上一步的GameSession物件，其中包含遊戲屬性、遊戲工作階段資料、分房系統資料，以及有關遊戲工作階段的更多資訊。

6. 將邏輯添加到您的遊戲服務器，以便您的服務器進程響應onStartGameSession()消息ActivateGameSession()。該操作向亞馬遜發送確認，表明您GameLift的服務器收到並接受了創建遊戲會話消息。如需詳細資訊，請參閱 亞馬遜GameLift服務器 SDK 參考。

您的遊戲伺服器現在正在執行遊戲工作階段，供您測試並用於反覆運算。若要瞭解如何在遊戲伺服器上重複執行，請繼續下一節。

在遊戲伺服器上進行迭代

在此使用案例中，請考慮您已設定並測試遊戲伺服器並發現錯誤的案例。使用亞馬遜GameLiftAnywhere，您可以對代碼進行迭代，避免使用 Amazon EC2 叢集的繁重設置。

1. 如果可能的話GameSession，請清理您現有的。如果遊戲服務器崩潰或無法調用ProcessEnding()，亞馬遜會在遊戲服務器停止發送運行狀態檢查GameSession後進行GameLift清理。
2. 對您的遊戲伺服器進行程式碼變更、編譯並為下一次測試做準備。
3. 您之前的Anywhere機隊仍處於作用中狀態，而您的筆記型電腦仍在叢集中註冊為運算資源。要再次開始測試，請創建一個新的調試實例。

   a. 在您創建的機群中檢索筆記本電腦的授權令牌。使用get-compute-auth-token命令（或GetComputeAuthTokenAPI 操作）。

   ```
   aws gamelift get-compute-auth-token \
       --fleet-id fleet-1234 \
       --compute-name DevLaptop
   ```

   輸出範例：

   ```
   ComputeAuthToken {
     FleetId = fleet-1234,
     ComputeName = DevLaptop,
     AuthToken = hijklmnop456,
     ExpirationTime = 1897492857.11
   }
   ```

   b. 執行遊戲伺服器執行檔的除錯執行個體。若要執行除錯執行個體，您的遊戲伺服器必須呼叫InitSDK()。在該過程準備好主持遊戲會話之後，遊戲服務器會調用ProcessReady()。
4. 您的叢集現在有可用的伺服器處理序。創建您的遊戲會話並執行下一個測試。使用create-game-session命令（或CreateGameSessionAPI 操作）。

   ```
   aws gamelift create-game-session \
       --fleet-id fleet-1234 \
       --name SecondDebugSession \
       --maximum-player-session-count 2
   ```

   Amazon GameLift 會將onStartGameSession()訊息傳送到您註冊的伺服器程序。訊息包含上一步的GameSession物件，其中包含遊戲屬性、遊戲工作階段資料、分房系統資料，以及有關遊戲工作階段的更多資訊。

5. 將邏輯添加到您的遊戲服務器，以便您的服務器進程響應onStartGameSession()消息ActivateGameSession()。該操作向亞馬遜發送確認，表明您GameLift的服務器收到並接受了創建遊戲會話消息。如需詳細資訊，請參閱 亞馬遜GameLift服務器 SDK 參考。

完成遊戲伺服器測試後，您可以繼續使GameLift用 Amazon 管理叢集和遊戲伺服器。如需詳細資訊，請參閱 創建一個 Amazon GameLift Anywhere 車隊。

使用亞馬遜GameLift本地測試您的整合

> **Note**
>
> 如果您使用的是 4.x 版或更早版本的 Amazon GameLift 伺服器開發套件版本，請使用此測試程序。您的伺服器 SDK 套件包含相容版本的亞馬遜本機版GameLift本。如果您使用的是伺服器 SDK 版本 5.x，請參閱使使用亞馬遜機GameLiftAnywhere隊測試您的整合用 Amazon GameLift Anywhere 叢集進行本機測試。

使用 Amazon GameLift 本機在本機裝置上執行受管 Amazon GameLift 服務的限制版本，並測試您的遊戲整合。此工具在對遊戲整合進行重複性開發時非常有用。另一種方式 — 將每個新組建上傳到 Amazon GameLift 並設定叢集來託管您的遊戲 — 每次可能需要 30 分鐘或更長時間。

使用亞馬遜GameLift本地，您可以驗證以下內容：

- 您的遊戲伺服器已與 Server SDK 正確整合，並與 Amazon GameLift 服務正確通訊，以開始新的遊戲工作階段、接受新玩家，以及報告健康狀態和狀態。
- 您的遊戲用戶端已與適用於 Amazon 的 AWS SDK 正確整合，GameLift並能擷取現有遊戲工作階段的相關資訊、開始新的遊戲工作階段、加入玩家參與遊戲，以及連線到遊戲工作階段。

亞馬遜本GameLift地是一種命令列工具，可啟動受管亞馬遜GameLift服務的獨立版本。Amazon L GameLift ocal 也提供執行中的事件日誌，其中包含伺服器處理序初始化、運作狀態檢查以及 API 呼叫和回應。亞馬遜GameLift本地識別亞馬遜的 AWS SDK 操作的一個子集GameLift。您可以從 AWS CLI 或從您的遊戲用戶端進行呼叫。所有 API 動作都會在本機執行，就像在 Amazon GameLift Web 服務中一樣。

每個服務器進程只能託管一個遊戲會話。遊戲工作階段是您用來連線到 Amazon GameLift 本地端的可執行檔。遊戲會話完成後，您應該調用GameLiftServerSDK::ProcessEndning然後退出該過程。使用 Amazon GameLift 本機進行本機測試時，您可以啟動多個伺服器程序。每個過程都將連接到亞馬

遜GameLift本地。然後，您可以為每個服務器進程創建一個遊戲會話。當遊戲工作階段結束時，您的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處理序。

亞馬遜GameLift本地支持以下 API：

- CreateGameSession
- CreatePlayerSession
- CreatePlayerSessions
- DescribeGameSessions
- DescribePlayerSessions

設置亞馬遜GameLift本地

亞馬遜GameLift本地作為與服務器 SDK 捆綁在一起的可執行.jar文件提供。它可以在 Windows 或 Linux 上運行，並與任何亞馬遜GameLift支持的語言一起使用。

執行 Local 之前，您必須同時安裝下列項目。

- 亞馬遜GameLift服務器開發套件 3.1.5 到 4.x 版的構建。
- Java 8

測試遊戲伺服器

如果您只想測試遊戲伺服器，可以使用模擬遊戲AWS CLI用戶端對 Amazon GameLift 本地服務的呼叫。這會驗證您的遊戲伺服器是否依下列操作執行：

- 遊戲伺服器會正確啟動並初始化亞馬遜GameLift伺服器開發套件。
- 在啟動過程中，遊戲伺服器會通知 Amazon GameLift 伺服器已準備好託管遊戲工作階段。
- 遊戲服務器在運行時GameLift每分鐘都會向亞馬遜發送健康狀態。
- 遊戲伺服器回應請求以啟動新遊戲工作階段。

1. 啟動亞馬遜GameLift本地。

   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GameLift整合 Amazon 的遊戲伺服器。您不需要變更遊戲伺服器的端點。

   在 [本機命令提示字元] 視窗中，記錄訊息指出您的遊戲伺服器已連線至 Amazon GameLift 本機服務。這表示您的遊戲伺服器成功初始化了 Amazon GameLift 伺服器 SDK (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。在遊戲伺服器執行時，Amazon 會GameLift記錄來自遊戲伺服器的每個健康狀態報告。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沒有找到 PID: 的處理程序:<process ID>! 是否被調用了？」
   - 警告：「使用 PID 進程的進程狀態已經存在：<process ID>! 是ProcessReady (...) 不止一次被調用？」

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默認情況下使用亞馬遜GameLift網絡服務端點。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
```

在指AWS CLI令提示視窗中，AWS gamelift指令會產生回應，如《指AWS CLI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 [本機命令提示字元] 視窗中，記錄訊息指出 Amazon L GameLift ocal 已傳送您的遊戲伺服器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，Amazon 會以包含遊戲工作階段 ID 的遊戲工作階段物件進行GameLift回應。請注意，新遊戲工作階段的狀態為 Activating (啟動中)。一旦您的遊戲伺服器呼叫 ActivateGameSession，狀態就會變更為「作用中」。若要查看狀態變更，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。這可讓您測試從遊戲用戶端到 Amazon GameLift 本機的程式設計呼叫。您可以驗證下列動作：

- 遊戲用戶端已成功向 Amazon L GameLift ocal 服務發出 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. 啟動亞馬遜GameLift本地。

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GameLift整合 Amazon 的遊戲伺服器。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 L GameLift ocal 服務搭配使用，您必須對遊戲用戶端的設定進行下列變更，如中所述在後端服務 GameLift 上設置 Amazon：

   - 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 [本機命令提示字元] 視窗中，一旦啟動遊戲用戶端，記錄訊息應指出它已初始化，`GameLiftClient`並且已成功與 Amazon GameLift 服務通訊。

4. 測試遊戲用戶端對 Amazon GameLift 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。Amazon L GameLift ocal 提示您的遊戲伺服器啟動遊戲工作階段 (`onStartGameSession`回呼)，並在遊戲伺服器叫用時取得成功`ActivateGameSession`時，記錄訊息會顯示。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。如果您透過 AWS CLI 來呼叫 `DescribePlayerSessions()`，玩家工作階段的狀態應該會從 Reserved (保留) 變更為 Active (作用中)。

6. 確認您的遊戲伺服器正在向 Amazon GameLift 服務回報遊戲和玩家狀態。

   為了GameLift讓 Amazon 管理玩家需求並正確報告指標，您的遊戲伺服器必須向 Amazon 報告各種狀態GameLift。驗證 Local 是否正在記錄與下列動作相關的事件。您可能也想要使用 AWS CLI 追蹤狀態變更。

   - 玩家與遊戲工作階段中斷連線 — Amazon GameLift 本機記錄訊息應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 本機記錄訊息應顯示您的遊戲伺服器呼叫`TerminateGameSession()`。

     > **Note**
     >
     > 以前的指導是在結束遊戲會話`TerminateGameSession()`時調用。此方法已與亞馬遜GameLift服務器 SDK v4.0.1 棄用。請參閱 結束遊戲工作階段。

   - 伺服器程序已終止 — Amazon GameLift 本機日誌訊息應顯示您的遊戲伺服器呼叫`ProcessEnding()`。此外，系統對 `DescribeGameSessions()` 所進行的 AWS CLI 呼叫，應該會反映出狀態從 `Active` 變更為 `Terminated` (或 `Terminating`)。

與本地的變化

使用亞馬遜GameLift本地時，請記住以下幾點：

- 與亞馬遜GameLift網絡服務不同，本地不會跟踪服務器的運行狀態並啟動onProcessTerminate回調。Local 只會停止記錄遊戲伺服器的執行狀況報告。
- 使用 AWS 開發套件呼叫時，機群 ID 不需經過驗證，且可以是符合參數需求 (^fleet-\S+) 的任何字串值。
- 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
```

將遊戲與 Amazon GameLift 即時伺服器整合

本主題提供受管 Amazon GameLift 與即時伺服器解決方案的概觀。概述解釋了此解決方案何時適合您的遊戲，以及實時服務器如何支持多人遊戲。

如需啟動並執行遊戲的完整藍圖，請參閱亞馬遜GameLift託管路線圖。

> **Tip**
>
> 若要試用 Amazon GameLift 遊戲伺服器託管，請參閱開始使用亞馬遜 GameLift。

什麼是實時服務器？

實時服務器是亞馬遜為您GameLift提供的輕量級ready-to-go遊戲服務器，供您與多人遊戲一起使用。即時伺服器會移除自訂遊戲伺服器的開發、測試和部署程序。此解決方案有助於最大限度地減少完成遊戲所需的時間和精力。

主要功能

- 用於遊戲客戶端和服務器交互的全網絡堆棧
- 核心遊戲伺服器功能
- 自訂伺服器邏輯
- 實時更新實時配置和服務器邏輯
- FlexMatch配對
- 靈活控制託管資源

透過建立叢集並提供設定指令碼來設定即時伺服器。有關創建實時服務器以及如何準備遊戲客戶端的更多信息，請參閱準備您的實時服務器。

實時服務器如何管理遊戲會話

您可以通過將其構建到實時腳本中來為遊戲會話管理添加自定義邏輯。您可以撰寫程式碼來存取伺服器特定物件、使用回呼新增事件驅動邏輯，或根據非事件案例新增邏輯。

實時客戶端和服務器如何交互

在遊戲會話期間，遊戲客戶端通過後端服務向實時服務器發送消息進行交互。接著，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，您可以通過將遊戲邏輯添加到實時腳本來自定義客戶端和服務器的交互方式。使用自定義遊戲邏輯，實時服務器可能會實現回調以啟動事件驅動的響應。

通訊協定

實時服務器和連接的遊戲客戶端通過兩個渠道進行通信：用於可靠交付的 TCP 連接和用於快速交付的 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 UDP 通道無法使用，亞馬遜GameLift會使用 TCP 作為後援傳送訊息。

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 可識別特定玩家活動或遊戲事件，而裝載則提供與操作代碼相關的其他資料。這兩個元素都是由開發人員定義。您的遊戲用戶端會根據其收到的訊息中的作業碼進行動作。

玩家群組

實時服務器提供管理玩家組的功能。根據預設，Amazon GameLift 會將連線到遊戲的所有玩家都放在「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分享遊戲資料。群組的一個可能用途是設定玩家團隊和管理團隊通訊。

具有 TLS 證書的實時服務器

使用實時服務器，服務器身份驗證和數據包加密內置到服務中。當您開啟 TLS 憑證產生時，會啟用這些安全性功能。當遊戲用戶端嘗試與即時伺服器連線時，伺服器會自動回應用戶端驗證的 TLS 憑證。亞馬遜使用 TLS 進行 TCP (WebSockets) 通訊，並針對 UDP 流量使用 DTLS 來GameLift處理加密。

自訂即時伺服器

實時服務器作為無狀態中繼服務器執行。實時服務器在連接到遊戲的遊戲客戶端之間轉發消息和遊戲數據包。但是，實時服務器不會評估消息，處理數據或執行任何遊戲邏輯。以這種方式使用，每個遊戲客戶端都會維護自己的遊戲狀態視圖，並通過轉送服務器向其他玩家提供更新。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。

您可以通過添加到實時腳本功能來自定義服務器。例如，使用遊戲邏輯，您可以使用伺服器授權的遊戲狀態檢視來建立可設定狀態的遊戲。

Amazon 為即時指令碼GameLift定義了一組伺服器端回呼。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要求加入群組。
- 確定何時從某個玩家或目標玩家發送消息，或進行其他響應處理。
- 當玩家離開群組或與伺服器中斷連線時，請通知所有玩家。
- 查看遊戲會話對象或消息對象的內容，並使用數據。

部署和更新實時服務器

實時服務器的一個主要優勢是能夠隨時更新腳本。當您更新腳本時，Amazon GameLift 會在幾分鐘內將新版本分發到所有託管資源。Amazon GameLift 部署新指令碼後，在該時間點之後建立的所有新遊戲工作階段都將使用新的指令碼版本。（現有的遊戲階段將繼續使用原始版本。）

開始將您的遊戲與即時伺服器整合：

- 為即時伺服器整合遊戲用戶端
- 創建實時腳本

為即時伺服器整合遊戲用戶端

本主題說明如何準備您的遊戲用戶端，以便能夠加入並參與 Amazon GameLift 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。

- 使您的遊戲客戶端能夠加入託管在實時服務器上的遊戲會話並交換消息。

查找或創建遊戲會話和玩家會話

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，使用該GameLift服務向 Amazon 服務發出直接請求。後端服務接著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 用GameLift戶端，然後將其設定為使用叢集和佇列中的託管資源。此開發AWS套件提供多種語言版本；請參閱 Amazon 開GameLift發套針對自訂用戶端服務件。
2. 將GameLift功能添加到您的後端服務。如需更詳細的指示，請參閱添加 Amazon GameLift 到您的遊戲客戶端和新增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智慧地放置新遊戲工作階段的能力，以及使用玩家延遲資料將遊戲延遲降至最低。GameLift您的後端服務至少必須能夠要求新的遊戲工作階段，並處理遊戲工作階段資料以作回應。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 Amazon GameLift 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。

連接到實時服務器上的遊戲

讓您的遊戲用戶端直接與即時伺服器上的代管遊戲工作階段連線，並與伺服器和其他玩家交換訊息。

1. 獲取實時客戶端 SDK，構建它，並將其添加到您的遊戲客戶端項目中。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，請參閱即時伺服器用戶端 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 實時服務器客戶端 API（C＃）參考：異步回調。

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

基本實時客戶端（C＃）

此範例說明了與即時用戶端 SDK (C#) 的基本遊戲用戶端整合。如圖所示，該示例初始化 Realtime 客戶端對象，設置事件處理程序並實現客戶端回調，連接到實時服務器，發送消息以及斷開連接。

```
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 GameLift Realtime client SDK
     *
     * 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 GameLift 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 GameLift
            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);
```

```
        }
    }
}
```

創建實時腳本

要在遊戲中使用實時服務器，您需要提供腳本（以某些JavaScript代碼的形式）來配置和可選地自定義實時服務器群。本主題涵蓋建立即時指令碼的關鍵步驟。指令碼準備就緒後，將其上傳到 Amazon GameLift 服務並使用它來建立叢集 (請參閱將實時服務器腳本上傳到亞馬遜 GameLift)。

要準備腳本以與實時服務器一起使用，請將以下功能添加到實時腳本中。

管理遊戲工作階段生命週期 (必要)

Realtime 腳本至少必須包含該`Init()`函數，該函數可準備實時服務器以啟動遊戲會話。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()`調用時，回調函數會傳遞一個實時會話對象，該對象包含實時服務器的接口。請參閱 實時服務器接口以取得此界面的詳細資訊。

為了正常結束遊戲會話，腳本還必須調用實時服務器的`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。

具有最基本配置-服務器進程初始化和終止的實時服務器-基本上充當無狀態中繼服務器。實時服務器在連接到遊戲的遊戲客戶端之間轉發消息和遊戲數據，但不會採取獨立的操作來處理數據或執行邏輯。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。

添加服務器端遊戲邏輯（可選）

您可以選擇將遊戲邏輯添加到實時腳本中。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 亞馬遜GameLift即時伺服器指令碼。

- 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 實時服務器的腳本回調。
- 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` 界面來剖析訊息內容 (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 與您車隊的其他AWS資源進行溝通。

- 允許遊戲邏輯存取執行所在執行執行個體的叢集資訊。如需詳細資訊，請參閱 取得 Amazon GameLift 執行個體的叢集資料。

實時服務器腳本示例

此示例說明了部署實時服務器以及一些自定義邏輯所需的基本腳本。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 GameLift
// 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
};
```

用於 Unity 的 Amazon GameLift 插件集成遊戲

本節中的主題描述了適用於 Unity 的 Amazon GameLift 外掛程式，以及如何使用它來準備您的多人遊戲專案以便在 Amazon 託管 GameLift。使用插件的引導式工作流程完全在您的 Unity 開發環境中工作，以完成在 Amazon 託管的基本要求 GameLift。

Amazon GameLift 是一項全受管服務，可讓遊戲開發人員管理和擴展工作階段型多人遊戲的專用遊戲伺服器。如需 Amazon GameLift 託管的詳細資訊，請參閱亞馬遜如何GameLift工作。

- Amazon GameLift 插件統一指南服務器 SDK 5.x，版本 2.0.0，適用於服務器 SDK 5.x 和支持 Amazon。 GameLift Anywhere
- Amazon GameLift 插件統一指南服務器 SDK 4.x，版本 1.0.0，適用於伺服器 SDK 4.x 或更早版本。此版本使用 Amazon 本 GameLift 地進行集成測試。

Amazon GameLift 插件統一指南服務器 SDK 5.x

Amazon GameLift 提供了準備您的多人遊戲服務器與 Amazon 一起工作的工具 GameLift。用於 Unity 的 Amazon GameLift 插件可以更輕鬆地 GameLift 將 Amazon 集成到您的統一遊戲項目中，測試與 Amazon 的集成 GameLift Anywhere，並部署用於雲託管的 Amazon GameLift 資源。

該插件使用AWS CloudFormation模板部署常見遊戲場景的託管解決方案。請依照提供的方式使用這些解決方案，或視需要自訂您的遊戲。

主題

關於插件

Unity 的外掛程式提供了簡化的入門體驗，用於與 Amazon 整合和託管您的 Unity 多人遊戲 GameLift。您可以利用插件功能和預先構建的組件來快速啟動並運行遊戲。

該插件將工具和功能添加到 Unity 編輯器中。使用引導式工作流程 GameLift 將 Amazon 整合到您的遊戲專案中、在本機進行測試，然後將遊戲伺服器部署到 Amazon GameLift 雲端託管。

使用插件的預構建託管解決方案來部署您的遊戲。將您的本機工作站設定為主機的 Amazon GameLift 無所不在叢集。對於雲託管，請從兩種常見的部署方案中進行選擇，以不同方式平衡玩家延遲、遊戲工作階段可用性和成本。一種情境包括簡單的 FlexMatch分房系統和規則集。使用這些案例來放置基本的生產就緒託管解決方案，然後根據需要進行優化和自訂。

該插件包括以下組件：

- Unity 編輯器的插件模塊。安裝插件後，一個新的主菜單項使您可以訪問 Amazon GameLift 功能。
- 具有用戶端功能的 Amazon GameLift 服務 API 的 C# 程式庫。
- 適用於 Amazon GameLift 服務器 SDK（版本 5.x）的 C＃ 庫。
- 遊戲內容範例，包括資產和場景，因此 GameLift 即使您沒有可用於建置的多人遊戲，也可以試用 Amazon。

- 解決方案組態 (以AWS CloudFormation範本形式提供)，此外掛程式會在將遊戲伺服器部署到雲端進行託管時使用。

插件工作流

下列步驟說明整合和部署遊戲專案與 Unity 的 Amazon GameLift 外掛程式的典型方法。您可以在 Unity 編輯器和您的遊戲程式碼中工作來完成這些步驟。

1. 建立連結至您AWS帳戶的使用者設定檔，並為具有使用 Amazon 權限的有效帳戶使用者提供存取登入資料 GameLift。
2. 將伺服器程式碼新增至您的遊戲專案，以便在執行中的遊戲伺服器與使用 Amazon GameLift 服務之間建立通訊。
3. 將用戶端程式碼新增至您的遊戲專案，讓遊戲用戶端傳送請求 GameLift 至 Amazon 以開始或加入遊戲工作階段，然後連線至遊戲伺服器。
4. 使用 Anywhere 工作流程將本機工作站設定為遊戲伺服器的 Anywhere 主機。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。
5. 使用 EC2 託管工作流程上傳您的整合式遊戲伺服器，並部署雲端託管解決方案。當您的遊戲伺服器準備就緒時，請在本機啟動遊戲用戶端，連線至遊戲工作階段並登入，然後進行遊戲。

在插件中工作時，您將創建和使用AWS資源，這些操作可能會向正在使用的AWS帳戶產生費用。如果您不熟悉AWS，可能會涵蓋AWS免費方案中的動作。

安裝團結的外掛程式

本節介紹如何將插件添加到 Unity 項目。安裝外掛程式後，當您在 Unity 編輯器中開啟專案時，就可以使用外掛程式功能。

開始之前

下面是你需要使用 Amazon GameLift 插件統一:

- 統一視窗 2022 LTS 或統一的 MacOS
- Amazon GameLift 插件統一下載. [下載網站] 下載包括兩個軟件包：
  - Amazon GameLift 獨立插件的統一
  - 用於統一的 Amazon GameLift C＃ 服務器 SDK
- Microsoft 視覺工作室 2019 或更新版本。

- 使用 C# 遊戲程式碼的多人遊戲專案。
- 第三方範圍的登錄。 UnityNuGet此工具可管理協力廠商 DLL。如需詳細資訊，請參閱 UnityNuGetGithub 儲存庫。

將外掛程式新增至您的遊戲專案

完成以下任務，在 Unity 編輯器和您的遊戲項目文件中工作。

步驟 1：新增 UnityNuGet 至您的遊戲專案

如果您沒有為您的遊戲專案 UnityNuGet 設定，請使用下列步驟使用 Unity 套件管理員來安裝工具。或者，您可以使用 NuGet CLI 手動下載 DLL。如需詳細資訊，請參閱適用於統一的 Amazon GameLift C# 伺服器開發套件README。

1. 在 Unity 編輯器中打開項目後，轉到主菜單並選擇編輯，項目設置。從選項中，選擇「Package 件管理員」區段，然後開啟「範圍登錄」群組。
2. 選擇 + 按鈕，並為 UnityNuGet 範圍登錄輸入下列值：

```
Name: Unity NuGet
URL: https://unitynuget-registry.azurewebsites.net
Scope(s): org.nuget
```

對於統一 2021 版本的用戶：

設置後 UnityNuGet，檢查 Unity 控制台中顯示的`Assembly Version Validation`錯誤。如果 NuGet 封裝中強式名稱組件的繫結重新導向未正確解析為 Unity 專案中的路徑，就會發生這些錯誤。若要解決此問題，請設定 Unity 的組件版本驗證：

1. 在 Unity 編輯器中，轉到主菜單，然後選擇編輯，項目設置，然後打開播放器部分。
2. 取消選取「組件版本驗證」選項。

第 2 步：添加插件和 C # 服務器 SDK 包

1. 解壓縮 Amazon GameLift 插件統一下載，其中包含兩個軟件包。
2. 在 Unity 編輯器中打開項目後，轉到主菜單並選擇窗口，Package 管理器。
3. 選擇「+」按鈕以新增套件。選擇選項從壓縮包中添加包裹。

4. 在 [選取磁碟上的套件] 中，找出 Unity 下載檔案的 Amazon GameLift C# 伺服器 SDK 外掛程式，然後選擇`com.amazonaws.gameliftserver.sdk-<version>.tgz`檔案。選擇「開啟」以安裝外掛程式。
5. 在 [選取磁碟上的套件] 中，找出 Unity 下載檔案的 Amazon GameLift 獨立外掛程式，然後選擇檔案`com.amazonaws.gamelift-<version>.tgz`。選擇「開啟」以安裝外掛程式。
6. 確認獨立外掛程式已新增至您的專案。返回統一編輯器窗口。檢查新的 Amazon 菜單按鈕的主 GameLift菜單。

步驟 3：導入示例遊戲（可選）

Unity 的外掛程式隨附一組範例遊戲資產 (包括場景)，您可以將其新增到遊戲專案中。匯入範例遊戲可讓您快速透過 Amazon 測試、建置和部署簡單的多人遊戲 GameLift。範例遊戲已與 Amazon GameLift SDK 完全整合，因此您可以略過整合任務並完成剩餘的工作流程任務。

使用範例遊戲時，您可以在幾分鐘內設定並加入本機託管的 Amazon GameLift Anywhere 叢集。您可以在一小時內將遊戲部署到 Amazon， GameLift並在一小時內加入即時雲端託管遊戲。

若要匯入範例遊戲：

1. 在 Unity 編輯器中打開您的遊戲項目後，轉到 Amazon GameLift 菜單並選擇示例遊戲，導入示例遊戲。
2. 導入文件後，再次轉到 Amazon GameLift 菜單，然後選擇示例遊戲，初始化設置。此步驟會設定您的專案以建置遊戲用戶端和伺服器。

安裝完成後，您會看到兩個新場景新增到您的遊戲專案中。您也會看到一些額外的專案資產，包括GameLiftClientSettings資產。

如需範例使用者介面和遊戲玩法的詳細資訊，請參閱範例遊戲讀我。

設定AWS使用者設定檔

安裝插件後，設置配置文件並將其鏈接到有效的AWS帳戶用戶。您可以維護多個設定檔，但一次只能啟用一個設定檔。每當您在插件中工作時，請選擇要使用的配置文件。

維護多個配置文件使您能夠在不同的託管方案之間切換。例如，您可以使用相同的AWS認證，但不同的AWS區域設定設定檔。或者，您可以使用不同的AWS帳戶或使用不同的用戶/權限集來設置配置文件。

> **Note**
>
> 如果您已在工作站上安裝 AWS CLI 並且已設定描述檔，Amazon GameLift 外掛程式可以偵測到它，並將其列為現有設定檔。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。

若要設定您的設定AWS檔

1. 在 Unity 編輯器主功能表中，選擇 Amazon， GameLift然後選取 [設定AWS帳戶設定檔]。這個動作會開啟外掛程式視窗。打開頁面AWS用戶配置文件。
2. 如果插件檢測到現有配置文件，則不會提示您創建一個配置文件。選取現有的設定檔，或選擇「新增其他描述檔」來建立新的描述檔。
3. 如果插件未檢測到現有配置文件，則會提示您創建一個配置文件。您可以使用新帳戶或現有AWS帳戶建立新的設定檔。

   > **Note**
   >
   > 您必須使用AWS管理主控台來建立新AWS帳戶，並使用適當的權限集建立或更新使用者。

   設定設定檔時，您需要下列資訊：

   - 一個 AWS 帳戶。如果您需要建立新AWS帳戶，請依照提示建立帳戶。如需詳細資訊，請參閱建立AWS帳戶。
   - 具有使用 Amazon GameLift 和其他必要AWS服務權限的AWS帳戶使用者。設置一個 AWS 帳戶如需使用 Amazon GameLift 許可設定 AWS Identity and Access Management (IAM) 使用者的指示，請參閱。
   - 您的AWS使用者的認證。此使用者還需要具有長期認證的程式設計存取。這些認證包含AWS存取金鑰 ID 和AWS秘密金鑰。如需詳細資訊，請參閱取得存取金鑰。
   - AWS 區域。這是您要創建用於託管AWS資源的地理位置。在開發過程中，我們建議使用靠近您實際位置的區域，以最大程度地減少延遲。請參閱支援的AWS地區清單。
4. 當您選取或建立設定檔時，請檢查設定檔的啟動程序狀態，並視需要採取動作。必須啟動所有設定檔才能使用 Amazon GameLift 外掛程式功能。

若要啟動您的設定檔：

啟動安裝會指定 Amazon S3 儲存貯體，以搭配選取的使用者設定檔使用。Amazon S3 是資料和物件儲存的核心AWS服務。用於存儲項目配置，構建工件和其他依賴項的存儲桶。值區不會在其他設定檔之間共用。

> **Note**
>
> 啟動安裝會建立新AWS資源並可能產生成本。

1. 在插件窗口中查看配置文件用AWS戶配置文件時，選擇要使用的配置文件。如果設定檔尚未啟動載入，則會顯示警告訊息。
2. 在「啟動您的設定檔」區段中，從下拉式清單中選取設定檔，然後檢查啟動程序狀態。如果狀態指示沒有存在存儲桶，請選擇按鈕 Bootstrap 配置文件。您可以將值區名稱設定為新值區名稱、輸入您有權存取的現有值區，或保留自動產生的名稱。
3. 等待引導狀態更改為「活動」。這可能需要幾分鐘的時間。當狀態為「活動」時，您可以使用配置文件來使用插件功能

使用 Amazon 設定您的遊戲以進行本機測試 GameLift Anywhere

在此工作流程中，您可以新增 Amazon GameLift 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。當您完成整合任務後，請使用外掛程式來建置您的遊戲用戶端和伺服器元件。

要啟動 Amazon GameLift 任何地方工作流程：

- 在 Unity 編輯器主菜單中，選擇 Amazon， GameLift然後選擇主機與任何地方。此操作將打開插件頁面，用於設置您的遊戲 @ Anywhere 艦隊。此頁面顯示整合、建置和啟動遊戲元件的五個步驟程序。

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與設定檔的AWS帳戶相關聯，並放置在設定檔的預設AWS區域中。設定檔使用者的權限決定您對AWS資源和動作的存取權限。

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有個人資料或想創建一個新的個人資料，請轉到 Amazon GameLift 菜單並選擇設置AWS帳戶個人資料。
2. 如果啟動程序狀態不是「活動」，請選擇 Bootstrap 配置文件，然後等待狀態更改為「活動」。

將您的遊戲與 Amazon 整合 GameLift

> **Note**
>
> 如果您導入了示例遊戲，則可以跳過此步驟。範例遊戲資產已經有必要的伺服器和用戶端程式碼。

對於工作流程中的這個步驟，您可以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 Amazon GameLift 服務通訊，才能接收啟動遊戲工作階段、提供遊戲工作階段連線資訊和報告狀態的提示。
- 遊戲用戶端必須能夠取得有關遊戲工作階段的資訊、加入或開始遊戲工作階段，以及取得連線資訊才能加入遊戲。

整合伺服器程式碼

如果您將自己的遊戲專案與自訂場景搭配使用，請使用提供的範例程式碼將所需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，開啟資`Assets/Scripts/Server`料夾。如果它不存在，請創建它。
2. 轉到 GitHub 回購 aws/ amazon-gamelift-plugin-unity 並打開路徑`Samples~/SampleGame/Assets/Scripts/Server`。
3. 找到 GameLiftServer .c. 檔案並將其複製到遊戲專案的伺服器資料夾中。當您建置伺服器可執行檔時，請使用此檔案做為建置目標。

範例程式碼包含下列使用 Amazon GameLift C# 伺服器 SDK (版本 5) 的最低必要元素：

- 初始化 Amazon GameLift API 用戶端。Amazon 無 GameLift 處不在叢集需要具有伺服器參數的`InitSDK()`呼叫。這些設置會自動設置為在插件中使用。
- 實作必要的回呼函數，以回應 Amazon GameLift 服務的請求`OnStartGameSession`，包括`OnProcessTerminate`、和`onHealthCheck`。

- 在伺 GameLift 服器處理序準備好主持遊戲工作階段時，`ProcessReady()`使用指定的連接埠呼叫以通知 Amazon 服務。

如果您想要自訂範例伺服器程式碼，請參閱下列資源：

- 添加亞馬遜GameLift到您的遊戲服務器
- 用於 C# 和統一的亞馬遜GameLift服務器 SDK 5.x 參考

整合用戶端程式碼

如果您將自己的遊戲專案與自訂場景搭配使用，則需要將基本功能整合到您的遊戲用戶端中。您還需要添加 UI 元素，以便玩家可以登錄並加入遊戲會話。使用 Amazon GameLift 服務 API (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。

建立使用 Anywhere 叢集進行本機測試的用戶端時，您可以新增直接呼叫至 Amazon GameLift 服務。當您開發用於雲端託管的遊戲時，或者如果您打算使用 Anywhere 叢集進行生產託管，則需要建立用戶端後端服務來處理遊戲用戶端與 Amazon 服務之間的所有通訊。 GameLift

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

- 將客戶端與 GitHub回購 aws/ amazon-gamelift-plugin-unity 中的 GameLiftCoreApi 類集成。此類別提供玩家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 軟體庫 aws/ amazon-gamelift-plugin-unity、`Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`.
- 請按照將 Amazon 添加 GameLift 到您的 Unity 遊戲客戶端中的說明

對於連接到 Anywhere 機隊的遊戲客戶端，您的遊戲客戶端需要以下信息。該插件會自動更新您的遊戲項目以使用您在插件中創建的資源。

- FleetId -您的任何地方機隊的唯一標識符。
- FleetLocation -您的任何地方機隊的自定義位置。
- AwsRegion -您的任何AWS地方機隊託管的地區。這是您在使用者設定檔中設定的區域。
- ProfileName -本機電腦上允許存取 AWS SDK 的AWS認證設定檔 GameLift。遊戲用戶端會使用這些登入資料來驗證對 Amazon GameLift 服務的請求。

> **Note**
>
> 憑證設定檔由外掛程式產生並儲存在本機電腦上。因此，您必須在本機電腦 (或具有相同設定檔的電腦上) 執行用戶端。

Connect 到任何地方的機隊

在此步驟中，您可以指定要使用的任何地方叢集。Anywhere 叢集定義了一系列運算資源，這些資源可以位於任何地方，以供遊戲伺服器託管使用。

- 如果您目前使用的AWS帳戶具有現有的 Anywhere 叢集，請開啟 [叢集名稱] 下拉式欄位並選擇叢集。此下拉式清單只會顯示目前作用中使用者設定檔的 [AWS區域] 中的 [任何地方] 叢集。
- 如果沒有現有的車隊，或者您想要建立新的叢集，請選擇 [建立新的任何地方] 叢集並提供叢集名稱。

在您為專案選擇任何地方叢集之後，Amazon 會 GameLift 驗證叢集狀態為使用中廣告顯示叢集 ID。您可以在 Unity 編輯器的輸出日誌中跟踪此請求的進度。

註冊運算

在此步驟中，您將本機工作站註冊為新的 Anywhere 叢集中的計算資源。

1. 輸入本機電腦的運算名稱。如果您在叢集中新增多個計算，名稱必須是唯一的。
2. 選擇 [註冊運算]。您可以在虛幻編輯器的輸出日誌中追蹤此請求的進度。

該插件將 IP 地址設置為本地主機（127.0.0.1）註冊您的本地工作站。此設定假設您將在同一台機器上執行遊戲用戶端和伺服器。

為了回應此動作，Amazon GameLift 會驗證其是否可以連線到運算，並傳回有關新註冊運算的資訊。

啟動遊戲

在此步驟中，您將構建遊戲組件並啟動它們以進行遊戲。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。該插件使用此資產來存儲您的遊戲客戶端需要連接到 Amazon GameLift 服務的某些信息。

a. 如果您沒有匯入並初始化範例遊戲，請建立新GameLiftClientSettings資產。在 Unity 編輯器主功能表中，選擇 [資產]、[建立] GameLift、[用戶端設定]。如果您 GameLiftClientSettings 在專案中建立了多個複本，外掛程式會自動偵測到這一點，並通知您外掛程式將更新哪個資產。

b. 在 [啟動遊戲] 中，選擇 [設定用戶端：隨處套用設定] 此動作會更新您的遊戲用戶端設定，以使用您剛剛設定的 Anywhere 叢集。

2. 建置並執行您的遊戲用戶端。

a. 使用標準 Unity 構建過程構建客戶端可執行文件。在 [檔案] 的 [建置設定] 中，將平台切換至視窗、Mac、Linux。如果您導入了示例遊戲並初始化了設置，則構建列表和構建目標將自動更新。

b. 啟動新建立的遊戲用戶端可執行檔的一或多個執行個體。

3. 在您的任何地方機隊中啟動遊戲伺服器。選擇伺服器：在編輯器中啟動伺服器。此工作會啟動即時伺服器，只要 Unity 編輯器保持開啟狀態，您的用戶端就可以連線到該伺服器。

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您如何向用戶端新增功能。

如果您使用的是範例遊戲用戶端，它具有下列特性：

- 玩家登入元件。連線至 Anywhere 叢集上的遊戲伺服器時，沒有玩家驗證。您可以輸入任何值以加入遊戲階段。
- 一個簡單的加入遊戲界面。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，用戶端會要求新的遊戲工作階段。如果有可用的遊戲工作階段，用戶端會要求加入可用的遊戲工作階段。使用多個並行用戶端測試您的遊戲時，第一個用戶端會啟動遊戲工作階段，其餘的用戶端會自動加入現有的遊戲工作階段。
- 具有四個玩家插槽的遊戲會話。您最多可以同時啟動四個遊戲用戶端執行個體，這些執行個體將會加入相同的遊戲工作階段。

從伺服器執行檔啟動 (選用)

您可以建置並啟動遊戲伺服器可執行檔，以便在 Anywhere 叢集上進行測試。

1. 使用標準 Unity 構建過程構建服務器可執行文件。在 [檔案] 的 [建置設定] 中，將平台切換至專用伺服器並建置。

2. get-compute-auth-token使用您的 Anywhere 叢集 ID 和AWS區域呼叫 AWS CLI 命令，以取得短期驗證權杖。當您建立叢集時，叢集 ID 會顯示在「Connect 至任何地方」叢集中。當您選取作用中的設定檔時，「設定您的設定檔」中會顯示「AWS地區」。

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

3. 從命令行啟動新構建的遊戲服務器可執行文件，並傳入有效的身份驗證令牌。

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

使用受管 EC2 叢集將您的遊戲部署到雲端託管

在此工作流程中，您可以使用外掛程式準備遊戲，以便在 Amazon 管理的雲端運算資源上託管 GameLift。您可以為 Amazon GameLift 功能添加客戶端和服務器遊戲代碼，然後將服務器構建上傳到 Amazon GameLift 服務進行託管。完成此工作流程後，您將擁有在雲端中執行的遊戲伺服器，以及可連線到這些伺服器的有效遊戲用戶端。

若要啟動 Amazon GameLift 託管 EC2 工作流程：

- 在 Unity 編輯器主功能表中，選擇 Amazon， GameLift然後選取主機與受管 EC2。此工作流程提供整合、建置、部署和啟動遊戲元件的六個步驟程序。

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與設定檔的AWS帳戶相關聯，並放置在設定檔的預設AWS區域中。設定檔使用者的權限決定您對AWS資源和動作的存取權限。

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有個人資料或想創建一個新的個人資料，請轉到 Amazon GameLift 菜單並選擇設置AWS帳戶個人資料。
2. 如果啟動程序狀態不是「活動」，請選擇 Bootstrap 配置文件，然後等待狀態更改為「活動」。

將您的遊戲與 Amazon 整合 GameLift

對於這項工作，您可以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 Amazon GameLift 服務通訊，才能接收啟動遊戲工作階段、提供遊戲工作階段連線資訊和報告狀態的提示。
- 遊戲用戶端必須能夠取得有關遊戲工作階段的資訊、加入或開始遊戲工作階段，以及取得連線資訊才能加入遊戲。

> **Note**
>
> 如果您導入了示例遊戲，則可以跳過此步驟。範例遊戲資產已經有必要的伺服器和用戶端程式碼。

整合伺服器程式碼

在自訂場景中使用您自己的遊戲專案時，請使用提供的範例程式碼將所需的伺服器程式碼新增至您的遊戲專案。如果您整合了遊戲專案以便與 Anywhere 叢集進行測試，則表示您已完成此步驟中的指示。

1. 在您的遊戲專案檔案中，開啟資`Assets/Scripts/Server`料夾。如果它不存在，請創建它。
2. 轉到 GitHub 回購 aws/ amazon-gamelift-plugin-unity 並打開路徑`Samples~/SampleGame/Assets/Scripts/Server`。
3. 找到檔案`GameLiftServer.cs`並將其複製到遊戲專案的`Server`資料夾中。當您建置伺服器可執行檔時，請使用此檔案做為建置目標。

範例程式碼包含下列使用 Amazon GameLift C# 伺服器 SDK (版本 5) 的最低必要元素：

- 初始化 Amazon GameLift API 用戶端。Amazon 無 GameLift 處不在叢集需要具有伺服器參數的 InitSDK () 呼叫。這些設置會自動設置為在插件中使用。
- 實作必要的回呼函數，以回應 Amazon GameLift 服務的請求`OnStartGameSession`，包括`OnProcessTerminate`、和`onHealthCheck`。
- 在伺 GameLift 服器處理序準備好主持遊戲工作階段時，`ProcessReady()`使用指定的連接埠呼叫以通知 Amazon 服務。

如果您想要自訂範例伺服器程式碼，請參閱下列資源：

- 添加亞馬遜GameLift到您的遊戲服務器
- 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考

整合用戶端程式碼

對於連線到雲端遊戲伺服器的遊戲用戶端，最佳做法是使用用戶端後端服務撥打 Amazon GameLift 服務，而不是直接從遊戲用戶端撥打電話。

在託管 EC2 叢集上託管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 表格，用來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 閘道做為代理伺服器。
- Amazon Cognito 使用者集區，可產生唯一的玩家 ID 並驗證玩家連線。

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

- 在 AWS Cognito 使用者集區中建立播放程式使用者並進行驗證。
- 加入遊戲工作階段並接收連線資訊。
- 使用配對功能加入遊戲。

使用以下資源作為指南。

- 將客戶端與 GitHub 回購 aws/ amazon-gamelift-plugin-unity 中的 GameLiftCoreApi類集成。此類別提供玩家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 軟體庫 aws/ amazon-gamelift-plugin-unity、`Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`.
- 將 Amazon 添加 GameLift 到您的 Unity 遊戲客戶端。

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。您可以使用任何情境來進行多個遊戲部署。

- 單一區域叢集：將您的遊戲伺服器部署到作用中設定檔預設AWS區域中的單一主機資源叢集。這個案例是測試伺服器整合與伺服器組建組態AWS的良好起點。它會部署下列資源：
  - AWS已安裝並執行遊戲伺服器組建的叢集 (隨選)。
  - Amazon Cognito 使用者集區和用戶端可讓玩家驗證並開始遊戲。
  - 將使用者集區與 API 連結的 API 閘道授權者。
  - WebACL 用於限制對 API 網關的過多播放器調用。

  - API 網關 + Lambda 功能，供玩家請求遊戲位置。`CreateGameSession()`如果沒有可用此函數調用。
  - API 閘道 + Lambda 函數可讓玩家取得遊戲請求的連線資訊。
- FlexMatch 艦隊：將您的遊戲伺服器部署到一組艦隊，並設定具有建立玩 FlexMatch 家對戰規則的分房系統。此案例使用低成本的 Spot 主機，搭配多叢集、多地點結構，以提供持久的可用性。當您準備好開始為您的主機解決方案設計分房系統元件時，此方法非常有用。在這個案例中，您將建立此解決方案的基本資源，您可以在稍後視需要自訂這些資源。它會部署下列資源：
  - FlexMatch 配對配對配置和配對規則設置為接受玩家請求和形式匹配。
  - 您的遊戲伺服器組建安裝並在多個位置執行的三個AWS艦隊。包括兩個 Spot 叢集和一個隨需叢集做為備份。
  - AWS遊戲會話放置隊列，通過找到可能的最佳託管資源（基於可行性，成本，玩家延遲等）並啟動遊戲會話來滿足提議的匹配請求。
  - Amazon Cognito 使用者集區和用戶端可讓玩家驗證並開始遊戲。
  - 將使用者集區與 API 連結的 API 閘道授權者。
  - WebACL 用於限制對 API 網關的過多播放器調用。
  - API 網關 + Lambda 功能，供玩家請求遊戲位置。此函數會呼叫`StartMatchmaking()`。
  - API 閘道 + Lambda 函數可讓玩家取得遊戲請求的連線資訊。
  - Amazon DynamoDB 資料表可用來儲存玩家的配對票證和遊戲工作階段資訊。
  - SNS 主題 + Lambda 函數來處理 GameSessionQueue 事件。

設定遊戲參數

在此步驟中，您將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。此名稱在外掛程式中使用。
- 叢集名稱：為您的受管 EC2 叢集提供有意義的名稱。Amazon 在AWS主控台中列出資源時，GameLift 會使用此名稱 (以及叢集 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 並用於部署的伺服器組建副本。
- 啟動參數：輸入在受管 EC2 叢集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：提供包含您伺服器組建的本機資料夾路徑。
- 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，將遊戲部署到雲端主機解決方案。此程序可能需要長達 40 分鐘的時間，同時AWS驗證您的伺服器組建、佈建主機資源、安裝遊戲伺服器、啟動伺服器程序，以及讓它們準備好主持遊戲工作階段。

若要開始部署，請選擇部署 CloudFormation。您可以在此處跟踪遊戲託管的狀態。如需詳細資訊，您可以登入AWS管理主控台以查看事件通知。AWS請務必使用與插件中活躍用戶配置文件相同的帳戶，用戶和AWS區域登錄。

部署完成後，您將遊戲伺服器安裝在 AWS EC2 執行個體上。至少有一個伺服器處理序正在執行，並準備好啟動遊戲工作階段。

啟動遊戲客戶端

成功部署叢集後，您現在可以執行遊戲伺服器，並可用於主持遊戲工作階段。您現在可以構建客戶端，啟動它，連接以加入遊戲會話。

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。該插件使用此資產來存儲您的遊戲客戶端需要連接到 Amazon GameLift 服務的某些信息。

   a. 如果您沒有匯入並初始化範例遊戲，請建立新`GameLiftClientSettings`資產。在 Unity 編輯器主功能表中，選擇 [資產]、[建立] GameLift、[用戶端設定]。如果您 GameLiftClientSettings 在專案中建立了多個複本，外掛程式會自動偵測到這一點，並通知您外掛程式將更新哪個資產。

   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 叢集。

2. 建立您的遊戲用戶端。使用標準 Unity 構建過程構建客戶端可執行文件。在 [檔案] 的 [建置設定] 中，將平台切換至視窗、Mac、Linux。如果您導入了示例遊戲並初始化了設置，則構建列表和構建目標將自動更新。
3. 啟動新建立的遊戲用戶端可執行檔。若要開始遊玩遊戲，請啟動兩到四個用戶端執行個體，並在每個執行個體中使用 UI 加入遊戲工作階段。

如果您使用的是範例遊戲用戶端，它具有下列特性：

- 玩家登入元件。連線至 Anywhere 叢集上的遊戲伺服器時，沒有玩家驗證。您可以輸入任何值以加入遊戲階段。

- 一個簡單的加入遊戲界面。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，用戶端會要求新的遊戲工作階段。如果有可用的遊戲工作階段，用戶端會要求加入可用的遊戲工作階段。使用多個並行用戶端測試您的遊戲時，第一個用戶端會啟動遊戲工作階段，其餘的用戶端會自動加入現有的遊戲工作階段。
- 具有四個玩家插槽的遊戲會話。您最多可以同時啟動四個遊戲用戶端執行個體，這些執行個體將會加入相同的遊戲工作階段。

Amazon GameLift 插件統一指南服務器 SDK 4.x

> **Note**
>
> 本主題提供適用於 Unity 的舊版 Amazon GameLift 外掛程式的資訊。版本 1.0.0（於 2021 年發布）使用 Amazon GameLift 服務器 SDK 4.x 或更早版本。有關使用服務器 SDK 5.x 並支持 Amazon 的最新版本的插件的文檔 GameLift Anywhere，請參閱Amazon GameLift 插件統一指南服務器 SDK 5.x。

Amazon GameLift 提供了準備您的多人遊戲服務器在 Amazon 上運行的工具 GameLift。用於統一的 Amazon GameLift 插件可以更輕鬆地 GameLift 將 Amazon 集成到您的統一遊戲項目中，並部署 Amazon GameLift 資源進行雲託管。使用 Unity 的外掛程式存取 Amazon GameLift API 並部署常見遊戲案例的AWS CloudFormation範本。

設置好外掛程式後，您可以在上試用 Amazon GameLift Unity 範例 GitHub。

主題

- 將 Amazon GameLift 與 Unity 遊戲服務器項目集成
- 將 Amazon GameLift 與 Unity 遊戲客戶端項目集成
- 安裝和設置插件
- 在本機測試您的遊戲
- 部署案例
- GameLift 在統一中與 Amazon 整合遊戲
- 匯入並執行範例遊戲

將 Amazon GameLift 與 Unity 遊戲服務器項目集成

> **Note**
>
> 本主題提供適用於 Unity 的舊版 Amazon GameLift 外掛程式的資訊。版本 1.0.0（於 2021 年發布）使用 Amazon GameLift 服務器 SDK 4.x 或更早版本。有關使用服務器 SDK 5.x 並支持 Amazon 的最新版本的插件的文檔 GameLift Anywhere，請參閱Amazon GameLift 插件統一指南服務器 SDK 5.x。

本主題可協助您準備自訂遊戲伺服器，以便在 Amazon 上託管 GameLift。遊戲伺服器必須能夠通 GameLift 知 Amazon 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，請參閱添加亞馬遜GameLift到您的遊戲服務器。

必要條件

在整合您的遊戲伺服器之前，請先完成下列工作：

- 為 Amazon 設置 IAM 服務角色 GameLift
- 安裝團結的外掛程式

設定新的伺服器處理序

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

設定與 Amazon 的通訊， GameLift 並報告伺服器程序已準備好主持遊戲工作階段。

1. 通過調用初始化服務器 SDK `InitSDK()`。
2. 要準備服務器接受遊戲會話，請`ProcessReady()`使用連接端口和遊戲會話位置詳細信息進行調用。包括 Amazon GameLift 服務調用的回調函數的名稱，例如`OnGameSession()`，，`OnGameSessionUpdate()``OnProcessTerminate()`，`OnHealthCheck(`GameLift 可能需要幾分鐘才能提供回調。
3. Amazon 將服務器進程的狀態 GameLift 更新為`ACTIVE`。

4. Amazon onHealthCheck 定期 GameLift 致電。

下面的代碼示例演示了如何設置一個簡單的服務器進程與 Amazon GameLift。

```
//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;
}
```

開始遊戲工作階段

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

遊戲初始化完成後，您可以開始遊戲工作階段。

1. 實作回呼函式 onStartGameSession。Amazon GameLift 調用此方法以在服務器進程上啟動新的遊戲會話並接收玩家連接。
2. 若要啟動遊戲工作階段，請撥打電話ActivateGameSession()。如需 SDK 的詳細資訊，請參閱亞馬遜GameLift服務器 SDK（C＃）參考：操作。

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

```
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**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

GameLift 當遊戲會話結束時通知 Amazon。最佳做法是在遊戲工作階段完成後關閉伺服器程序，以回收和重新整理主機資源。

1. 設置一個名為接收onProcessTerminate來自 Amazon 的請求 GameLift 和調用的函數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);
```

創建服務器構建並上傳到 Amazon GameLift

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

將遊戲伺服器與 Amazon 整合後 GameLift，請將建置檔案上傳到叢集，以便 Amazon GameLift 可以部署它進行遊戲託管。如需如何將伺服器上傳到 Amazon 的詳細資訊 GameLift，請參閱將自訂伺服器組建上傳到亞馬遜 GameLift。

將 Amazon GameLift 與 Unity 遊戲客戶端項目集成

> **Note**
>
> 本主題提供適用於 Unity 的舊版 Amazon GameLift 外掛程式的資訊。版本 1.0.0（於 2021 年發布）使用 Amazon GameLift 服務器 SDK 4.x 或更早版本。有關使用服務器 SDK 5.x 並支持 Amazon 的最新版本的插件的文檔 GameLift Anywhere，請參閱Amazon GameLift 插件統一指南服務器 SDK 5.x。

本主題可協助您設定遊戲用戶端，透過後端服務連線到 Amazon GameLift 託管的遊戲工作階段。使用 Amazon GameLift API 啟動配對、請求遊戲工作階段配置等等。

將代碼添加到後端服務項目以允許與 Amazon GameLift 服務進行通信。後端服務會處理所有遊戲用戶端與 GameLift 服務的通訊。如需後端服務的詳細資訊，請參閱設計您的遊戲用戶端服務。

後端伺服器會處理下列遊戲用戶端工作：

- 為您的玩家自定義身份驗證。
- 從 Amazon GameLift 服務要求有關使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。
- 將玩家從現有的遊戲工作階段中移除。

主題

- 必要條件
- 初始化遊戲用戶端
- 在特定艦隊上建立遊戲工作階段
- 將玩家加入遊戲工作階段
- 從遊戲工作階段移除玩家

必要條件

在設定與 Amazon GameLift 用戶端的遊戲伺服器通訊之前，請先完成以下任務：

- 設置一個 AWS 帳戶
- 安裝團結的外掛程式
- 將 Amazon GameLift 與 Unity 遊戲服務器項目集成
- 建立亞馬遜GameLift車隊

初始化遊戲用戶端

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

添加代碼以初始化遊戲客戶端。在啟動時運行此代碼，對於其他 Amazon GameLift 功能來說是必需的。

1. 初始化AmazonGameLiftClient。AmazonGameLiftClient使用預設用戶端組態或自訂組態來呼叫。如需如何設定用戶端的詳細資訊，請參閱在後端服務 GameLift 上設置 Amazon。
2. 為每個玩家生成一個獨特的玩家 ID，以連接到遊戲會話。如需更多資訊，請參閱 產生玩家 ID。

   下列範例顯示如何設定 Amazon 用 GameLift 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 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 GameLift Client with default client configuration.
        aglc = new AmazonGameLiftClient(credentials, config);

    }
}
```

在特定艦隊上建立遊戲工作階段

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 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 建立新的遊戲工作階段並傳回後GameSession，您可以將玩家新增至該遊戲階段。

- 提出新遊戲工作階段的請求。
  - 如果您的遊戲使用艦隊，請使CreateGameSession()用艦隊或別名 ID、工作階段名稱以及遊戲的同時玩家數目上限來呼叫。
  - 如果您的遊戲使用佇列，請撥打電話StartGameSessionPlacement()。

下列範例說明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;
}
```

將玩家加入遊戲工作階段

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

在 Amazon GameLift 建立新的遊戲工作階段並傳回GameSession物件後，您可以將玩家新增至其中。

1. 透過建立新的玩家工作階段，在遊戲工作階段中預留一個玩家角位。使用CreatePlayerSession或CreatePlayerSessions與每個玩家的遊戲會話 ID 和唯一 ID 一起使用。
2. Connect 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處理序的直接連線：
   a. 使用指定的連接埠，以及伺服器處理序的 DNS 名稱或 IP 位址。
   b. 使用您艦隊的 DNS 名稱和連接埠。如果您的叢集已啟用 TLS 憑證產生功能，則需要使用 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。

以下範例說明如何在遊戲工作階段中保留玩家位置。

```
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;
}
```

下面的代碼說明了如何將玩家與遊戲會話聯繫起來。

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

從遊戲工作階段移除玩家

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

您可以在玩家離開遊戲時將其從遊戲工作階段中移除。

1. 通知 Amazon GameLift 服務玩家已與服務器進程斷開連接。`RemovePlayerSession`使用玩家的工作階段 ID 呼叫。
2. 驗證`RemovePlayerSession`返回`Success`。然後，Amazon 將播放器插槽 GameLift 更改為可用，Amazon GameLift 可以將其分配給新玩家。

下列範例說明如何移除玩家工作階段。

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

安裝和設置插件

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

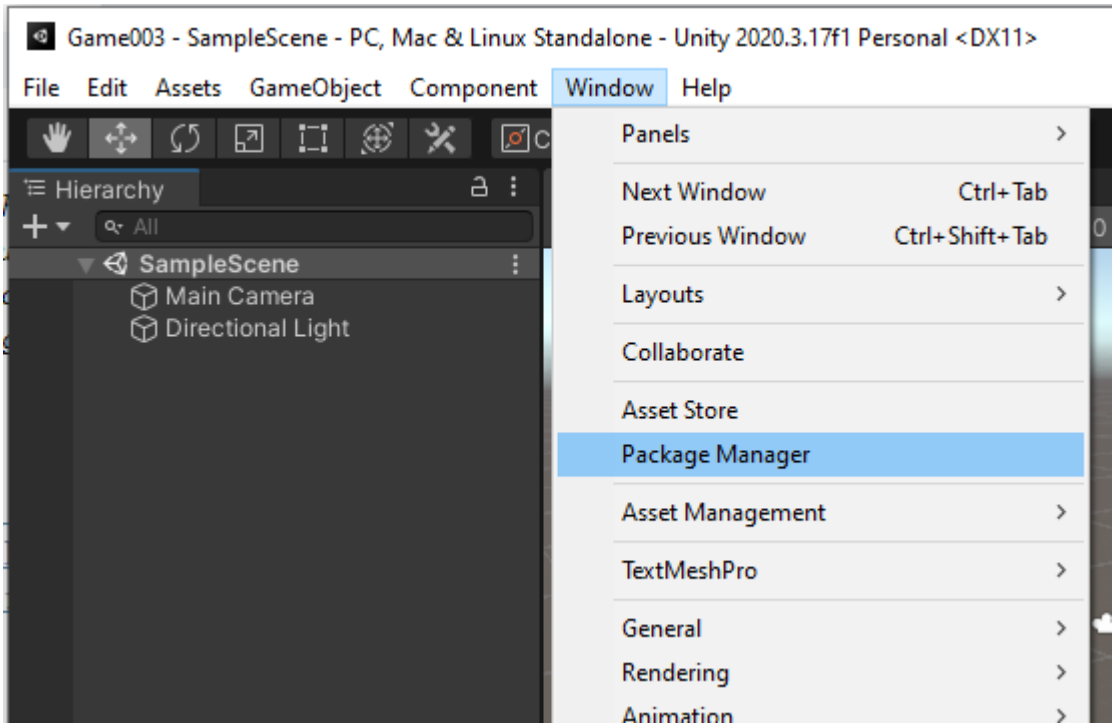
本節介紹如何下載，安裝和設置 Unity 的 Amazon GameLift 插件，版本 1.0.0。

必要條件

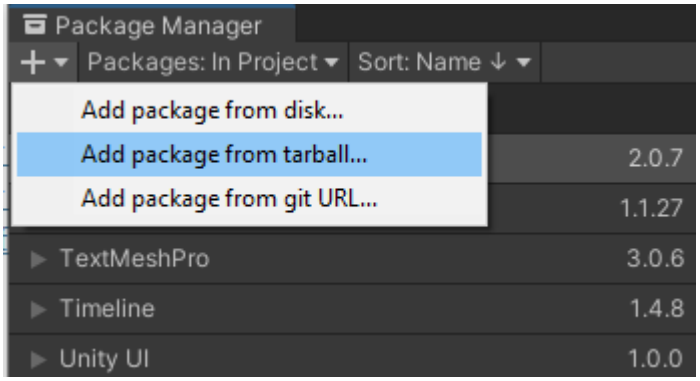
- 統一視窗 2019.4 LTS, 視窗 2020.3 LTS, 或統一的 MacOS
- 目前版本的 Java
- 目前版本的 .NET 4.x

若要下載並安裝 Unity 的外掛程式

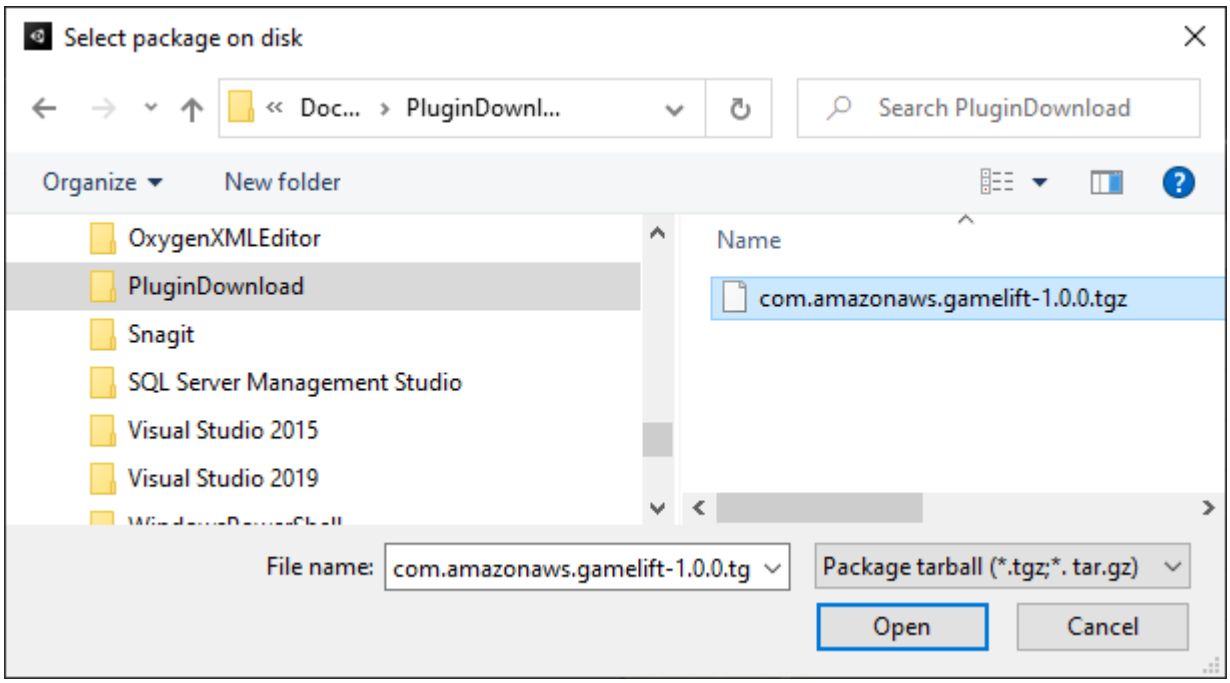
1. 下載 Amazon GameLift 插件統一. 您可以在用於 Unity 存儲庫頁面的 Amazon GameLift 插件上找到最新版本。在最新版本下，選擇 [資產]，然後下載`com.amazonaws.gamelift-`*`version`*`.tgz`檔案。
2. 啟動 Unity 並選擇一個項目。
3. 在頂部導航欄的窗口下選擇 P ackage 管理器：



4. 在軟件包管理器選項卡下選擇 +，然後選擇從 tarball 添加包... :



5. 在「選取磁碟上的套件」視窗中，瀏覽至`com.amazonaws.gamelift`資料夾，選擇檔案`com.amazonaws.gamelift-`*`version`*`.tgz` ，然後選擇「開啟」 :



6. 統一加載插件後，Amazon GameLift 顯示為統一菜單中的一個新項目。安裝和重新編譯指令碼可能需要幾分鐘的時間。Amazon GameLift 外掛程式設定索引標籤會自動開啟。

7. 在 [開發套件] 窗格中，選擇 [使用 .NET 4.x]。

   設定完成後，狀態會從 [未設定] 變更為 [已設定]。

在本機測試您的遊戲

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

使用 Amazon GameLift 本地在您的本地設備 GameLift 上運行 Amazon。您可以在幾秒鐘內使用 Amazon GameLift 本機驗證程式碼變更，而無需網路連線。

設定本機測試

1. 在 Unity 插件窗口中，選擇測試選項卡。
2. 在「測試」窗格中，選擇「下載 Amazon GameLift 本機」。Unity 的插件打開一個瀏覽器窗口，並將`GameLift_06_03_2021.zip`文件下載到您的下載文件夾。

   下載包括 C# 伺服器 SDK、.NET 原始程式碼檔案，以及與統一相容的 .NET 元件。
3. 解壓縮所下載的 `GameLift_06_03_2021.zip` 檔案。
4. 在 Amazon GameLift 外掛程式設定視窗中，選擇 Amazon GameLift 本機路徑，導覽至解壓縮的資料夾，選擇檔案`GameLiftLocal.jar`，然後選擇開啟。

   設定後，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。

   安裝並設定 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 標籤的外掛程式中，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。選擇... 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 「總帳本機」連接埠。
5. 選擇部署和運行以部署和運行服務器。
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**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

案例會使用AWS CloudFormation範本來建立您需要的資源，以便為您的遊戲部署雲端主機解決方案。本節說明 Amazon GameLift 提供的案例以及如何使用這些案例。

必要條件

若要部署案例，您需要 Amazon GameLift 服務的 IAM 角色。如需如何為 Amazon 建立角色的相關資訊 GameLift，請參閱設置一個 AWS 帳戶。

每個案例都需要下列資源的權限：

- Amazon GameLift
- Amazon S3
- AWS CloudFormation
- API 閘道
- AWS Lambda
- AWS WAFV2
- Amazon Cognito

案例

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

適用於 Unity 的 Amazon GameLift 外掛程式包括下列案例：

僅驗證

這個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驗 範本會在您的帳戶中建立下列資源：

- 用於儲存玩家身份驗證資訊的 Amazon Cognito 使用者集區。
- Amazon API Gateway REST 端點支援的AWS Lambda處理常式，可啟動遊戲並檢視遊戲連線資訊。

單一區域艦隊

此案例會使用單一 Amazon GameLift 叢集建立遊戲後端服務。它創建了以下資源：

- 一個 Amazon Cognito 使用者集區，讓玩家進行身份驗證和開始遊戲。
- 一個AWS Lambda處理程序，用於在艦隊中搜索現有的遊戲會話，並具有打開的玩家位置。如果找不到開啟的老虎機，就會建立新的遊戲工作階段。

具有佇列和自訂分房系統的多區域車隊

這種情況是使用 Amazon GameLift 佇列和自訂分房系統將等候池中最古老的玩家組合在一起，形成比對。它創建了以下資源：

- Amazon 向其 GameLift 發布消息的 Amazon 簡單通知服務主題。如需 SNS 主題與通知的詳細資訊，請參閱設定遊戲工作階段位置的事件通知。
- 由訊息叫用的 Lambda 函數，可傳達位置和遊戲連線詳細資料。
- 用於存放位置和遊戲連線詳細資訊的 Amazon DynamoDB 表格。 `GetGameConnection`從此表格讀取呼叫，並將連線資訊傳回至遊戲用戶端。

使用隊列和自定義分房系統來發現艦隊

這個案例會使用 Amazon GameLift 佇列和自訂分房系統來形成比對，並設定三個叢集。它創建了以下資源：

- 兩個 Spot 叢集包含不同的執行個體類型，可為 Spot 無法使用提供耐用性。
- 一種隨需叢集，可做為其他 Spot 叢集的備份。如需設計艦隊的詳細資訊，請參閱亞馬遜GameLift車隊設計指南。
- Amazon GameLift 佇列可讓伺服器保持高可用性並降低成本。如需佇列的詳細資訊和最佳作法，請參閱設計遊戲工作階段佇列。

FlexMatch

這個案例會使用 FlexMatch託管配對服務來配對遊戲玩家。有關更多信息 FlexMatch，請參閱什麼是 Amazon GameLift FlexMatch。此案例會建立下列資源：

- Lambda 函數，用於在收到`StartGame`請求後建立配對票證。
- 一個單獨的 Lambda 函數，用於偵聽 FlexMatch 匹配事件。

為了避免不必要的費用AWS 帳戶，請在您完成使用後移除每個案例所建立的資源。刪除相應的AWS CloudFormation堆疊。

更新AWS認證

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

適用於 Unity 的 Amazon GameLift 外掛程式需要安全登入資料才能部署案例。您可以建立新認證或使用現有的認證。

如需有關設定認證的詳細資訊，請參閱瞭解並取得AWS認證。

若要更新AWS認證

1. 在 Unity 中，在 Unity 索引標籤的外掛程式中，選擇部署索引標籤。
2. 在 [部署] 窗格中，選擇 [AWS認證]。
3. 您可以建立新的AWS認證或選擇現有的認證。
   - 若要建立認證，請選擇 [建立新的認證設定檔]，然後指定 [新設定檔名稱]、[AWS存取金鑰 ID]、[AWS秘密金鑰] 和AWS 區域。
   - 若要選擇現有的身分證明，請選擇 [選擇現有的認證設定檔]，然後選取設定檔名稱 AWS 區域
4. 在「更新AWS身份證明」視窗中，選擇「更新認證設定檔

更新帳戶引導

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

啟動程序位置是部署期間使用的 Amazon S3 儲存貯體。它用於存儲遊戲服務器資產和其他依賴項。AWS 區域您為值區選擇的區域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 簡單儲存服務儲存貯體。

更新帳戶啟動程序位置

1. 在 Unity 中，在 Unity 索引標籤的外掛程式中，選擇部署索引標籤。
2. 在 [部署] 窗格中，選擇 [更新帳戶啟動]。
3. 在帳戶啟動安裝視窗中，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 簡易儲存服務儲存貯體，然後選擇政策。該政策指定 Amazon S3 儲存貯體的到期時間。選擇「建立」以建立值區。

部署遊戲案例

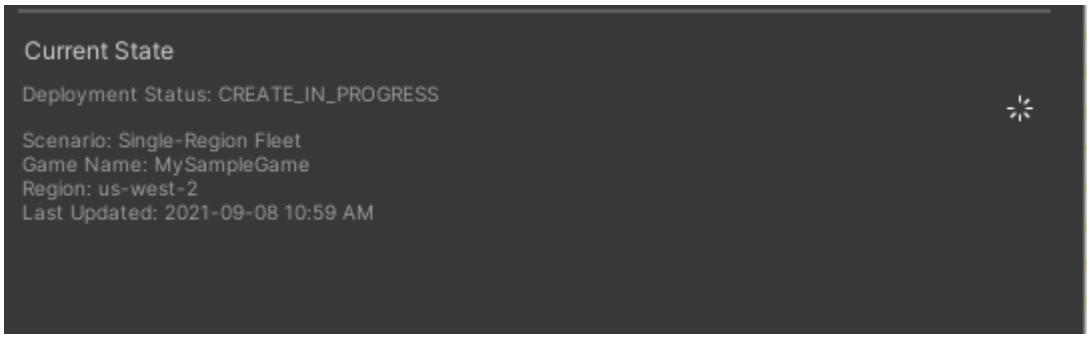
> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

您可以使用場景來測試您的遊戲與 Amazon GameLift。每個案例都使用AWS CloudFormation範本來建立包含所需資源的堆疊。大多數情況下都需要遊戲服務器可執行文件和構建路徑。部署案例時，Amazon 會 GameLift將遊戲資產複製到啟動程序位置，做為部署的一部分。

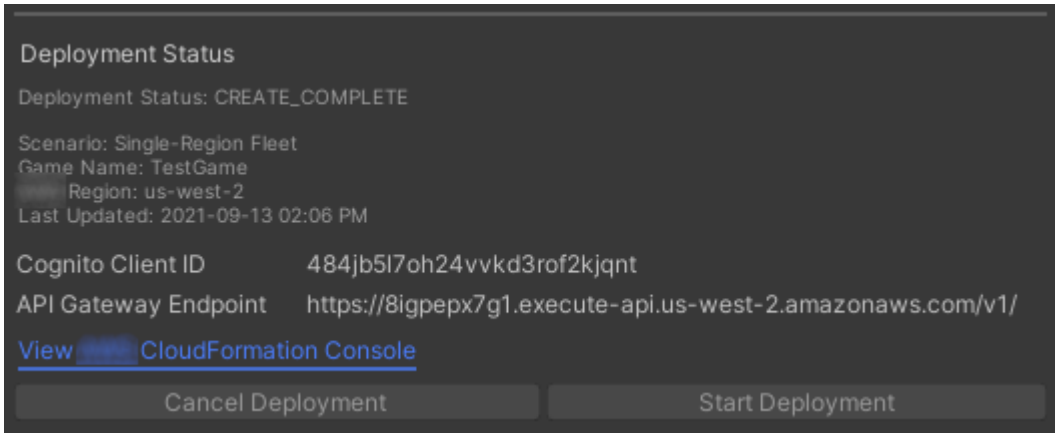
您必須設定AWS認證和AWS帳戶啟動程序，才能部署案例。

若要部署案例

1. 在 Unity 中，在 Unity 索引標籤的外掛程式中，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。當您部署場景時，遊戲名稱是AWS CloudFormation堆疊名稱的一部分。
5. 選擇遊戲伺服器組建資料夾路徑。build 文件夾路徑指向包含服務器可執行文件和依賴項的文件夾。
6. 選擇遊戲伺服器組建 .exe 檔案路徑。構建可執行文件路徑指向遊戲服務器的可執行文件。
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」視窗中追蹤更新的狀態。部署案例最多可能需要 30 分鐘。



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 功能表上選擇 [移至用戶端連線設定]。這將在 Unity 屏幕的右側顯示 Inspector 選項卡。
10. 取消選取本機測試模式。

11. 輸入 API Gateway 端點和可變客戶端 ID。選擇AWS 區域您用於案例部署的相同項目。然後，您可以使用部署的情境資源重建和運行遊戲客戶端。

刪除案例所建立的資源

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

若要刪除針對案例建立的資源，請刪除對應的AWS CloudFormation堆疊。

若要刪除案例所建立的資源

1. 在適用於 Unity 部署的 Amazon GameLift 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主機中，選擇 [堆疊]，然後選擇包含部署期間指定之遊戲名稱的堆疊。
3. 選擇「刪除」以刪除堆疊。刪除堆疊可能需要幾分鐘的時間。AWS CloudFormation刪除案例所使用的堆疊後，其狀態會變更為`ROLLBACK_COMPLETE`。

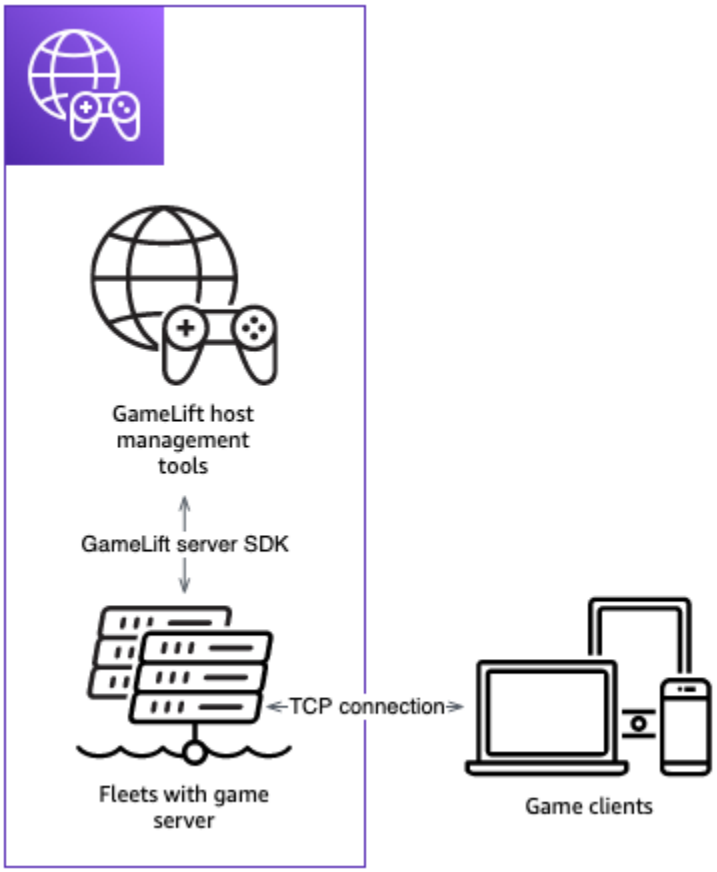
GameLift 在統一中與 Amazon 整合遊戲

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

GameLift 通過完成以下任務將您的 Unity 遊戲與 Amazon 集成：

- 將 Amazon GameLift 與 Unity 遊戲服務器項目集成
- 將 Amazon GameLift 與 Unity 遊戲客戶端項目集成

下圖顯示整合遊戲的範例流程。在圖中，帶有遊戲服務器的艦隊部署到 Amazon GameLift。遊戲客戶端與遊戲服務器進行通信，該服務器與 Amazon 進行通信。 GameLift



匯入並執行範例遊戲

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

用於 Unity 的 Amazon GameLift 插件包括一個示例遊戲，您可以用來探索將遊戲與 Amazon 集成的基礎知識 GameLift。在本節中，您要建置遊戲用戶端和遊戲伺服器，然後使用 Amazon L GameLift ocal 進行本機測試。

必要條件

- 設置一個 AWS 帳戶
- 安裝和設置插件

建置並執行範例遊戲伺服器

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 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 中，在菜單上，選擇 Amazon GameLift，然後選擇導入示例遊戲。
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 的功能表上，選擇 Amazon GameLift，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]。在您設定遊戲伺服器設定之後，Unity 會重新編譯資產。
4. 在 Unity 的功能表上，選擇 [檔案]，然後選擇 [建置]。選擇 [伺服器組建]，選擇 [建置]，然後選擇專門用於伺服器檔案的建置資料夾。

   Unity 構建示例遊戲服務器，將可執行文件和所需的資產放置在指定的構建文件夾中。

建置並執行範例遊戲用戶端

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 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 的功能表上，選擇 Amazon GameLift，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]。設定完遊戲用戶端設定之後，Unity 將重新編譯資產。
2. 在 Unity 的功能表上，選取 [移至用戶端設定]。這將在 Unity 屏幕的右側顯示 Inspector 選項卡。在 Amazon 用 GameLift 戶端設定索引標籤中，選取本機測試模式。
3. 建立遊戲用戶端。在 Unity 中，選擇功能表上的 [檔案]。確認未核取 [伺服器組建]，選擇 [建置]，然後選取專門針對用戶端檔案的建置資料夾。

   Unity 構建示例遊戲客戶端，將可執行文件和所需資產放置在指定的客戶端構建文件夾中。

4. 您尚未建置遊戲伺服器和用戶端。在接下來的步驟中，您可以運行遊戲並查看它如何與 Amazon GameLift 交互。

在本機測試範例遊戲

> **Note**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

運行您使用 Amazon GameLift 本地導入的示例遊戲。

1. 啟動遊戲伺服器 在 Unity 中，在 Unity 索引標籤的外掛程式中，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「部署並執行」。Unity 的外掛程式會啟動遊戲伺服器，並開啟 Amazon GameLift 本機記錄視窗。窗口包含日誌消息，包括遊戲服務器和 Amazon GameLift 本地之間發送的消息。
5. 啟動遊戲客戶端。使用示例遊戲客戶端找到構建位置，然後選擇可執行文件。
6. 在 Amazon GameLift 範例遊戲中，提供電子郵件和密碼，然後選擇「登入」。電子郵件和密碼未驗證或使用。
7. 在 Amazon 示 GameLift 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。然後，遊戲客戶端會啟動遊戲工作階段。您可以在日誌中看到遊戲活動。

遊戲伺服器記錄範例

```
...
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: playerIdx
 0 joined
2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerIdx
 1 joined
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerIdx 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 playerIdx 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
```

Amazon 本 GameLift 地日誌示例

```
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**
>
> 本主題指的是統一版本 1.0.0 的 Amazon GameLift 插件，它使用服務器 SDK 4.x 或更早版本。

在您完成範例遊戲之後，請在 Unity 中關閉伺服器。

1. 在遊戲客戶端中，選擇退出或關閉窗口以停止遊戲客戶端。
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。

將遊戲與虛幻引擎的 Amazon GameLift 插件集成

本節中的主題描述了虛幻引擎 (UE) 的 Amazon GameLift 外掛程式，以及如何使用它來準備您的多人遊戲專案，以便在 Amazon GameLift 託管。使用外掛程式的引導式工作流程完全在您的 UE 開發環境中運作，以完成使用 Amazon 託管的基本要求 GameLift。

Amazon GameLift 是一項全受管服務，可讓遊戲開發人員管理和擴展工作階段型多人遊戲的專用遊戲伺服器。如需 Amazon GameLift 託管的詳細資訊，請參閱亞馬遜如何GameLift工作。

主題

- 關於插件
- 插件工作流
- 安裝虛幻插件
- 設定AWS使用者設定檔
- 設置您的遊戲以進行 Amazon 測試 GameLift Anywhere
- 使用受管 EC2 叢集將您的遊戲部署到雲端託管

關於插件

該插件將 Amazon GameLift 工具和功能添加到 UE 編輯器中。該插件的指導工作流程可 GameLift 將 Amazon 集成到您的遊戲項目中，將工作站指定為本地主機進行測試，並將遊戲服務器部署到 Amazon GameLift 雲託管。

使用插件的預構建託管解決方案來部署您的遊戲。將您的本機工作站設定為主機的 Amazon GameLift 無所不在叢集。對於雲託管，請從兩種常見的部署方案中進行選擇，以不同方式平衡玩家延遲、遊戲工作階段可用性和成本。一種情境包括簡單的 FlexMatch分房系統和規則集。使用這些解決方案可以快速開始使用生產就緒的託管結構，然後根據需要優化和自定義。

該插件包括以下組件：

- 用於 UE 編輯器的插件模塊。安裝插件後，一個新的主菜單按鈕使您可以訪問 Amazon GameLift 功能。
- 具有客戶端功能的 Amazon GameLift 服務 API 的 C ++ 庫。
- Amazon GameLift 服務器 SDK 的虛幻庫（版本 5）。
- 用於測試的內容，包括啟動遊戲地圖和兩個測試地圖，其中包含基本藍圖和 UI 元素，可用於測試伺服器整合。
- 外掛程式在部署遊戲伺服器進行託管時使用的可編輯配置 (AWS CloudFormation範本形式)。

插件工作流

以下步驟描述了使用虛幻引擎的 Amazon GameLift 外掛程式整合和部署遊戲專案的典型方法。您可以透過使用 UE 編輯器和遊戲程式碼來完成這些步驟。

1. 建立連結至您AWS帳戶的使用者設定檔，並為有權使用 Amazon 的有效帳戶使用者提供存取登入資料 GameLift。
2. 將伺服器程式碼新增至您的遊戲專案，以便在執行中的遊戲伺服器與使用 Amazon GameLift 服務之間建立通訊。
3. 將用戶端程式碼新增至您的遊戲專案，讓遊戲用戶端傳送請求 GameLift 到 Amazon 以開始新的遊戲工作階段，然後連線到他們。
4. 使用 Anywhere 工作流程將本機工作站設定為遊戲伺服器的 Anywhere 主機。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。
5. 使用 EC2 主機工作流程上傳整合式遊戲伺服器並部署雲端代管解決方案。當您的遊戲伺服器準備就緒時，請透過外掛程式在本機啟動遊戲用戶端、連線到遊戲工作階段並進行遊戲。

在插件中工作時，您將創建和使用AWS資源，這些操作可能會向正在使用的AWS帳戶產生費用。如果您不熟悉AWS，這些動作可能會涵蓋在AWS免費方案中。

安裝虛幻插件

本節說明將外掛程式新增至虛幻引擎專案的初始安裝工作。當您在虛幻編輯器中打開項目時，可以使用插件功能。

> **Note**
>
> 您可以將 Amazon GameLift 外掛程式與標準版本的 UE 編輯器搭配使用，但是當您封裝遊戲伺服器組建時，您需要使用原始碼建置版本。

開始之前

下面是你需要使用 Amazon GameLift 插件虛幻引擎:

- Amazon GameLift 插件虛幻引擎發布包。[下載網站]。
- Microsoft 視覺工作室 2019 或更新版本。
- 虛幻引擎編輯器的源代碼構建版本。您需要一個源構建的版本來打包多人遊戲的服務器組件。如需詳細資訊，包括其他先決條件，請參閱虛幻引擎文件：
  - 在 GitHub 您需要 GitHub和 Epic Games 帳戶上訪問虛幻引擎源代碼。
  - 從源代碼教程構建虛幻引擎。
- 使用 C++ 遊戲程式碼的多人遊戲專案。如果您正在使用 Blueprint 項目，請參閱有關如何為您的項目生成 C ++ 源代碼的虛幻文檔。

將外掛程式新增至您的遊戲專案

完成下列任務，即可將外掛程式新增至您的遊戲專案。

建置 Amazon GameLift C++ 伺服器開發套件

1. 解壓縮虛幻引擎發布包的 Amazon GameLift 插件以提取兩個 zip 文件：
   - `amazon-gamelift-plugin-unreal-<>-sdk-<>.zip`
   - `GameLift-Cpp-ServerSDK-<>.zip.`

解壓縮這些文件。

2. 開啟資GameLift-Cpp-ServerSDK-<>料夾，然後針對您的平台完成下列指示：Linux 或 Microsoft 視窗。

Linux

1. 執行下列命令：

```
mkdir out
cd out
cmake -DBUILD_FOR_UNREAL=1 ..
make
```

這些指令會建立/lib/aws-cpp-sdk-gamelift-server.so檔案。

2. 複製/lib/aws-cpp-sdk-gamelift-server.so到目amazon-gamelift-plugin-unreal/GameLiftPlugin/Source/GameliftServer/ThirdParty/GameLiftServerSDK/Linux/x86_64-unknown-linux-gnu/錄。

Microsoft Windows

1. 執行下列命令：

```
mkdir out
cd out
cmake -G "Visual Studio 17 2022" -DBUILD_FOR_UNREAL=1 ..
msbuild ALL_BUILD.vcxproj /p:Configuration=Release
```

這些命令會建立下列二進位檔案。

- prefix\bin\aws-cpp-sdk-gamelift-server.dll
- prefix\lib\aws-cpp-sdk-gamelift-server.lib

2. 將檔案複製到目amazon-gamelift-plugin-unreal\GameLiftPlugin\Source\GameliftServer\ThirdParty\GameLiftServerSDK\Win64\錄中。

完成下列工作，在您的遊戲專案檔案中工作。

1. 安裝插件文件。

   a. 找到您的遊戲專案根資料夾，例如... > Unreal Projects/[project-name]/。如果插件文件夾不存在那裡，然後創建它。

   b. 轉到從amazon-gamelift-plugin-unreal-<>-sdk-<>.zip中解壓縮的amazon-gamelift-plugin-unreal文件夾。將GameLiftPlugin資料夾從資gamelift-plugin-unreal料夾複製到遊戲專案目錄中的Plugins資料夾。

2. 將插件添加到**.uproject**文件中。

   a. 在您的遊戲專案根目錄資料夾中，開啟.uproject檔案。

   b. 更新檔案以將 "" 和 GameLiftPlugin "WebBrowserWidget" 新增至Plugins區段並啟用它們。下列程式碼會顯示名為「MyGame」之遊戲的更新.uproject檔案。

```
UnrealProjects > MyGame > MyGame.uproject
{
  ...
  "Plugins": [
    {
      "Name": "ModelingToolsEditorMode",
      "Enabled": true,
      "TargetAllowList": [ "Editor" ]
    },
    {
      "Name": "GameLiftPlugin",
      "Enabled": true
    },
    {
      "Name": "WebBrowserWidget",
      "Enabled": true
    }
  ]
}
```

3. 變更專案的 UE 編輯器版本。

   如果您為一個編輯器版本創建了一個項目，現在想要更改為另一個版本（例如源構建版本），則需要更新該項目。

   在您的遊戲項目根文件夾中，選擇.uproject文件，然後選擇選項切換虛幻引擎版本. 選取新的編輯器版本。

4. 使用您的更新重建項目解決方案。

   a. 在專案根資料夾中，尋找解決方案 (*.sln) 檔案。如果不存在，請選擇該文.uproject件，然後選擇選項生成 Visual Studio 項目文件。

   b. 開啟解決方案檔案，然後建置或重建專案。

5. 驗證外掛程式已在 UE 編輯器中啟用。

   > **Note**
   >
   > 如果您已經開啟編輯器，您可能需要重新啟動編輯器，才能辨識新的外掛程式。

   a. 在您選擇的 UE 編輯器中打開項目。

   b. 檢查新的 Amazon GameLift 菜單按鈕的主編輯器工具欄 [需要圖像]。

   c. 在內容瀏覽器中查看 Amazon GameLift 外掛程式資產。確定您的「檢視選項」設定已選取「顯示外掛程式內容」選項。

設定AWS使用者設定檔

安裝插件後，設置配置文件並將其鏈接到有效的AWS帳戶用戶。您可以維護多個設定檔，但一次只能啟用一個設定檔。每當您在插件中工作時，請選擇要使用的配置文件。

維護多個配置文件使您能夠在不同的託管方案之間切換。例如，您可以使用相同的AWS認證，但不同的AWS區域設定設定檔。或者，您可以使用不同的AWS帳戶或使用不同的用戶/權限集來設置配置文件。

> **Note**
>
> 如果您已在工作站上安裝 AWS CLI 並且已設定描述檔，Amazon GameLift 外掛程式可以偵測到它，並將其列為現有設定檔。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。

若要管理您的AWS設定檔

1. 在虛幻編輯器主工具欄中，選擇 Amazon GameLift 菜單，然後選擇設置AWS用戶配置文件。此動作會開啟外掛程式的「專案設定」。展開「AWS使用者設定檔」區段。

2. 如果插件未檢測到現有配置文件，則會提示您創建一個配置文件。您可以使用新帳戶或現有AWS帳戶建立新的設定檔。

   > **Note**
   >
   > 您必須使用AWS管理主控台來建立新AWS帳戶，並使用適當的權限集建立或更新使用者。

   設定設定檔時，您需要下列資訊：

   - 一個 AWS 帳戶。如果您需要建立新AWS帳戶，請依照提示建立帳戶。如需詳細資訊，請參閱建立AWS帳戶。
   - 有權使用 Amazon GameLift 和其他必要AWS服務的使用AWS者。如設置一個 AWS 帳戶需設定具有 Amazon GameLift 許可的 AWS Identity and Access Management (IAM) 使用者以及使用長期登入資料進行程式設計存取的說明，請參閱。
   - 您的AWS使用者的認證。這些認證包含AWS存取金鑰 ID 和AWS秘密金鑰。如需詳細資訊，請參閱取得存取金鑰。
   - AWS區域。這是您要創建用於託管AWS資源的地理位置。在開發過程中，我們建議您使用靠近您實際位置的區域。請參閱支援的AWS地區清單。
3. 如果插件檢測到現有配置文件，則不會提示您創建一個配置文件。如果您要更新設定檔或建立新的設定檔，請選擇 [管理您的設定檔]。

若要啟動您的設定檔：

必須啟動所有設定檔，才能與 Amazon GameLift 外掛程式搭配使用。啟動安裝會建立特定於該設定檔的 Amazon S3 儲存貯體。它用於存儲項目配置，構建工件和其他依賴項。值區不會在其他設定檔之間共用。

引導涉及創建新的AWS資源，並可能產生成本。

1. 在虛幻編輯器主工具欄中，選擇 Amazon GameLift 圖標，然後選擇設置AWS用戶配置文件。此動作會開啟外掛程式的「專案設定」。展開「AWS使用者設定檔」區段。
2. 在「啟動您的設定檔」區段中，從下拉式清單中選取設定檔，然後檢查啟動程序狀態。如果狀態指出沒有儲存貯體存在，請選擇 Bootstrap 按鈕並建立設定檔，為所選設定檔建立 Amazon S3 儲存貯體。
3. 等待引導狀態更改為「活動」。這可能需要幾分鐘的時間。

設置您的遊戲以進行 Amazon 測試 GameLift Anywhere

在此工作流程中，您可以新增 Amazon GameLift 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。當您完成整合任務後，請使用外掛程式來建置您的遊戲用戶端和伺服器元件。

要啟動 Amazon GameLift 任何地方工作流程：

- 在虛幻編輯器主工具欄中，選擇 Amazon GameLift 菜單，然後選擇託管在任何地方。這個動作會開啟外掛程式頁面隨處部署，提供六個步驟來整合、建置和啟動遊戲元件的程序。

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與設定檔的AWS帳戶相關聯，並放置在設定檔的預設AWS區域中。設定檔使用者的權限決定您對AWS資源和動作的存取權限。

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有個人資料或想要創建一個新的個人資料，請轉到 Amazon GameLift 菜單並選擇設置AWS用戶配置文件。
2. 如果啟動程序狀態不是「活動」，請選擇 Bootstrap 配置文件，然後等待狀態更改為「活動」。

步驟 2：設定您的遊戲代碼

在此步驟中，您會對用戶端和伺服器程式碼進行一系列更新，以新增主機功能。如果您尚未設置虛幻編輯器的源代碼構建版本，則該插件將提供指令和源代碼的鏈接。

使用該插件，在集成遊戲代碼時可以利用一些便利。您可以做一個最小的集成來設置基本的託管功能。您還可以進行更廣泛的自定義集成。本節中的資訊說明最小整合選項。使用外掛程式隨附的測試地圖，將用戶端 Amazon GameLift 功能新增至您的遊戲專案。若要進行伺服器整合，請使用提供的程式碼範例來更新專案的遊戲模式。

整合伺服器遊戲模式

將伺服器程式碼新增至您的遊戲，以啟用遊戲伺服器與 Amazon GameLift 服務之間的通訊。您的遊戲伺服器必須能夠回應 Amazon 的請求 GameLift，例如開始新的遊戲工作階段，並報告遊戲伺服器運作狀態和玩家連線的狀態。

1. 在程式碼編輯器中，開啟您遊戲專案的 solution (`.sln`) 檔案，通常位於專案根資料夾中。例如：`GameLiftUnrealApp.sln`。

2. 解決方案打開後，找到項目遊戲模式頭文件:[project-name]GameMode.hfile。例如：GameLiftUnrealAppGameMode.h。

3. 將標頭檔案變更為與下列範例程式碼對齊。請務必將 "GameLiftServer" 取代為您自己的專案名稱。這些更新僅適用於遊戲伺服器；我們建議您製作原始遊戲模式檔案的備份副本，以便與您的用戶端搭配使用。

```
// 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 "GameLiftServerGameMode.generated.h"

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftServerGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

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

4. 開啟相關的來源[project-name]GameMode.cpp檔案 (例如GameLiftUnrealAppGameMode.cpp)。變更程式碼，使其與下列範例程式碼對齊。請務必將 "GameLiftUnrealApp" 取代為您自己的專案名稱。這些更新專屬於遊戲伺服器；我們建議您製作原始檔案的備份副本，以便與您的用戶端搭配使用。

下面的示例代碼演示了如何添加與 Amazon 服務器集成的最低必需元素 GameLift：

- 初始化 Amazon GameLift API 客戶端。Amazon 無 GameLift 處不在叢集需要具有伺服器參數的`InitSDK()`呼叫。當您連線到 Anywhere 叢集時，外掛程式會將伺服器參數儲存為主控台引數。範例程式碼可以在執行階段存取值。
- 實作必要的回呼函數以回應來自 Amazon GameLift 服務的請求`OnStartGameSession`，包括`OnProcessTerminate`、和`onHealthCheck`。
- 撥打指定`ProcessReady()`的連接埠，在準備好主持遊戲工作階段時通知 Amazon GameLift 服務。

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

#include "GameLiftServerGameMode.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);

AGameLiftServerGameMode::AGameLiftServerGameMode() :
    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 AGameLiftServerGameMode..."));
```

```
}

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

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

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

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

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

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

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

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

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

        FString glAnywhereProcessId = "";
        if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereProcessId="),
glAnywhereProcessId))
        {
            ServerParametersForAnywhere.m_processId =
TCHAR_TO_UTF8(*glAnywhereProcessId);
        }
        else
        {
            // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
            ServerParametersForAnywhere.m_processId =
                TCHAR_TO_UTF8(
                    *FText::Format(
                        FText::FromString("ProcessId_{0}"),
                        FText::AsNumber(std::time(nullptr))
                    ).ToString()
                );
        }

        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);
        }
```

```
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Web Socket 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, 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, GameLift 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. GameLift 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 GameLift we are indeed going to shutdown.
    ProcessParameters->OnTerminate.BindLambda([=]()
    {
        UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
        GameLiftSdkModule->ProcessEnding();
    });

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

    //The game server calls ProcessReady() to tell GameLift 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
}
```

整合您的客戶端遊戲地圖

啟動遊戲地圖包含藍圖邏輯和 UI 元素，這些元素已包含要求遊戲工作階段的基本程式碼，以及使用連線資訊連線至遊戲工作階段。您可以按原樣使用地圖，也可以根據需要修改這些貼圖。將啟動遊戲地圖與其他遊戲資產一起使用，例如虛幻引擎提供的第三人稱模板項目。這些資產可在「內容瀏覽器」中使用。您可以使用它們來測試插件的部署工作流程，或者作為為遊戲創建自定義後端服務的指南。

啟動地圖具有以下特性：

- 它包括任何地方叢集和受管 EC2 叢集的邏輯。當您執行用戶端時，您可以選擇連線至任一叢集。
- 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 (這是部署的任何地方解決方案的一部分)。

使用啟動遊戲地圖

1. 在 UE 編輯器中，打開項目設置，地圖和模式頁面，然後展開默認地圖部分。
2. 對於「編輯器啟動對應StartupMap」，請從下拉式清單中選取「」。您可能需要搜尋位於中的檔案... > `Unreal Projects/[project-name]/Plugins/Amazon GameLift Plugin Content/Maps`。
3. 在遊戲預設地圖中，從下拉式清單中選擇相同的 StartupMap「」。
4. 對於「伺服器預設對映」，請選取「ThirdPersonMap」。這是遊戲專案中包含的預設地圖。這張地圖是為遊戲中的兩名玩家設計的。
5. 開啟伺服器預設對映的詳細資料面板。將「GameMode 覆寫」設定為「無」。
6. 展開「預設模式」區段，然後將「全域預設伺服器遊戲模式」設定為您為伺服器整合而更新的遊戲模式。

在您對專案進行這些變更之後，您就可以建置遊戲元件了。

建置您的遊戲元件

1. 建立新伺服器和用戶端目標檔案

   a. 在您的遊戲項目文件夾中，轉到源文件夾並找到Target.cs文件。

   b. 將檔案複製[project-name]Editor.Target.cs到兩個名為[project-name]Client.Target.cs和的新檔案[project-name]Server.Target.cs。

   c. 編輯每個新檔案以更新類別名稱和目標類型值，如下所示：

```
UnrealProjects > MyGame > Source > MyGameClient.Target.cs
// Copyright Epic Games, Inc. All Rights Reserved.

using UnrealBuildTool;
using System.Collections.Generic;

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

```
UnrealProjects > MyGame > Source > MyGameServer.Target.cs
// Copyright Epic Games, Inc. All Rights Reserved.

using UnrealBuildTool;
using System.Collections.Generic;

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

```
}
```

2. 更新.Build.cs檔案。

   a. 開啟專.Build.cs案的檔案。此檔案位於 UnrealProjects/[project name]/Source/[project name]/[project name].Build.cs：

   b. 更新ModuleRules類別，如下列程式碼範例所示。

```
public class MyGame : ModuleRules
{
  public GameLiftUnrealApp(TargetInfo Target)
  {
    PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
 "Engine", "InputCore" });
    bEnableExceptions = true;

    if (Target.Type == TargetRules.TargetType.Server)
    {
      PublicDependencyModuleNames.AddRange(new string[]
 { "GameLiftServerSDK" });
      PublicDefinitions.Add("WITH_GAMELIFT=1");
    }
    else
    {
      PublicDefinitions.Add("WITH_GAMELIFT=0");
    }
  }
}
```

3. 重建您的遊戲專案解決方案。

4. 在虛幻引擎編輯器的源代碼版本中打開您的遊戲項目。

5. 對用戶端和伺服器執行下列動作：

   a. 選擇一個目標。移至「平台」、「視窗」，然後選取下列其中一項：

      - 伺服器：[your-application-name]Server
      - 用戶端：[your-application-name]Client

   b. 啟動組建。前往「平台」、「視窗」、「Package 專案」。

每個封裝程序都會產生一個可執行檔：[your-application-name]Client.exe或[your-application-name]Server.exe。

在插件中，設置本地工作站上客戶端和服務器構建可執行文件的路徑。

步驟 3：Connect 到任何地方的機隊

在此步驟中，您可以指定要使用的任何地方叢集。Anywhere 叢集定義了一系列運算資源，這些資源可以位於任何地方，以供遊戲伺服器託管使用。

- 如果您目前使用的AWS帳戶具有現有的 Anywhere 叢集，請開啟 [叢集名稱] 下拉式欄位並選擇叢集。此下拉式清單只會顯示目前作用中使用者設定檔的 [AWS區域] 中的 [任何地方] 叢集。
- 如果沒有現有的車隊，或者您想要建立新的叢集，請選擇 [建立新的任何地方] 叢集並提供叢集名稱。

在您為專案選擇任何地方叢集之後，Amazon 會 GameLift 驗證叢集狀態為使用中廣告顯示叢集 ID。您可以在虛幻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，您將本機工作站註冊為新的 Anywhere 叢集中的計算資源。

1. 輸入本機電腦的運算名稱。如果您在叢集中新增多個計算，名稱必須是唯一的。
2. 提供本機電腦的 IP 位址。此欄位預設為您電腦的公用 IP 位址。只要您在同一台機器上運行遊戲客戶端和服務器，您也可以使用本地主機（127.0.0.1）。
3. 選擇 [註冊運算]。您可以在虛幻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 會驗證其是否可以連線到運算，並傳回有關新註冊運算的資訊。它還會創建初始化與 Amazon GameLift 服務的通信時，您的遊戲可執行文件所需的控制台參數。

步驟 5：生成身份驗證令牌

在 Anywhere 運算上執行的遊戲伺服器程序需要驗證 Token 才能對 GameLift 服務進行呼叫。每當您從插件啟動遊戲服務器時，該插件都會自動生成並存儲 Anywhere 叢集的身份驗證令牌。auth 令牌值存儲為命令行參數，您的服務器代碼可以在運行時檢索該參數。

您不必在此步驟中採取任何動作。

步驟 6：啟動遊戲

此時，您已經完成了使用 Amazon 在本地工作站上啟動和玩多人遊戲所需的所有任務 GameLift。

1. 啟動您的遊戲伺服器。遊戲伺服器會在準備好主持遊戲工作階段 GameLift 時通知 Amazon。
2. 啟動您的遊戲客戶端並使用新功能開始新的遊戲工作階段。此請求會 GameLift 透過新的後端服務傳送至 Amazon。作為回應 GameLift，Amazon 調用在本地計算機上運行的遊戲服務器以啟動新的遊戲會話。當遊戲工作階段準備好接受玩家時，Amazon 會 GameLift 提供遊戲用戶端加入遊戲工作階段的連線資訊。

使用受管 EC2 叢集將您的遊戲部署到雲端託管

在此工作流程中，您可以使用外掛程式修改遊戲，以便在 Amazon 管理的雲端運算資源上託管 GameLift。您可以為 Amazon GameLift 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 服務，以部署到雲端資源。完成此工作流程後，您將擁有一個可以連接到雲端中的遊戲伺服器的有效遊戲用戶端。

若要啟動 Amazon GameLift 託管 EC2 工作流程：

- 在虛幻編輯器主工具列中，選擇 Amazon GameLift 功能表，然後選擇託管 EC2。此動作會開啟外掛程式頁面部署 Amazon EC2 叢集，提供六個步驟來整合、建置、部署和啟動遊戲元件的程序。

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與設定檔的AWS帳戶相關聯，並放置在設定檔的預設AWS區域中。設定檔使用者的權限決定您對AWS資源和動作的存取權限。

1. 從可用設定檔的下拉式清單中選取設定檔。如果您還沒有個人資料或想要創建一個新的個人資料，請轉到 Amazon GameLift 菜單並選擇設置AWS用戶配置文件。
2. 如果啟動程序狀態不是「活動」，請選擇 Bootstrap 配置文件，然後等待狀態更改為「活動」。

步驟 2：設定您的遊戲代碼

在此步驟中，您會對用戶端和伺服器程式碼進行一系列更新，以新增主機功能。如果您尚未設置虛幻編輯器的源代碼構建版本，則該插件將提供指令和源代碼的鏈接。

如果您整合了遊戲以搭配 Anywhere 叢集使用，則不需要對遊戲程式碼進行任何變更。如果您使用的是啟動遊戲地圖，這也適用於 EC2 部署。

- 設定您的遊戲代碼 (任何地方)
- 建置您的遊戲元件

建立遊戲伺服器後，請完成以下任務，準備上傳至 Amazon GameLift。

若要封裝伺服器組建以進行雲端部署

在默認情況下，虛幻編輯器打包服務器構建文件的文件`WindowsServer`夾中，進行以下添加

1. 將插件下載中包含的安裝腳本複製到文件`WindowsServer`夾的根目錄中。尋找檔案`[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat`。Amazon GameLift 使用此文件在每個 EC2 託管資源上安裝服務器構建。
2. 將包含在 Visual Studio 安裝中的`VC_redist.x64.exe`檔案複製到`WindowsServer`資料夾的根目錄中。此檔案通常位於`C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`。
3. 將您遊戲伺服器內建的 OpenSSL DLL 複製到資料夾中。`WindowsServer/MyGame/Binaries/Win64`確保 DLL 與服務器構建中使用的版本相同。複製下列檔案：
   - `libssl-3-x64.dll`
   - `libcrypto-3-x64.dll`

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。您可以使用任何情境來進行多個遊戲部署。

- 單一區域叢集：將您的遊戲伺服器部署到作用中設定檔預設AWS區域中的單一主機資源叢集。這個案例是測試伺服器整合與伺服器組建組態AWS的良好起點。它會部署下列資源：
  - AWS已安裝並執行您的遊戲伺服器組建的叢集 (隨選)。
  - Amazon Cognito 使用者集區和用戶端可讓玩家進行身份驗證和開始遊戲。
  - 將使用者集區與 API 連結的 API 閘道授權者。
  - WebACL 用於限制對 API 網關的過多播放器調用。
  - API 網關 + Lambda 功能，供玩家請求遊戲位置。`CreateGameSession()`如果沒有可用這個函數調用。

  - API 閘道 + Lambda 函數可讓玩家取得遊戲請求的連線資訊。
- FlexMatch 艦隊：將您的遊戲伺服器部署到一組艦隊，並設定具有建立玩 FlexMatch 家對戰規則的分房系統。此案例使用低成本的 Spot 主機，搭配多叢集、多地點結構，以提供持久的可用性。當您準備好開始為您的主機解決方案設計分房系統元件時，此方法非常有用。在這個案例中，您將建立此解決方案的基本資源，您可以在稍後視需要自訂這些資源。它會部署下列資源：
  - FlexMatch 配對配對配置和配對規則設置為接受玩家請求和形式匹配。
  - 您的遊戲伺服器組建安裝並在多個位置執行的三個AWS艦隊。包括兩個 Spot 叢集和一個隨需叢集做為備份。
  - AWS遊戲會話放置隊列，通過找到可能的最佳託管資源（基於可行性，成本，玩家延遲等）並啟動遊戲會話來滿足提議的匹配請求。
  - Amazon Cognito 使用者集區和用戶端可讓玩家進行身份驗證和開始遊戲。
  - 將使用者集區與 API 連結的 API 閘道授權者。
  - WebACL 用於限制對 API 網關的過多播放器調用。
  - API 網關 + Lambda 功能，供玩家請求遊戲位置。此函數會呼叫`StartMatchmaking()`。
  - API 閘道 + Lambda 函數可讓玩家取得遊戲請求的連線資訊。
  - Amazon DynamoDB 資料表可用來儲存玩家的配對票證和遊戲工作階段資訊。
  - SNS 主題 + Lambda 函數來處理 GameSessionQueue 事件。

步驟 4：設定遊戲參數

在此步驟中，您將描述要上傳到的遊戲AWS。

- 伺服器組建名稱：為您的遊戲伺服器組建提供有意義的名稱。 AWS使用此名稱來參考已上傳並用於部署的伺服器組建副本。
- 服務器構建操作系統：輸入您的服務器構建運行的操作系統。這會告訴您要使用AWS哪種類型的運算資源來託管您的遊戲。
- 遊戲伺服器資料夾：識別本機伺服器建置資料夾的路徑。
- 遊戲服務器構建：識別遊戲服務器可執行文件的路徑。
- 遊戲客戶端路徑：識別遊戲客戶端可執行文件的路徑。
- 客戶端配置輸出：此字段需要指向客戶端構建中包含AWS配置的文件夾。在以下位置尋找它：`[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，將遊戲部署到雲端主機解決方案。此程序可能需要長達 40 分鐘的時間，同時AWS驗證您的伺服器組建、佈建主機資源、安裝遊戲伺服器、啟動伺服器程序，以及讓它們準備好主持遊戲工作階段。

若要開始部署，請選擇「部署」 CloudFormation。您可以在此處跟踪遊戲託管的狀態。如需詳細資訊，您可以登入AWS管理主控台以查看事件通知。AWS請務必使用與插件中活躍用戶配置文件相同的帳戶，用戶和AWS區域登錄。

部署完成後，您將遊戲伺服器安裝在 AWS EC2 執行個體上。至少有一個伺服器處理序正在執行，並準備好啟動遊戲工作階段。

步驟 6：啟動用戶端

此時，您已經完成了啟動和遊玩由 Amazon 託管的多人遊戲所需的所有任務 GameLift。若要進行遊戲，請啟動您的遊戲用戶端執行個體。

如果您部署了單一叢集案例，則可以使用一個播放器開啟單一用戶端執行個體、輸入伺服器對應並四處移動。開啟遊戲用戶端的其他實體，將第二位玩家新增至同一伺服器的遊戲地圖。

如果您部署了這個 FlexMatch 案例，解決方案會等待至少兩個用戶端排入遊戲工作階段的位置，然後玩家才能進入伺服器地圖。

取得 Amazon GameLift 執行個體的叢集資料

在某些情況下，您的自定義遊戲構建或實時服務器腳本可能需要有關 Amazon GameLift 車隊的信息。例如，您的遊戲組建或指令碼可能包含下列項目的程式碼：

- 根據車隊資料監控活動。
- 彙整指標以依車隊資料追蹤活動。（許多遊戲將此數據用於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。

叢集資訊可在下列位置以 JSON 檔案的形式取得，以每個執行個體的形式存取：

- Windows：`C:\GameMetadata\gamelift-metadata.json`
- Linux：`/local/gamemetadata/gamelift-metadata.json`

該`gamelift-metadata.json`文件包括亞馬遜GameLift叢集資源的屬性。

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",
    "fleetName":"ReallyFunGameTestFleet08",
    "fleetType":"ON_DEMAND",
    "instanceRoleArn":"arn:aws:iam::123456789012:role/S3AccessForGameLift",
    "instanceType":"c5.large",
    "serverLaunchPath":"/local/game/reallyfungame.exe"
}
```

新增FlexMatch配對

使用亞馬遜GameLiftFlexMatch將玩家配對功能添加到您的亞馬遜GameLift託管遊戲中。您可以FlexMatch與自定義遊戲服務器或實時服務器一起使用。

FlexMatch 會將配對服務與自訂規則引擎結合在一起。您可以根據對您的遊戲有意義的玩家屬性和遊戲模式來設計如何將玩家匹配在一起。FlexMatch管理評估正在尋找比賽的球員，與一個或多個團隊組成比賽，並開始遊戲會話主持比賽的螺母和螺栓。

若要使用完整FlexMatch服務，您必須使用佇列設定主機資源。Amazon GameLift 使用佇列為多個區域和運算類型的遊戲尋找最佳託管位置。特別是，Amazon GameLift 佇列可以使用遊戲用戶端提供的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包括將配對整合至遊戲的詳細說明的詳細資訊，請參閱下列 Amazon GameLift FlexMatch 開發人員指南主題：

- 亞馬遜如何GameLiftFlexMatch工作
- FlexMatch整合步驟

使用 Amazon GameLift 容器管理託管

本文件適用於公開預覽版本中的功能。內容可能變動。

Amazon GameLift 提供完整的雲端託管服務，以支援遊戲伺服器託管的容器化解決方案。使用 Amazon GameLift 容器叢集，您可以充分利用容器優勢，例如可攜性、敏捷性和容錯能力。

主要功能

下列功能適用於 Amazon GameLift 容器叢集。

- 使用輕量型容器開發自訂容器架構，以在 Amazon GameLift 託管資源上執行遊戲伺服器軟體。
- 包括 Amazon GameLift 代理程式，以管理容器內遊戲伺服器程序的生命週期。計算上的代理程式會執行您的指示，說明何時以及如何啟動伺服器處理序，以及遊戲工作階段託管要維護的數量。
- 自訂 Amazon 提供的資源， GameLift 以使用您的遊戲伺服器應用程式建置容器映像檔。使用提供的碼頭文件來創建基於 Linux 的容器映像。將容器叢集的映像存放在亞馬遜彈性容器登錄 (Amazon ECR) 私有儲存庫中。
- 將容器叢集資源部署到 Amazon GameLift 支援的任何 AWS 區域 或本機區域，提供低延遲的玩家體驗。建立多地點容器叢集以簡化叢集管理。請參閱Amazon GameLift 託管地點。
- 使用 Amazon GameLift Anywhere 叢集測試您的容器化遊戲託管解決方案。使用任何地方在本機測試您的解決方案開發，包括 Amazon GameLift SDK 整合和容器映像組態。
- 利用容器特定效能指標追蹤遊戲主機效能。使用硬體指標監控叢集資源的健全狀況。
- 使用 Amazon GameLift 遊戲工作階段放置工具 (包括佇列和 FlexMatch配對)，將玩家與容器叢集上託管的最佳遊戲工作階段進行配對。
- 使用適用於 Amazon 的 AWS CloudFormation 範本來管理容器叢集資源 GameLift。

在公開預覽期間使用容器叢集

新的容器叢集功能目前正在公開預覽中。在此階段，支援下列 Amazon GameLift 功能：

- 使用容器叢集來託管專為 Linux 建置的遊戲伺服器。容器叢集會使用`Amazon_Linux_2023`並支援 Linux 容器映像檔。不支援視窗容器。
- 僅將遊戲服務器項目與 Amazon GameLift 服務器 SDK 版本 5+ 集成。不支援以前的版本。

- 使用 Amazon GameLift 支援的任何 Amazon EC2 隨需執行個體類型。目前不支援 Spot 叢集。

容器在 Amazon 的運作方式 GameLift

本文件適用於公開預覽版本中的功能。內容可能變動。

Amazon GameLift 容器叢集的設計旨在提供您部署和擴展容器化應用程式的彈性。它使用 Amazon Elastic Container Service (Amazon ECS) 來管理 Amazon 車 GameLift 隊的任務部署和執行。本主題說明在 Amazon GameLift 受管叢集上執行容器的基本結構元素、說明常見架構，並概述了一些核心概念。

貨櫃車隊元件

機群

容器叢集是由 Amazon 管理的 Amazon EC2 執行個體集合，可執行您的容器化遊戲伺服器。GameLift建立叢集時，您可以設定容器架構和遊戲伺服器軟體的部署方式至每個叢集執行個體。您可以將容器叢集部署到單一 AWS 區域 或多個地理位置。您可以使用 Amazon GameLift 手動或自動擴展工具擴展容器叢集的容量，以託管遊戲工作階段和玩家。

執行個體

Amazon EC2 執行個體是為您的遊戲託管提供運算容量的虛擬伺服器。使用 Amazon GameLift，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、儲存和網路容量組合。

建立容器叢集時，Amazon GameLift 會根據您選擇的執行個體類型和叢集組態部署執行個體。每個已部署的叢集執行個體都是相同的，並以相同的方式執行您的容器化遊戲伺服器軟體。叢集中的執行個體數量決定了叢集的大小和遊戲託管容量。

容器群組

Amazon GameLift 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「工作」或「pod」。在每個容器群組中，您可以定義容器如何共用可用的 CPU 和記憶體資源。您也可以設定容器之間的相依性，以及管理容器群組的生命週期。

容器群組可跨每個叢集執行個體進行複寫，以最佳化資源使用 您可以透過設定容器群組的排程策略來管理複寫，如下所示：

- 複本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。所有容器叢集都必須定義複本容器群組。複本群組在每個叢集執行個體上可能會有多個副本，視容器群組的需求和使用中的執行個體類型的資源而定。複本群組中的所有容器會自動跨執行個體擴充。

- 守護程式容器群組 (選擇性) 對於執行背景服務或公用程式 (例如監視) 可能很有用。您的遊戲伺服器軟體並不直接依賴守護程式群組中的處理程序。精靈容器群組不會複製-每個叢集執行個體最多只有一個精靈群組副本。這表示精靈群組中的容器不會與複本群組中的容器一起跨叢集執行個體擴充。

容器叢集必須有一個複本容器群組，並且可以選擇性地擁有一個精靈群組。

容器

容器是容器架構中最基本的元素。它由帶有軟件可執行文件和依賴文件的容器映像組成。當您定義要與 Amazon 搭配使用的容器時 GameLift，您可以設定軟體在容器中的執行方式。

容器叢集中的每個容器群組都必須有一個指定為「必要」的容器。必要的容器驅動容器群組的生命週期。如果基本容器失敗，整個容器群組就會重新啟動。

容器類型包括：

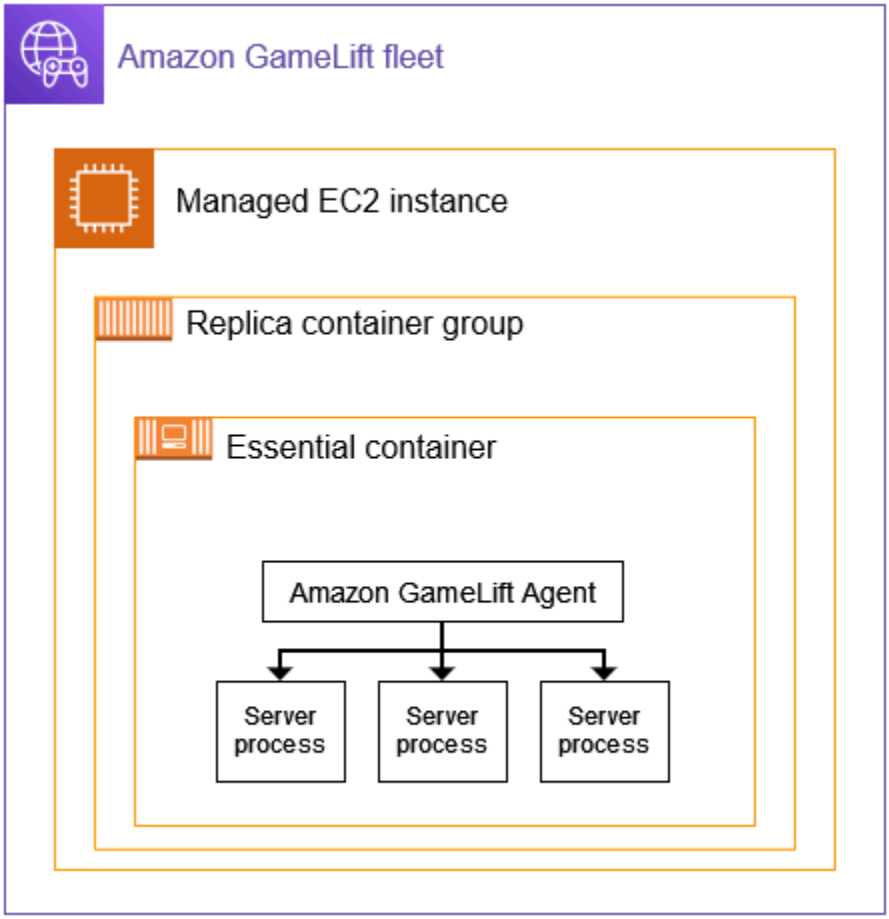
- 基本複製容器包含執行遊戲伺服器程序和為玩家主持遊戲工作階段所需的一切。它包括與 Amazon 伺服器 SDK 整合的遊戲 GameLift 伺服器組建，以及相依軟體。它還包括管理遊戲伺服器程序生命週期的 Amazon GameLift 代理程式。叢集的複本容器群組只有一個基本複本容器。
- 非必要的複本容器 (也稱為「附屬」容器) 可執行軟體來支援您的遊戲伺服器應用程式。使用並行容器可讓您在遊戲伺服器旁邊執行和擴充支援軟體，但可以作為單獨的容器進行管理。如果此類型的容器失敗，則只有容器本身會重新啟動；容器群組不會受到影響。
- 守護程式容器會執行精靈服務來管理背景處理程序。協助程式容器的常見用途是執行 Amazon CloudWatch (CloudWatch) 代理程式來收集容器的指標、日誌和追蹤。精靈容器可能是必要的或非必要的，具體取決於容器故障必須導致容器群組重新啟動的時間。

運算

運算是一種託管資源的叢集，已向 Amazon GameLift 服務註冊，並且能夠與服務進行通訊。在容器叢集中，運算是具有管理計算註冊程序的程序的容器。在容器叢集的基本複本容器中，Amazon GameLift Agent 會自動將此容器註冊為運算。

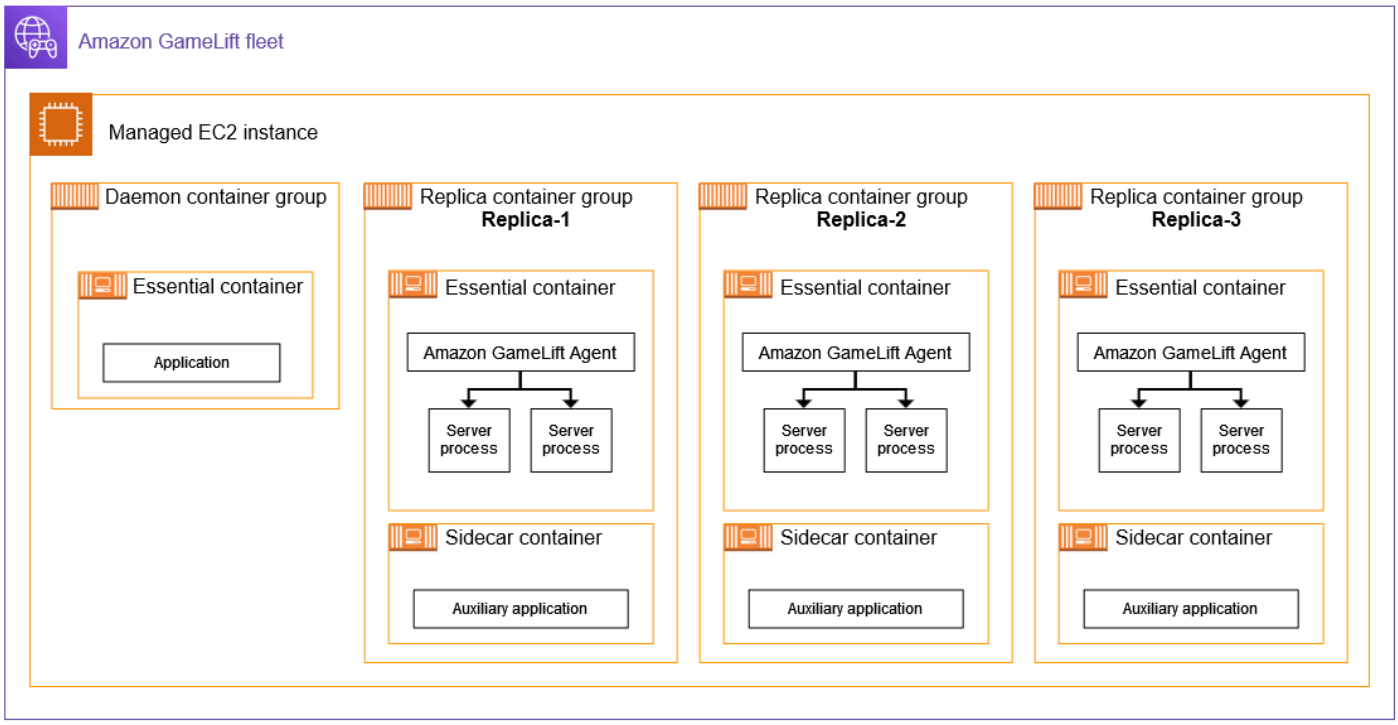
通用架構

下圖說明最簡單的容器叢集結構。在此結構中，叢集中的每個執行個體都會維護複本容器群組的一個副本。容器群組有一個執行 Amazon GameLift Agent、遊戲伺服器應用程式的必要容器，以及用於託管遊戲工作階段的所有支援軟體。代理程式會實作叢集特定指令，以同時執行三個伺服器處理序。因為每個執行個體都有一個複本容器群組，所以每個叢集執行個體會同時執行三個伺服器處理



第二個範例說明更複雜的容器叢集設計。在此範例中，叢集具有具有多個容器的複本容器群組，以及具有一個容器的精靈容器群組。叢集設定會在每個叢集執行個體上放置三個複本容器群組副本。永遠不會複寫精靈容器群組。

中的每組複本群組容器在每個執行個體上都有三個副本。在每個基本複本容器中，會指示代理程式同時執行兩個伺服器處理序。因此，每個叢集執行個體會同時執行六個伺服器處理序 (三個必要複本容器中的每個處理序都有兩個處理序)。



核心概念

本節總結了 Amazon 如何 GameLift 實作某些基本容器概念。如需如何使用容器叢集的指示，請參閱本指南中的相關主題。

集裝箱群包裝

在開發容器叢集中進行部署的容器結構時，共同的目標是將可用的運算能力使用最佳化。若要達成此目標，請找出可放置在叢集執行個體上的最多複本容器群組，而不會影響遊戲伺服器效能。

Amazon GameLift 可以幫助你做到這一點。它會根據下列資訊，計算每個執行個體的複本群組數目上限：

- 叢集的執行個體類型以及可用的 CPU 和記憶體資源。
- 您為複本群組中的所有容器設定的 CPU 和記憶體需求。

  您為精靈群組中所有容器設定的 CPU 和記憶體需求 (如果有的話)。
- 保留的資源可用來管理每個執行個體上的容器和其他重要應用

建立容器叢集時，您可以選擇使用計算出的最大數目，也可以指定所需的數字來覆寫已計算的數目。最佳做法是嘗試您的容器化遊戲伺服器軟體，以判斷正確的資源需求。使用此資料找出遊戲伺服器效能的最佳封裝策略。

遊戲伺服器和 Amazon GameLift 代理

建置基本複本容器時，您可以將遊戲伺服器軟體和 Amazon GameLift Agent 封裝在同一個容器映像中。這個運算上的代理程式可控制容器中遊戲伺服器的生命週期。在每個複本容器群組中，必要的複本容器會執行代理程式和所有遊戲伺服器處理序。

Amazon GameLift 代理程式會在容器叢集的執行階段組態中執行指示。執行階段組態會識別 (1) 要開始執行的可執行檔、(2) 一組選擇性的啟動參數，以及 (3) 要同時執行的處理序數目。一個運行時配置可以有多個不同的可執行文件的指令。遊戲伺服器可執行檔必須至少有一個指令。例如，執行階段設定可能會指示代理程式維護 10 個遊戲伺服器可執行程式的處理程序以供生產使用、1 個具有特殊啟動參數的相同執行程序以供測試，以及記錄公用程式的 1 個程序。

您可以隨時修改叢集的執行階段組態。Amazon GameLift 代理程式會定期向服務要求更新。當有更新的執行階段組態可用時，代理程式會接收該組態並開始實作指示。動作可能包括新增或關閉伺服器處理序。

Amazon GameLift 代理程式是 Amazon GameLift 用於受管 EC2 叢集的運算上代理程式的開放原始碼版本。本指南提供如何從來源建置代理程式並將其建置到容器映像中的指示。代理程式會處理下列工作：

伺服器處理序管理：

- 根據執行階段組態啟動、關閉和取代伺服器處理序。
- 關閉服務器進程，當他們沒有及時激活。
- 服務器進程終止 GameLift 時向 Amazon 報告。
- 發出伺服器處理序的叢集事件。

容器管理：

- 關閉伺服器程序以回應 Amazon 的提示 GameLift。
- 報告容器健康狀況。

記錄檔上傳工作：

- 將遊戲工作階段日誌上傳到指定的 Amazon S3 儲存貯體。
- 將運算上的代理程式日誌上傳到指定的 Amazon S3 儲存貯體。

擴展機群容量的規模

叢集容量衡量叢集可以在任何時間託管的遊戲工作階段數量。您還可以根據艦隊可以同時支持的玩家數量來測量容量。

若要增加或減少叢集的託管容量，您可以新增或移除叢集執行個體。容器叢集的封裝策略決定了在每個叢集執行個體上同時執行多少個遊戲工作階段。這個數字告訴您增加或減少艦隊容量時，您增加或減少的遊戲工作階段數量（和玩家位置）。

透過容器叢集，您可以使用 Amazon GameLift 提供的任何擴展方法。其中包含：

- 透過設定特定所需的叢集執行個體計數，手動設定叢集容量。
- 透過定位所需的可用執行個體緩衝區 (目標追蹤) 來設定自動擴展。此方法會自動維護一組閒置的託管資源，以便傳入的玩家始終可以快速進入遊戲。隨著玩家需求的增加或減少，此緩衝區的大小會不斷調整。
- 使用自定義縮放規則設置自動縮放（高級功能）。

遊戲客戶端/伺服器連線

受管 EC2 叢集和容器叢集以類似的方式處理遊戲用戶端與雲端託管遊戲伺服器之間的連線。Amazon GameLift 建立新的遊戲工作階段時，服務會傳達遊戲工作階段的連線資訊。遊戲用戶端會使用這些資訊直接連線到主控遊戲工作階段的遊戲伺服器。對於所有類型的叢集，連線資訊都包含 IP 位址和連接埠指派。

建立容器叢集時，您可以定義兩組連接埠範圍。首先，您需要定義一系列面向外部的連線連接埠，讓遊戲用戶端連線到遊戲。其次，您定義一組僅限內部的容器連接埠，這些連接埠會指派給容器中執行的每個遊戲伺服器處理序。Amazon 會將內部容器連接埠 GameLift 動態對應至外部連線連接埠，讓玩家能夠存取遊戲。此方法可保護您的遊戲伺服器避免直接存取容器連接埠，以提供額外的安全性。

為容器叢集定義連接埠範圍時，您必須提供足夠的連接埠範圍，以容納執行個體上容器同時執行的所有伺服器處理序。

若要進行其他控制，您也可以為叢集設定輸入權限。輸入權限會決定哪些連線通訊埠會針對傳入流量開啟。您可以隨時變更叢集的入埠權限。使用入站權限，您可以根據需要快速關閉所有連接埠、開啟部分連接埠或全部開啟。

Amazon GameLift 容器的開發藍圖

本文件適用於公開預覽版本中的功能。內容可能變動。

下列工作流程總結了讓您的遊戲伺服器在 Amazon GameLift 容器叢集上執行的步驟。

第 1 步：將您的遊戲與 Amazon 集成 GameLift

將功能新增至您的遊戲伺服器，以便在 Amazon GameLift 服務部署到容器叢集時可以與 Amazon 服務通訊。如果您使用的是 FlexMatch 配對功能，請將此功能新增至您的遊戲伺服器和用戶端。如需詳細資訊，請參閱 將您的遊戲與 Amazon 整合 GameLift。

- 取得 Amazon GameLift 伺服器 SDK (版本 5 以上)，並透過您的遊戲專案進行設定。服務器 SDK 是在 C ++，C# 和轉到可用。
- 修改您的遊戲伺服器程式碼以新增必要的伺服器 SDK 功能。
- 為 Linux Package 您的遊戲伺服器組建。如果您是在 Windows 上進行開發，則此步驟可能需要額外的工作來設定 Linux 環境。
- (選擇性) 使用 Amazon GameLift Anywhere 叢集測試您的遊戲伺服器整合。在準備容器映像之前進行測試，以隔離整合工作的問題。若要測試遊戲用戶端/伺服器連線，請同時整合您的遊戲用戶端。

> **Note**
>
> 如果您是在 Windows 上進行開發，請設定個別的 Linux 工作區，或使用工具，例如適用於 Linux 的視窗子系統 (WSL)。您需要 Linux 環境來測試您的遊戲伺服器組建，以及建置和測試容器映像檔。

步驟 2：準備您的遊戲伺服器容器映像檔

建立執行遊戲伺服器程序的容器映像，並將其存放在 Amazon Elastic Container Registry (Amazon ECR) 儲存庫中，以便與 Amazon GameLift 搭配使用。如需詳細說明，請參閱 使用您的遊戲伺服器軟體準備容器映像檔。

- 使用 Linux 遊戲建置、安裝指令碼，以及所有支援的軟體和相依性，為您的容器映像檔設定工作目錄。
- 取得 Amazon GameLift 代理程式原始程式碼、建置它，然後將`jar`檔案新增至您的工作目錄。
- 取得預設 Docker 檔案並加以修改，使用您的遊戲伺服器軟體設定容器映像檔。
- 建立您的容器映像檔。在 Linux 環境中執行此步驟。

- 建立 Amazon ECR 私有儲存庫，然後將您的容器映像推送至該儲存庫。在您計劃部署容器叢集的相同 AWS 帳戶 AWS 區域 位置中建立存放庫。
- (選擇性) 使用Anywhere叢集測試容器映像檔。您可以設定執行階段組態，將指示傳遞給 Amazon GameLift 代理程式。

步驟 3：建立容器和容器群組

為 Amazon 上的遊戲託管設計容器架構 GameLift。請參閱 設計 Amazon GameLift 貨櫃車隊 和 為 Amazon 容器叢集建立 GameLift 容器群組定義。

- 定義您的容器組態。對於每個容器，您將定義諸如運行時進程，內存分配，運行狀態檢查，網絡端口等問題。
- 使用 Amazon GameLift 主控台或 AWS CLI 以您的容器組態建立容器群組定義。當您建立容器群組定義時，Amazon GameLift 會在當時擷取每個容器映像的快照。

步驟 4：將您的容器化遊戲伺服器部署至容器叢集

使用在上一個步驟中建立的容器群組定義來建立容器叢集，並部署您的容器化遊戲伺服器軟體。請參閱建立 Amazon GameLift 容器叢集。

- 使用 Amazon GameLift 主控台或 AWS CLI 建立容器叢集。
- 在叢集執行個體部署和啟動時追蹤叢集狀態。檢查叢集建立事件，確認叢集是否已成功部署到所有位置。
- 確認遊戲用戶端可以要求並加入遊戲工作階段並進行遊戲。如果您已設定配對功能，請測試這些情境。

步驟 5：管理您的車隊

當您準備生產層級的使用時，建置您的遊戲主機解決方案並管理您的主機生命週期。

- 在其他車隊中創建多地點車隊和艦隊 AWS 區域 以支持您的玩家群。
- FlexMatch 使用佇列或配對功能設定遊戲託管位置。請參閱以下資源：
  - 為遊戲會話放置設置亞馬遜GameLift隊列
  - FlexMatch 開發人員指南
- 設定自動調整規模，以根據玩家對遊戲工作階段的需求來管理車隊容量。
- 為您的容器叢集設定監控。使用 Amazon GameLift 指標、擷取遊戲工作階段日誌和容器日誌、設定對個別容器的遠端存取。
- 建立容器車隊的長期管理。使用叢集別名簡化更新容器叢集的程序。建立 AWS CloudFormation 範本以管理叢集生命週期。請參閱以下資源：
  - 在 Amazon GameLift 叢集中新增別名

- 使用管理資源 AWS CloudFormation

將您的遊戲與 Amazon 整合 GameLift

本文件適用於公開預覽版本中的功能。內容可能變動。

在您可以使用遊戲伺服器軟體建立容器映像並將其部署到 Amazon GameLift 進行雲端託管之前，請先將您的遊戲專案與 Amazon GameLift 伺服器 SDK 整合，然後建置要在 Linux 上執行的遊戲伺服器。本主題介紹 Amazon GameLift 提供的各種整合工具。

託管遊戲伺服器必須能夠與 Amazon GameLift 服務進行通訊。將 Amazon GameLift 伺服器 SDK (版本 5 以上) 新增至您的遊戲專案，並修改遊戲的伺服器程式碼，以設定通訊。Amazon GameLift 提供伺服器 SDK 資源和文件，以支援多種語言和遊戲引擎。

容器化遊戲伺服器的整合程序與整合遊戲伺服器以在受管 EC2 或 Amazon GameLift Anywhere 叢集上託管的程序幾乎相同。

整合工具

Amazon 針對整合 GameLift 提供下列工具和語言支援：

虛幻引擎開發人員

使用虛幻的輕量級插件。此外掛程式包含具有所需 Amazon GameLift 功能的 C++ 伺服器 SDK 程式庫。使用文件為外掛程式設定您的虛幻遊戲專案，並使用提供的程式碼區塊更新您的遊戲程式碼，為您的伺服器和用戶端組建新增必要的功能。

- SDK 外掛程式下載
- 指南：將您的虛幻項目與 Amazon 集成 GameLift
- 參考指南：適用於虛幻的 C++ 伺服器 SDK 5

注意：虛幻引擎專用的 Amazon GameLift 獨立外掛程式不支援使用容器叢集。

對於統一開發者

使用 Unity 的輕量級插件。此外掛程式包含具有必要 Amazon GameLift 功能的 C# 伺服器 SDK 程式庫。使用文件為外掛程式設定您的虛幻遊戲專案，並使用提供的程式碼區塊更新您的遊戲程式碼，為您的伺服器和用戶端組建新增必要的功能。

- SDK 外掛程式下載
- 指南：將您的 Unity 項目與 Amazon 集成 GameLift
- 參考指南：適用於統一的 C# 伺服器 SDK 5

注意：適用於 Unity 的 Amazon GameLift 獨立外掛程式不支援使用容器叢集。

使用其他遊戲引擎的開發者

請遵循以下一般伺服器與用戶端整合指南：

- 整合遊戲伺服器
- 整合遊戲用戶端

Amazon GameLift 提供以下語言的服務器 SDK 5 庫：

- 適用於 C ++ 的服務器 SDK 5 [SDK 下載] [參考指南]
- 適用於 C＃ 的服務器 SDK 5 [SDK 下載] [參考指南]
- 適用於 Go 的服務器 SDK 5 [SDK 下載] [參考指南]

建置適用於 Linux 的遊戲伺服器

Amazon GameLift 容器叢集支援在 Linux 平台上執行的遊戲伺服器。以下是一些建立 Linux 目標遊戲伺服器的秘訣：

- 如果您使用 Unity 遊戲引擎開發遊戲，則遊戲編輯器會提供內建支援，而不需要針對 Linux 建置任何特殊需求。
- 如果您要使用 C++ 開發遊戲，則在建置適用於 C++ 的 Amazon GameLift 伺服器開發套件時，以及建置遊戲伺服器時，必須包含 Linux 專用 OpenSSL 程式庫。同時在您的遊戲伺服器容器映像檔中包含相同的程式庫。
- 如果您要在 Windows 上使用虛幻引擎開發遊戲，請考慮以下選項：
  - 使用虛幻引擎來設定交叉編譯工具鏈。
  - 設定個別的 Linux 工作區，或使用工具，例如 Windows 子系統 (WSL)。您可以使用此環境在 Linux 上執行虛幻編輯器來建置您的遊戲伺服器。

在本地測試您的整合

您可以使用 Amazon GameLift Anywhere 叢集在本機測試您的遊戲整合。此方法是協助隔離與整合直接相關的問題的最佳作法。Anywhere艦隊是運行測試應用程序和遊戲場景的有用工具，例如啟動/停止遊戲會話和跟踪玩家連接。您可以使用Anywhere叢集更快速地反覆建置和測試，從而提供更高的主機活動可見度。

使用亞馬遜機GameLiftAnywhere隊測試您的整合如需使用 Amazon GameLift Anywhere 叢集進行整合測試的說明，請參閱。設定測試環境的工作流程如下所示：

1. 設定執行 Linux 的本機裝置。
2. 建立一個Anywhere艦隊。為您的本機裝置建立自訂位置、建立Anywhere叢集，然後將本機裝置註冊為叢集中的運算。
3. 取得遊戲伺服器的驗證權杖。您的整合式伺服器程序需要憑證才能透過 Amazon GameLift 服務進行驗證。您可以針對同時執行的多個伺服器處理序重複使用相同的 Token。只有在使用Anywhere叢集進行整合測試時，才需要執行此步驟。

   > **Note**
   >
   > 身份驗證令牌是臨時的，必須定期刷新。考慮將腳本添加到服務器構建包以請求新令牌。

4. 更新您的遊戲伺服器程式碼Anywhere。在任何地方叢集上執行時，遊戲伺服器需要使用下列伺服器參數呼叫伺服器 SDK 動作 `InitSdk()` (C++) (C#) (Unreal)。只有在使用Anywhere叢集進行整合測試時，才需要執行此步驟。將 Amazon GameLift 代理程式新增至容器映像後，它會自動處理這些參數。

   最佳作法是設定伺服器程式碼，從環境變數或您在啟動時指定的主控台引數提取這些值。

   - `webSocketUrl`— 使用從呼叫傳回的`GameLiftServiceSdkEndpoint`值`register-compute`。
   - `processId`— 指派伺服器處理序的唯一識別碼。
   - `fleetId`— Anywhere 叢集識別碼，這是從呼叫傳回至`create-fleet`.
   - `authToken`-有效的身份驗證令牌，從調用返回`get-compute-auth-token`。
5. 在您的本機電腦上，設定遊戲伺服器建置軟體並啟動伺服器程序。

   如果伺服器整合成功，伺服器處理序會呼叫伺服器 SDK 動作`InitSDK()`以建立與 Amazon GameLift 服務的連線，然後呼叫通知服務已準備好主持遊戲工作階段。`ProcessReady()`

6. 開始遊戲工作階段。如果您已整合遊戲用戶端來要求遊戲工作階段，您可以使用它來要求新的遊戲工作階段。如果沒有，請使用 AWS CLI 指令 create-game-session。Amazon GameLift 創建一個GameSession對象並啟動該過程以啟動新的遊戲會話。

   如果您的整合正常運作，Amazon 會 GameLift 呼叫本機工作站上的伺服器程序來啟動新的遊戲工作階段 (使用onStartGameSession()回呼)。當遊戲會話準備好供玩家使用時，服務器進程調用ActivateGameSession()。作為回應，Amazon GameLift 更新GameSession狀態和連接信息，以便遊戲客戶端可以連接到遊戲會話並進行遊戲。

使用您的遊戲伺服器軟體準備容器映像檔

本文件適用於公開預覽版本中的功能。內容可能變動。

容器是 Amazon GameLift 容器叢集中最基本的元素。您的容器包括遊戲伺服器，以及其相依性，例如 SDK、軟體、目錄和檔案。

若要在容器叢集中運作，您的遊戲伺服器必須在 Linux 上執行，並與伺服器 SDK 5.x 整合。

主題

- 設定您的工作目錄
- 建立容器映像檔
- 將您的容器映像推送到 Amazon ECR

設定您的工作目錄

您的工作目錄是放置構建容器映像所需的所有文件並定義 Amazon 如何 GameLift 運行它的位置。

若要設定容器工作目錄

1. 建立您要使用 Amazon GameLift 容器映像的目錄。

   Example

   例如：

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .
.
./glc
```

```
./glc/gamebuild
```

2. 克隆 Amazon GameLift 代理。

   Example

   例如：

   ```
   [~/work]$ git clone https://github.com/aws/amazon-gamelift-agent.git
   Cloning into 'amazon-gamelift-agent'...
   ```

3. 建立使 GameLiftAgent 用 Maven 的。

   Example

   例如：

   ```
   [~/work]$ cd amazon-gamelift-agent
   ```

   Example

   ```
   [~/work/amazon-gamelift-agent]$ mvn clean compile assembly:single && \
   mv target ../glc && cd .. && find glc
   ```

4. 新增與伺服器 SDK 5.x 整合的遊戲伺服器，並建立並封裝到.ZIP檔案中。
5. 將您的.ZIP檔案複製到~/work/glc/gamebuild/.

如果您沒有 SDK 5.x 遊戲伺服器，您可以下載並使用我們的範例SimpleServer遊戲來嘗試使用容器叢集。

Example

```
[~/work]$ curl -o glc/gamebuild/SimpleServer.zip \
'https://ws-assets-prod-iad-r-iad-ed304a55c2ca1aee.s3.us-
east-1.amazonaws.com/086bb355-4fdc-4e63-8ca7-af7cfc45d4f2/
AmazonGameLiftSampleServerBinary.zip' &&
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
Dload  Upload   Total   Spent    Left  Speed
100 5140k  100 5140k    0     0  12.3M      0 --:--:-- --:--:-- --:--:-- 12.3M
glc
glc/target
```

```
glc/target/GameLiftAgent-1.0.jar
glc/gamebuild
glc/gamebuild/SimpleServer.zip
```

建立容器映像檔

您的 Docker 文件指定了構建容器的環境，軟件和指令。

建立您的碼頭檔

1. 移至glc子目錄。

   Example

   ```
   [~/work]$ cd glc && find

   .
   ./target
   ./target/GameLiftAgent-1.0.jar
   ./gamebuild
   ```

2. 創建並打開一個新的碼頭文件。

   Example

   例如：

   ```
   [~/work/glc]$ nano Dockerfile
   ```

3. 從以下模板之一複製，然後將內容粘貼到 Dockerfile 中。

碼頭文件模板為您的遊戲服務器

此範本包含容器在 Amazon GameLift 叢集中使用所需的最低指令。視需要修改遊戲伺服器的內容。

```
# Base image
# ----------
  # Add the base image that you want to use over here,
  # Make sure to use an image with the same architecture as the
  # Instance type you are planning to use on your fleets.
  # We require JDK to be installed in the base image, so that
  # it can be used to run the &AGS; Agent
```

```
FROM public.ecr.aws/amazoncorretto/amazoncorretto:17-amd64
  #
# Game build directory
# --------------------
  # Add your game build to gamebuild directory and add the zip file name in the
 'GAME_BUILD_ZIP' env variable below.
  # The game build provided over here needs to be integrated with gamelift server sdk.
  # This template assumes that the game build is in a zip format.
ENV GAME_BUILD_ZIP="<ADD_GAME_BUILD_ZIP_FILE_NAME>" \
  #
# Default directory
# -----------------
  # Default directory, the value provided here should be where the game executable
 exists.
  # Provide this same value as your launch path in RuntimeConfiguration when creating a
 fleet.
  # Ref: https://docs.aws.amazon.com/gamelift/latest/apireference/
API_ServerProcess.html
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME BUILD>" \
HOME_DIR="/local/game" \
  #
  # Registered compute in anywhere fleet (not used in container fleets)
  # ---------------------------------------------------------------------
  # Add the name for the registered compute in an anywhere fleet.
  # This environment variable is required only for anywhere fleets, but not for
 container fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
  GAMELIFT_COMPUTE_NAME="<ADD_COMPUTE_NAME>" \
  #
# Default Gamelift Agent jar
# --------------------------
GAMELIFT_AGENT_EXEC="GameLiftAgent-1.0.jar" \
  #
  # This env variable defines the name of the S3 bucket that stores the GameLift Agent
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
GAMELIFT_AGENT_LOGS_BUCKET_NAME="<ADD NAME OF GAMELIFT AGENT LOGS S3 BUCKET>" \
  #
# -----------
```

```
  # This env variable defines the name of the S3 bucket that stores the game session
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
# -----------
GAME_SESSION_LOGS_BUCKET_NAME="<ADD NAME OF GAME SESSION LOGS S3 BUCKET>" \
#
# -----------
GAMELIFT_AGENT_LOGS_PATH="/local/game/agentlogs/" \
  #
# NOT USED in container fleets - USED in Anywhere fleets
# ------------------------------------------------------------
  # Specifiy the type of compute resource used to host the game servers.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
COMPUTE_TYPE="ANYWHERE" \
  #
  # Specify the credential to be used for creating the client.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
CREDENTIAL_PROVIDER="environment-variable"

USER root

    # intall dependencies as necessary
    RUN yum install -y sudo \
                        unzip \
                        git \
                        shadow-utils \
                        iputils \
                        tar \
                        gcc \
                        make \
                        openssl-devel \
                        zlib-devel \
```

```
                        vim \
                        net-tools \
                        nc \
                        procps

    # Set up the ground for 'gamescale' user
    RUN groupadd -r gamescale -g 500 && \
      useradd -u 500 -r -g gamescale -m -s /sbin/nologin -c "Gamescale user" gamescale
&& \
      echo "gamescale ALL=(ALL) NOPASSWD: ALL" | (EDITOR="tee -a" visudo) && \
      mkdir -p $HOME_DIR && \
      mkdir $HOME_DIR/mono && \
      chown -R gamescale:gamescale $HOME_DIR

    WORKDIR $HOME_DIR

    # extract game build as necessary
    COPY ./gamebuild/$GAME_BUILD_ZIP .
    RUN unzip ./$GAME_BUILD_ZIP -d ./

    # copy Gamelift Agent jar
    COPY ./gameliftAgent/$GAMELIFT_AGENT_EXEC ./

    # Add permissions to game build and gamelift agent jar
    RUN chmod +x ./$GAME_EXECUTABLE
    RUN chmod +x ./$GAMELIFT_AGENT_EXEC

    # Check if java is installed on the image, if not then the Agent will not be able
to run
    RUN java --version

    USER gamescale

    ENV PATH="$PATH:$HOME_DIR/bin:$JAVA_HOME"

    # Change directory to bin
    WORKDIR $HOME_DIR

    # check path before starting the container
    RUN echo $PATH

    # Create logs directory for GameLift Agent & server processes
    RUN mkdir logs
    RUN mkdir agentlogs
```

```
    # Start the GameLift Agent
    ENTRYPOINT sleep 90 && java -jar $GAMELIFT_AGENT_EXEC -ip "192.168.1.1" -gslb
 "$GAME_SESSION_LOGS_BUCKET_NAME" -galb "$GAMELIFT_AGENT_LOGS_BUCKET_NAME" -galp
 "$GAMELIFT_AGENT_LOGS_PATH" -glc environment-variable
```

碼頭文件的示例 **SimpleServer**

```
# Base image
# ----------
  # Add the base image that you want to use over here,
  # Make sure to use an image with the same architecture as the
  # Instance type you are planning to use on your fleets.
  # We require JDK to be installed in the base image, so that
  # it can be used to run the &AGS; Agent
FROM public.ecr.aws/amazoncorretto/amazoncorretto:17-amd64
  #
# Game build directory
# --------------------
  # Add your game build to gamebuild directory and add the zip file name in the
 'GAME_BUILD_ZIP' env variable below.
  # The game build provided over here needs to be integrated with gamelift server sdk.
  # This template assumes that the game build is in a zip format.
ENV GAME_BUILD_ZIP="SimpleServer.zip" \
  #
# Default directory
# -----------------
  # Default directory, the value provided here should be where the game executable
 exists.
  # Provide this same value as your launch path in RuntimeConfiguration when creating a
 fleet.
  # Ref: https://docs.aws.amazon.com/gamelift/latest/apireference/
API_ServerProcess.html
GAME_EXECUTABLE="GameLiftSampleServer" \
HOME_DIR="/local/game" \
  #
  # Registered compute in anywhere fleet (not used in container fleets)
  # ------------------------------------------------------------------
  # Add the name for the registered compute in an anywhere fleet.
  # This environment variable is required only for anywhere fleets, but not for
 container fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
```

```
  GAMELIFT_COMPUTE_NAME="<ADD_COMPUTE_NAME>" \
  #
# Default Gamelift Agent jar
# -------------------------
GAMELIFT_AGENT_EXEC="GameLiftAgent-1.0.jar" \
  #
  # This env variable defines the name of the S3 bucket that stores the GameLift Agent
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
GAMELIFT_AGENT_LOGS_BUCKET_NAME="<ADD NAME OF GAMELIFT AGENT LOGS S3 BUCKET>" \
  #
# -----------
  # This env variable defines the name of the S3 bucket that stores the game session
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
# -----------
GAME_SESSION_LOGS_BUCKET_NAME="<ADD NAME OF GAME SESSION LOGS S3 BUCKET>" \
#
# -----------
GAMELIFT_AGENT_LOGS_PATH="/local/game/agentlogs/" \
  #
# NOT USED in container fleets - USED in Anywhere fleets
# ------------------------------------------------------
  # Specifiy the type of compute resource used to host the game servers.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
COMPUTE_TYPE="ANYWHERE" \
  #
  # Specify the credential to be used for creating the client.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
```

```
# -----------
CREDENTIAL_PROVIDER="environment-variable"

USER root

    # intall dependencies as necessary
    RUN yum install -y sudo \
                        unzip \
                        git \
                        shadow-utils \
                        iputils \
                        tar \
                        gcc \
                        make \
                        openssl-devel \
                        zlib-devel \
                        vim \
                        net-tools \
                        nc \
                        procps

    # Set up the ground for 'gamescale' user
    RUN groupadd -r gamescale -g 500 && \
      useradd -u 500 -r -g gamescale -m -s /sbin/nologin -c "Gamescale user" gamescale
 && \
      echo "gamescale ALL=(ALL) NOPASSWD: ALL" | (EDITOR="tee -a" visudo) && \
      mkdir -p $HOME_DIR && \
      mkdir $HOME_DIR/mono && \
      chown -R gamescale:gamescale $HOME_DIR

    WORKDIR $HOME_DIR

    # extract game build as necessary
    COPY ./gamebuild/$GAME_BUILD_ZIP .
    RUN unzip ./$GAME_BUILD_ZIP -d ./

    # copy Gamelift Agent jar
    COPY ./target/$GAMELIFT_AGENT_EXEC ./

    # Add permissions to game build and gamelift agent jar
    RUN chmod +x ./$GAME_EXECUTABLE
    RUN chmod +x ./$GAMELIFT_AGENT_EXEC
```

```
    # Check if java is installed on the image, if not then the Agent will not be able
to run
    RUN java --version

    USER gamescale

    ENV PATH "$PATH:$HOME_DIR/bin:$JAVA_HOME"

    # Change directory to bin
    WORKDIR $HOME_DIR

    # check path before starting the container
    RUN echo $PATH

    # Create logs directory for GameLift Agent & server processes
    RUN mkdir logs
    RUN mkdir agentlogs

    # Start the GameLift Agent
    ENTRYPOINT sleep 90 && java -jar $GAMELIFT_AGENT_EXEC -ip "192.168.1.1" -gslb
"$GAME_SESSION_LOGS_BUCKET_NAME" -galb "$GAMELIFT_AGENT_LOGS_BUCKET_NAME" -galp
"$GAMELIFT_AGENT_LOGS_PATH" -glc environment-variable
```

> **Note**
>
> 注意：Docker 檔案中的一些環境變數可以由. ContainerDefinition

若要建立容器映像檔

1. 建立您的容器映像檔。

   如果您使用自己的 SDK 5.x 服務器

   您可以指定所需的任何本地存儲庫名稱。

   Example

   ```
   [~/work/glc]$ docker build -t <local repository name>:<optional tag> .
   ```

如果您正在使用我們的**SimpleServer**樣品

Example

```
[~/work/glc]$ docker build -t simple-server:version-1 .
Successfully built 0123456789012
Successfully tagged simple-server:version-1
```

> **Note**
>
> 在下面的例子中，我們使用#######作為初始REPOSITORY值，並version-1作為值。TAG

2. 檢視影像清單，並記下REPOSITORY和IMAGE ID值。您將在下面的過程中需要它們。

Example

```
[~/work/glc]$ docker images
REPOSITORY       TAG          IMAGE ID        CREATED           SIZE
simple-server    version-1    0123456789012    14 minutes ago    1.24GB
```

將您的容器映像推送到 Amazon ECR

將您的容器映像上傳到 Amazon ECR 中的私有儲存庫。建立容器群組定義時，您會參考此儲存庫位置，以便 Amazon GameLift 可以拍攝容器映像的快照，並在部署容器叢集時使用它。

> **Note**
>
> 如果您還沒有 Amazon ECR 私有儲存庫，請建立一個。

獲取您的 Amazon ECR 憑據

- 在將容器映像推送至 Amazon ECR 之前，您必須先以臨時形式取得 AWS 登入資料並將其提供給 Docker。取得您的 Amazon ECR 登入資料，以便碼頭工人可以登入。

Example

```
[~/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
```

將您的容器映像推送到 Amazon ECR

1. 複製您要使用的 Amazon ECR 私有儲存庫的 URI。
2. 將 Amazon ECR 標籤套用至您的容器映像檔。

   Example

   ```
   [~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository
    URI>:<optional tag>
   ```
3. 將您的容器映像推送到 Amazon ECR

   Example

   ```
   [~/work/glc]$ docker image push  <Amazon ECR private repository URI>
   ```

設計 Amazon GameLift 貨櫃車隊

本文件適用於公開預覽版本中的功能。內容可能變動。

這些主題說明您在設定 Amazon GameLift 容器叢集時會做出的重要決定。您的決策會影響您設定容器、容器群組和叢集的設定方式。

主題

- 建構您的車隊貨櫃結構
- 設定資源限制

- 指定必要的容器
- 設定網路連線
- 設定容器的健康狀態檢查
- 設定容器相依性
- 設定容器叢集

建構您的車隊貨櫃結構

第一步，請確定託管遊戲伺服器所需的軟體和資源，包括：

- 您的遊戲伺服器應用程式 該應用程序必須與 Amazon GameLift 功能集成以進行託管，包括服務器 SDK 版本 5 +。請參閱將您的遊戲與 Amazon 整合 GameLift。
- Amazon GameLift 代理。這個運算上的代理程式會維持與 Amazon GameLift 服務的通訊，並管理所有遊戲伺服器程序的生命週期。如需詳細資訊，請參閱遊戲伺服器和 Amazon GameLift 代理。
- 視需要提供其他軟體和資源。這可能包括執行遊戲伺服器應用程式所需的軟體。常見的支援軟體可用於記錄和監控、安全性、內容交付和資料同步。

接下來，決定如何為 Amazon GameLift 容器叢集建構軟體和資源。Amazon GameLift 使用容器群組來組織容器。叢集永遠有一個複本容器群組，而且可以選擇性地擁有精靈容器叢集。如需詳細資訊，請參閱貨櫃車隊元件。

- 從設計複本容器群組開始。請考量下列準則：
  - 將您的遊戲伺服器應用程式和 Amazon GameLift 代理程式捆綁到同一個容器中。將此容器設為複本群組的唯一必要容器。
  - 將遊戲伺服器的所有其他軟體整理到容器中。您可以選擇將所有內容放入複本群組中的單一容器中。或者，您可以選擇創建一個或多個附屬容器。使用側車的一些原因包括：
    - 為個別軟體設定啟動/關閉順序。您可以通過將軟件放在單獨的容器中並設置它們之間的依賴關係來實現此目的。
    - 設定容器特定的記憶體和 CPU 使用量限制。
    - 為每個容器指定不同的容器組態設定，例如啟動命令、進入點、工作目錄、環境變數或健康狀態檢查。
- 決定是否需要叢集的精靈容器群組。考慮下列各項：
  - 守護程式容器通常用於執行背景或監視程序。

- 精靈群組中的容器不會在叢集執行個體上複寫。這表示精靈群組中的容器不會隨著複本容器群組一起擴充。
- 精靈群組可以有多個容器。您可以將精靈群組中的任何容器指定為必要的容器。

設定資源限制

針對每個容器群組，判斷群組執行其軟體所需的記憶體和 CPU 數量。Amazon GameLift 依賴此資訊來管理容器群組的資源。它也會使用此資訊來計算叢集映像可以容納多少個複本容器群組。您也可以設定個別容器的限制。

設定容器的選擇性限制

設定容器特定的資源限制可讓您更好地控制個別容器如何使用群組資源。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。共享提供了更大的靈活性，可以在需要的地方使用資源。它還增加了進程相互競爭並導致容器故障的可能性。

為任何容器設定下列任一`ContainerDefinition`屬性。

- `SoftLimit`(記憶體) — 保留最低容量的記憶體供容器專用使用。容器一律具有可用的保留金額。它可以在任何時候超過這個最低限度，如果有其他資源可用。
- `HardLimit`(記憶體) — 設定容器的最大記憶體限制。如果容器超過此限制，則會導致重新啟動。
- Cpulimit — 保留最少數量的 CPU 資源，以供容器專用使用。容器一律具有可用的保留金額。它可以在任何時候超過這個最低限度，如果有其他資源可用。(1024 個 CPU 單位相當於 1 個 vCPU。)

設定容器群組的資源總計限制

告訴 Amazon GameLift 每個容器群組需要多少記憶體和 CPU 資源。目標是分配足夠的資源來最佳化遊戲伺服器效能。Amazon GameLift 使用這些限制來計算如何在叢集執行個體上包裝複本容器群組。您也可以在為容器叢集選擇執行個體類型時使用它們。

計算一組中每個容器中所有進程所需的總內存和 CPU。考慮下列各項：

- 在容器群組中的所有容器中執行哪些程序？ 將這些程序所需的資源加起來。
- 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？ 您將此值設定為叢集的執行階段組態的一部分，但您必須在此處為它們規劃足夠的記憶體 (請參閱最佳化執行階段組態)。

根據您預估的容器群組需求，設定下列`ContainerGroupDefinition`屬性：

- `TotalMemoryLimit`— 設定容器群組的最大記憶體限制。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，總記憶體限制必須為：
  - 等於或大於所有容器軟性記憶體限制的總和
  - 等於或大於群組中容器的最高硬碟記憶體限制
- `TotalCpuLimit` — 設定容器群組的最大 CPU 限制。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，CPU 總限制必須為：
  - 等於或大於所有容器 CPU 限制的總和。最佳作法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以下三個容器的複本容器群組：

- 容器 A 是我們必不可少的複製品容器。它運行遊戲服務器進程和 Amazon GameLift 代理。我們估計一部遊戲伺服器的資源需求為 512 MiB 和 1024 處理器。我們計劃讓容器運行 10 個服務器進程。由於此容器執行我們最重要的軟體，因此我們設定了 6144 MiB 的軟體記憶體保留，而且沒有硬碟記憶體限制或 CPU 保留限制。
- 容器 B 執行的資源需求估計為 1024 MiB 和 1536 個 CPU 的支援軟體。我們將軟體記憶體保留限制設定為 1024 MiB、硬碟記憶體限制為 2048 MiB，以及 1024 個 CPU 的 CPU 保留限制。
- 容器 C 會執行非關鍵記錄和其他監視公用程式。我們設置了 512 MiB 的硬存儲器限制和 512 CPU 的 CPU 保留限制。

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 (1) 軟體記憶體限制 (6144+1024 MiB) 和 (2) 最高硬碟記憶體限制 (1024 MiB) 的總和。
- 中央處理器總限制：13312 中央處理器。這個值超過 CPU 限制的總和 (1024+512 中央處理器)。

指定必要的容器

對於每個容器，將容器指定為必要或非必要容器。所有容器群組都必須至少有一個必要容器。基本容器會執行容器群組的關鍵工作，例如託管您的遊戲伺服器。必要的容器始終預計將運行。如果失敗，整個容器群組就會重新啟動。

- 您的叢集複本容器群組只能有一個基本容器。此容器會執行 Amazon GameLift 代理程式及其管理的遊戲伺服器處理程序。
- 如果您的叢集有精靈容器群組，您可以指定多個基本容器。如果您希望容器失敗提示容器群組重新啟動，請將精靈容器設為必要項目。

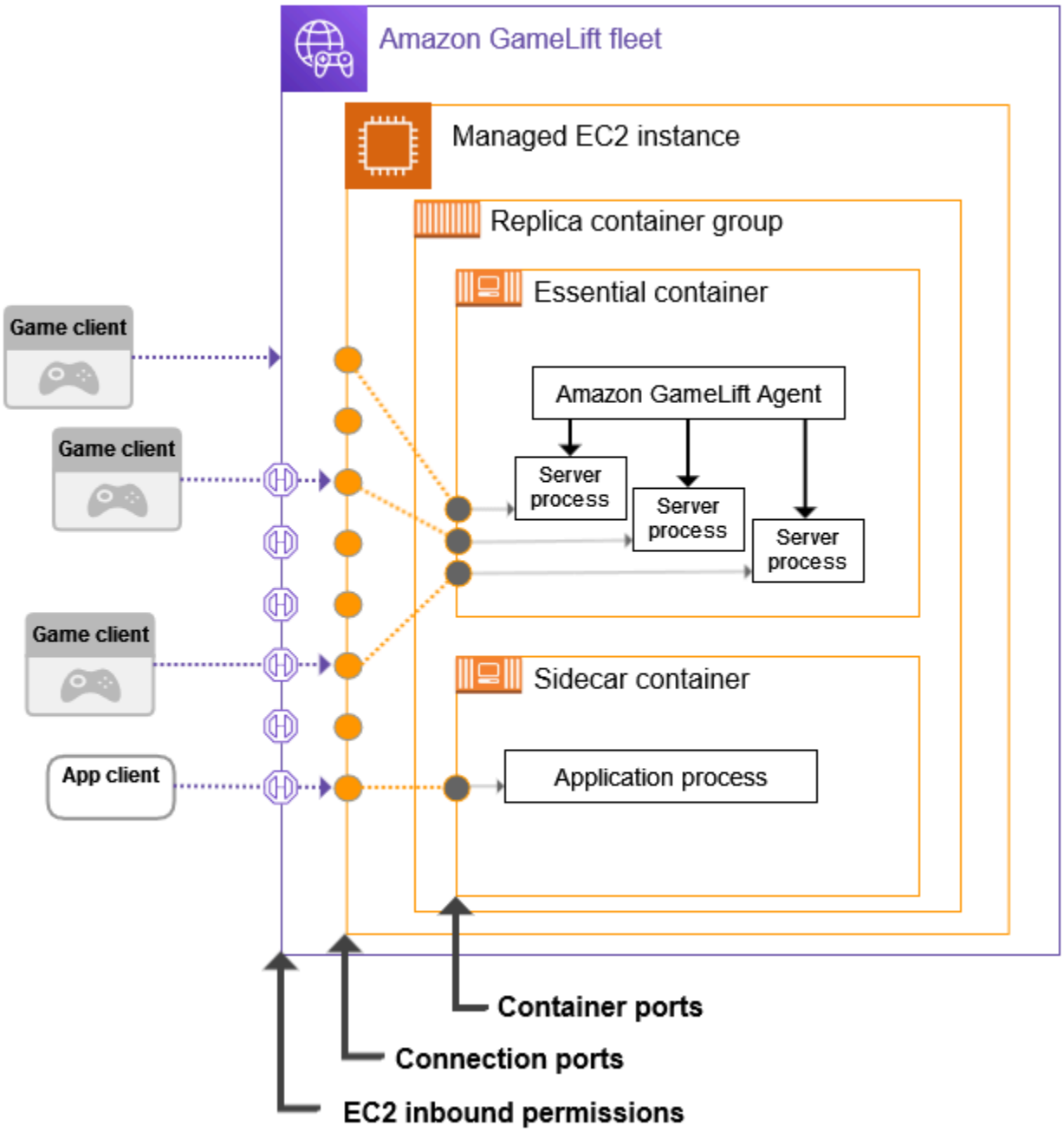
Essential將每個容器的ContainerDefinition屬性設定為真或假。

設定網路連線

您可以建立網路存取權，讓外部流量連線至容器叢集中的任何容器。例如，您必須與執行遊戲伺服器程序的容器建立網路連線，以便遊戲用戶端可以加入並暢玩您的遊戲。遊戲客戶端使用端口和 IP 地址連接到遊戲服務器。

在容器叢集中，用戶端與伺服器之間的連線不是直接的。在內部，容器中的處理序會偵聽容器連接埠。在外部，傳入流量會使用連線連接埠連線至叢集執行個體。Amazon 會 GameLift 維護內部容器連接埠與外部連線連接埠之間的對應，以便傳入流量路由到執行個體上的正確程序。

Amazon 為您的網絡連接 GameLift 提供了一層額外的控制。每個容器叢集都有輸入權限設定，可讓您控制每個對外連線連接埠的存取。您無法變更現有叢集的連接埠設定，但可以透過調整輸入權限，視需要允許或限制存取。例如，您可以移除所有連線連接埠的權限，以關閉對叢集容器的所有存取。



設定容器連接埠範圍

為任何需要外部存取的程序設定具有足夠容器連接埠的容器定義。有些容器不需要任何連接埠。其他人必須擁有足夠的連接埠，才能將一個連接埠指派給每個需要一個

執行遊戲伺服器的基本複本容器群組需要為每個同時執行的遊戲伺服器程序使用連接埠 (如叢集中的設定RuntimeConfiguration)。遊戲伺服器程序會監聽指派的連接埠，並將其報告給 Amazon GameLift。

建立容器群組定義時，請為每個需要網路存取的容器定義容器連接埠範圍 (請參閱 ContainerDefinitionInput：PortConfiguration)。確保範圍足夠大，可以為每個需要一個進程分配一個端口。處理序必須在容器的連接埠組態中指派連接埠號碼。

設定連接埠範圍

使用一組連接埠設定您的容器叢集。連線連接埠可讓您對執行容器的叢集執行個體提供外部存取權。Amazon GameLift 會根據需要指派連接埠，並將其對應至容器連接埠。

建立容器叢集時，請定義連線連接埠範圍 (請參閱 ContainerGroupsConfiguration：ConnectionPortRange)。確定範圍具有足夠的連接埠，可對應至叢集執行個體中的每個容器連接埠。若要計算所需的最小連接埠，請使用下列公式：

```
[Total number of container ports defined for containers in the replica container group] * [Number of replica container groups per instance] + [Total number of container ports defined for containers in the daemon container group]
```

最佳作法是將最小連接埠數目加倍。

> **Note**
>
> connecton 連接埠的數目可能會限制每個執行個體的複本容器群組數目。如果叢集只有足夠的連線連接埠可容納每個執行個體一個複本容器群組，Amazon 只 GameLift 會部署一個複本容器群組，即使執行個體具有足夠的運算能力供多個複本容器群組使用。

設定輸入權限

輸入權限可指定要針對內送流量開啟的連線連接埠，藉此控制對容器叢集的外部存取。您可以使用此設定，視需要開啟或關閉叢集的網路存取。

建立容器叢集時，請定義一組輸入許可 (請參閱：EC2 CreateFleet)。InboundPermissions設定輸入權限連接埠內容，以在叢集的連線連接埠設定中包含部分或所有值。若要變更現有容器叢集的輸入權限，請撥打 UpdateFleetPortSettings。

範例藍本

此範例說明如何設定所有三個網路連線屬性。

- 我們艦隊的複製容器組有 1 個容器，用於運行遊戲服務器進程。執行階段確認會告訴容器執行 10 個並行遊戲伺服器處理序。

  在複本容器群組定義中，我們設定此容器的PortConfiguration參數，如下所示：

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP"} ]
  }
```

- 我們的車隊也有一個守護進程容器組與 1 個容器。它有 1 個需要網絡訪問的進程。在守護進程容器組定義中，我們為此容器設置PortConfiguration參數，如下所示：

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ]  }
```

- 我們的機隊配置了每個車隊實例 3 副本容器組。有了這些信息，我們可以使用公式來計算我們需要的連接端口數量：
  - 最少：31 個連接埠 [10 個複本容器連接埠 * 每個執行個體 3 個複本容器群組 + 1 個精靈容器連接埠]
  - 最佳做法：62 個連接埠 [最少連接埠 * 2]

  建立容器叢集時，我們在中設定ConnectionPortRange參數，ContainerGroupsConfiguration如下所示：

```
"ConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- 我們希望允許訪問所有可用的連接端口。建立容器叢集時，我們設定EC2InboundPermissions參數如下：

```
"EC2InboundPermissions": [
  {"FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":
 "TCP"} ]
```

設定容器的健康狀態檢查

如果容器遇到終端機故障並停止運行，則會自動重新啟動。如果容器是必要的，整個容器群組都會重新啟動。

您可以定義其他自訂條件來測量容器健全狀況，並使用健康狀態檢查來測試該條件。若要設定容器健康狀態檢查，您可以在 docker 容器映像檔或容器定義中定義它。如果您在容器定義中設定健康狀態檢查，它會覆寫容器映像中的任何設定。

根據容器類型設定選擇性的健全狀況檢查，如下所示：

- 對於基本複本容器，請勿設定健康狀態檢查。Amazon GameLift 代理程式會自動處理此容器的運作狀態報告。
- 對於非必要的複本容器和任何精靈容器，您可以選擇性地設定健康狀態檢查參數。

設定容器健`ContainerDefinition`全狀況檢查的下列屬性：

- `Command`— 提供檢查容器健康狀況某些方面的命令。您可以決定用什麼標準來衡量健康狀況。指令必須導致結束值為 1 (狀況不良) 或 0 (狀況良好)。
- `StartPeriod`— 指定健全狀況檢查失敗開始計數之前的初始延遲。這種延遲使容器有時間來引導其進程。
- `Interval`— 決定執行健康狀態檢查命令的頻率。您希望偵測和解決容器故障的速度有多快？
- `Timeout`— 決定在重試健康狀態檢查命令之前等待成功或失敗的時間長度。健康狀態檢查命令需要多長時間才能完成？
- `Retries`— 在註冊失敗之前，健康狀態檢查命令應重試多少次？

設定容器相依性

在每個容器群組中，您可以根據容器狀態設定容器之間的相依性。相依容器可以根據另一個容器的狀態啟動或關閉時，相依性會受到影響。

相依性的主要使用案例是建立容器群組的啟動和關閉順序。

例如，您可能希望容器 A 在容器 B 和 C 啟動之前先啟動並成功完成。若要達成此目的，請先為容器 A 上的容器 B 建立相依性，並符合容器 A 必須順利完成的條件。然後使用相同條件為容器 A 上的容器 C 建立相依性。啟動順序會以相反的順序進行關機。

設定容器叢集

建立容器叢集時，請考量下列決策點。這些點大部分都取決於您的容器架構和配置。

決定您要在哪裡部署您的車隊

通常，您希望將車隊部署在玩家附近的地理位置，以最大程度地減少延遲。您可以將您的容器叢集部署到任何 Amazon GameLift 支援 AWS 區域 的每個項目。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叢集，包括 AWS 區域 和 Local Zones。對於多地點叢集，您可以在每個叢集位置獨立調整容量。如需支援的叢集位置的詳細資訊，請參閱Amazon GameLift 託管地點。

選擇叢集的執行個體類型和大小

Amazon GameLift 支援各種 Amazon EC2 執行個體類型，所有這些類型都可以與容器叢集搭配使用。執行個體類型的可用性和價格因地點而異。您可以在 Amazon GameLift 主控台 (在資源、執行個體和服務配額下) 檢視受支援的執行個體類型清單，並按位置篩選。

選擇例證類型時，首先要考慮例證族群。執行個體系列提供 CPU、記憶體、儲存和網路功能的各種組合。取得 EC2 執行個體系列的詳細資訊。在每個族群中，您都有一系列例證大小可供選擇。選取例證大小時，請考慮下列問題：

- 支援工作負載的最小執行個體大小是多少？ 使用此資訊可消除任何太小的執行個體類型。
- 哪些執行個體類型大小適合您的容器架構？ 理想情況下，您想要選擇可容納多個複本容器群組副本的大小，而浪費空間最少。
- 什麼縮放粒度對您的遊戲有意義？ 擴展叢集容量涉及新增或移除執行個體，而每個執行個體代表能夠託管特定數量的遊戲工作階段。考慮每個執行個體要新增或移除多少容量。如果玩家需求每分鐘都有數千個變化，那麼使用可以託管數百或數千個遊戲工作階段的大型執行個體可能是有意義的。相較之下，您可能更喜歡使用較小的執行個體類型進行更精細的縮放控制。
- 是否可以根據大小節省成本？ 您可能會發現某些執行個體類型的費用會因可用性而有所不同。

優化您的運行時配置

叢集的執行階段設定是一組指示，說明如何為遊戲工作階段主控執行伺服器處理序。這些指示由 Amazon GameLift 代理程式在叢集中的每個複本容器群組中實作。

叢集的執行階段設定會決定每個複本容器群組中同時執行的伺服器處理序數目。此設定會影響您計算容器群組資源限制的方式，以及為叢集選擇執行個體類型的方式。在設計艦隊時，您需要平衡這三個要素。

如需如何使用執行階段設定的詳細資訊，請參閱管理 Amazon 如何GameLift啟動遊戲伺服器。

設定其他選擇性叢集設定

設定容器叢集時，您可以使用下列選用功能：

- 設定您的遊戲伺服器以存取其他 AWS 資源。請參閱與您車隊的其他AWS資源進行溝通。
- 保護與活躍玩家的遊戲階段，避免在縮小活動期間過早終止。
- 在有限的時間範圍內，限制一個人可以在艦隊上建立的遊戲工作階段數量。

為 Amazon 容器叢集建立 GameLift 容器群組定義

本文件適用於公開預覽版本中的功能。內容可能變動。

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叢集。這是一個藍圖，可識別要在叢集上執行的容器集，以及如何執行它們。建立容器叢集時，您可以指定要部署至叢集的容器群組定義。如需容器群組的詳細資訊，請參閱貨櫃車隊元件。

開始之前

完成下列任務：

- 設計用於託管遊戲伺服器的容器架構。請參閱設計 Amazon GameLift 貨櫃車隊。
- 規劃要包含在容器群組中的容器定義。如果您使用 AWS CLI，請在 JSON 檔案中建立容器定義。
- 將最終容器映像推送至您打算建立容器群組的 Amazon 彈性容器登錄檔 (Amazon ECR) 登錄。AWS 區域 Amazon 會在您建立容器群組定義時 GameLift 存放每個映像的快照，並在部署到容器叢集時使用該副本。請參閱使用您的遊戲伺服器軟體準備容器映像檔。
- 確認您的 AWS 使用者具有存取權管理權限以存取 Amazon ECR 儲存庫。請參閱管理 Amazon 的用戶許可 GameLift。您至少需要下列動作的權限：
  - `ecr:DescribeImages`
  - `ecr:BatchGetImage`
  - `ecr:GetDownloadUrlForLayer`

複製容器群組定義

您可以使用 Amazon GameLift 主控台複製現有的容器群組定義。

複製容器群組

1. 在 Amazon 主 GameLift 控台中，移至左側導覽窗格，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。選取容器群組之後，[複製] 按鈕就會處於作用中狀態。
3. 選擇複製。此動作會以預先填入的設定開啟容器群組建立精靈。
4. 輸入複製的容器群組的新名稱。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、檢閱並建立新的容器群組。

建立複本容器群組定義

複本容器群組會管理您的遊戲伺服器軟體。複本容器群組至少有一個執行 Amazon GameLift 代理程式和遊戲伺服器程序的容器。群組可能有額外的「附屬」容器來執行支援軟體。

本主題說明如何使用 Amazon 主 GameLift 控台或 AWS CLI 工具建立容器群組定義。如需有關設定容器群組組態的詳細資訊，請參閱設計 Amazon GameLift 貨櫃車隊。

Console

在 Amazon 主 GameLift 控台中，選取您 AWS 區域 要建立容器群組的位置。

開啟主控台左側的導覽列，然後選擇 [容器群組]。在 [容器群組] 頁面上，選擇 [建立容器群組]。

步驟 1：定義群組詳細資訊。

1. 輸入貨櫃群組定義名稱。此名稱對於「區域」 AWS 帳戶 而言必須是唯一的。在主控台中，群組定義會依名稱列出，因此指派有意義的標籤會很有幫助。
2. 選取複本排程策略。
3. 在總記憶體限制中，輸入容器群組可用的最大記憶體。如需計算此值的說明，請參閱設定資源限制。
4. 在「CPU 總計限制」中，輸入容器群組可用的最大運算能力。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。

使用您的遊戲伺服器應用程式和 Amazon GameLift 代理程式定義容器。這是您必不可少的複本容器。

1. 提供容器定義 「名稱」。為群組定義的每個容器都必須具有唯一的名稱值。
2. 識別容器映像檔的 Amazon ECR 映像 URI。輸入下列任一格式：

- 僅限影像 URI：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]
- 圖片 URI + 摘要：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]@[digest]
- 圖片編號 + 標籤：[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]:[tag]

3. 對於 「基本」容器，會針對第一個容器定義自動選取「是」。如果您新增其他容器定義，您可以針對每個定義切換此設定為開啟或關閉。如需詳細資訊，請參閱指定必要的容器。
4. 設定一或多個內部容器連接埠範圍。此容器代管您的遊戲伺服器，因此請定義具有足夠連接埠的範圍，讓每個伺服器處理序在容器群組中執行。如需詳細資訊，請參閱設定網路連線。
5. 選用設定覆寫和環境變數可讓您指定要在啟動時傳遞至容器的值。您在此設定的值會覆寫容器映像檔中已有的任何設定。
6. 設定選擇性容器限制，以管理此容器的資源配置。如需詳細資訊，請參閱設定資源限制。
7. 視需要定義其他非必要貨櫃：

   - 提供容器定義名稱和 ECR 映像 URI。非必要容器不得執行 Amazon GameLift 代理程式。
   - 只有在容器具有需要網路存取的程序時，才設定內部容器連接埠範圍。
   - 選擇性地設定容器的 Health 檢查。當非必要容器未通過健康狀態檢查時，它只會提示重新啟動失敗的容器。
   - 視需要選擇性設定 「覆寫」、「環境變數」及「資源配置限制」。

步驟 3：設定相依性。

如果您的容器群組定義中有多個容器，您可以定義它們之間的相依性。使用相依性根據容器條件設定啟動和關閉順序。如需詳細資訊，請參閱設定容器相依性。

1. 識別您要為其新增相依性的容器名稱。這個容器不會啟動，直到依賴條件滿足。
2. 確定依賴容器名稱和條件。此容器必須符合條件，才能啟動相依容器。
3. 視需要設定其他相依性。您可以為任何容器建立多個相依性。避免建立循環相依性。

步驟 4：檢閱並建立。

1. 檢閱所有容器群組定義設定。您無法在建立容器群組定義之後變更它的組態。使用 「編輯」可變更任何區段，包括群組的每個容器定義。
2. 完成審核後，請選擇 [建立]。

如果您的要求成功，主控台會顯示新容器群組定義資源的詳細資訊頁面。最初的狀態是COPYING，因為 Amazon GameLift 開始為該組拍攝所有容器映像的快照。完成此階段時，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，您才能使用它建立容器叢集。

AWS CLI

當您使用 AWS CLI 建立容器群組定義時，請在不同的JSON檔案中維護容器定義組態。您可以在 CLI 命令中引用該文件。建立容器定義JSON檔如需架構範例，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 `create-container-group-definition` CLI 命令。如需有關此命令的詳細資訊，請參閱《AWS CLI 命令參考》create-container-group-definition中的〈〉。

Example ：複本容器群組

此範例說明複本容器群組定義的要求。建立複本和精靈群組定義的指令結構基本上是相同的。各個容器定義中描述了每種類型群組的特定詳細資訊。

此範例假設您已使用此群組的容器定義建立 JSON 檔案。

```
aws gamelift create-container-group-definition \
    --name MyAdventureGameContainerGroup \
    --operating-system AMAZON_LINUX_2023 \
    --scheduling-strategy REPLICA \
    --total-memory-limit 4096 \
    --total-cpu-limit 1024 \
    --container-definitions file://SimpleServer.json
```

建立容器定義JSON檔

建立容器群組定義時，也會定義群組的容器。容器定義會指定儲存容器映像的 Amazon ECR 儲存庫，以及網路連接埠的選用組態、CPU 和記憶體使用量限制以及其他設定。建議您建立單一JSON檔案，其中包含容器群組中所有容器的組態。維護文件對於存儲，共享和版本跟踪這些關鍵配置非常有用。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。

若要建立容器定義

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，並視需要為您的容器加以修改。如需容器定義語法的詳細資訊，請參閱 Amazon GameLift API 參考ContainerDefinitionInput中的。
3. 將檔案儲存在本機，以便您可以在 AWS CLI 指令中參考該檔案。

範例：基本複本容器定義

Example

此範例說明複本容器群組的基本容器。基本複本容器包括您的遊戲伺服器應用程式、Amazon GameLift Agent，並且可以包含用於遊戲託管的其他支援軟體。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": "SimpleServer",
    "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-containers:complex-
server",
    "Essential": true,
    "Cpu": 256,
    "MemoryLimits": {
      "HardLimit": 128
  },
     "PortConfiguration": {
            "ContainerPortRanges": [
               {
                  "FromPort": 2000,
                  "Protocol": "TCP",
                  "ToPort": 2100
               }
            ]
      }
    }
]
```

建立 Amazon GameLift 容器叢集

本文件適用於公開預覽版本中的功能。內容可能變動。

建立容器群組定義後，請使用 Amazon GameLift 主控台或 AWS Command Line Interface (AWS CLI) 建立容器叢集。

建立新叢集之後，Amazon 會將您的容器群組 GameLift部署到每個叢集執行個體並啟動遊戲伺服器時，叢集的狀態會經過數個階段。當艦隊達到狀態時ACTIVE，就可以主持遊戲工作階段了。如需建立叢集問題的協助，請參閱偵錯亞馬遜GameLift叢集問題。

Console

在 Amazon 主 GameLift 控台中，選取您 AWS 區域 要建立叢集的位置。容器群組定義必須位於您要建立叢集的相同區域中。

開啟主機左側導覽列，然後選擇 [叢集]。在 [叢集] 頁面上，選擇 [建立叢集]。

步驟 1：選擇運算類型

- 選擇容器運算類型。

步驟 2：定義車隊詳細資訊

1. 在 [叢集詳細資料] 區段中，輸入叢集名稱和說明。
2. 在 [容器群組詳細資料] 區段中，識別要部署至叢集的容器群組。您必須新增複本容器群組。您可以選擇性地新增精靈容器群組。每個群組都必須處於狀態READY。
3. 設定叢集的連線連接埠範圍。如需詳細資訊，請參閱設定網路連線。
4. 選擇性地指定要部署的每個執行個體所需的複本。您可以指定所需的數量，也可以讓 Amazon GameLift 計算可能的最大數量。如果您指定的所需數目大於計算的最大值，則叢集建立將會失敗。建立叢集之後，您無法變更此設定。如需複本容器群組封裝的詳細資訊，請參閱核心概念。
5. (選擇性) 在其他詳細資訊下：

    a. 針對執行個體角色，指定 IAM 角色，以授權遊戲組建中的應用程式存取帳戶中的其他 AWS 資源。如需詳細資訊，請參閱 與您車隊的其他AWS資源進行溝通。若要建立具有執行個體角色的叢集，您的帳戶必須具有 IAM PassRole 權限。如需詳細資訊，請參閱 亞馬遜的 IAM 許可示例 GameLift。

b. 針對測量結果群組，輸入新的或現有叢集測量結果群組的名稱。您可以將多個叢集的指標新增至相同的量度群組，來彙總這些指標。

步驟 3：定義實例詳細資訊

1. 在執行個體部署中，選取一或多個要部署執行個體的遠端位置。系統會自動選取主地區 (這是您建立叢集的地區)。如果您選取其他位置，叢集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 AWS 帳戶.
>
> - 您在 2022 年 2 月 28 日之前建立的未啟用區域的艦隊將不會受到影響。
> - 若要建立新的多位置叢集或更新現有的多位置叢集，請先啟用您選擇使用的任何區域。
>
> 如需有關預設未啟用的區域以及如何啟用的詳細資 AWS 區域訊，請參閱 AWS 一般參考.

2. 選取叢集的執行個體組態。主控台會自動計算所需的最小 vCPU 和記憶體 (根據您為每個容器群組設定的總限制)。它會根據資源需求與您輸入的位置來篩選可用執行環境類型的完整清單。您可以視需要新增其他篩選器。

   如需選擇執行個體類型的詳細資訊，請參閱設定容器叢集。您選擇的執行個體類型大小會影響將複本容器群組封裝到每個叢集執行個體的方式。根據您的選擇，請考慮針對每個執行個體所需的複本檢閱設定。

步驟 4：設定執行階段

執行階段設定會決定如何啟動和執行遊戲伺服器處理序。這些指示會傳遞給 Amazon GameLift 代理程式，該代理程式在每個複本容器群組中以相同的方式實作這些指示。您可以透過呼叫來更新叢集的執行階段設定UpdateRuntimeConfiguration。

1. 在「啟動路徑」中，輸入遊戲執行檔的路徑。
2. (選擇性) 對於 Launch 參數，請輸入要作為一組命令列參數傳遞至遊戲執行檔的資訊。

3. 指定要在每個複本容器群組中維持執行的並行處理作業數目。查看每個執行個體的伺服器處理序數目的 Amazon GameLift 配額。對每個執行個體之並行伺服器程序的限制適用於所有組態的並行程序總數。如果您將叢集設定為超出限制，則無法啟動叢集。
4. 設定同時啟動遊戲工作階段的選擇性限制。這些設定可讓您限制開始新遊戲工作階段時所耗用的資源量。遊戲工作階段啟動可能會影響現有遊戲工作階段的效能。
5. 設定 EC2 連接埠設定，以允許外部流量存取叢集上執行的程序。指定針對叢集定義的部分或所有連線連接埠號碼。您不必在建立叢集時設定這些連接埠，但沒有這些連接埠，就無法連線到您的遊戲伺服器。若要稍後更新叢集的連接埠設定，請撥打 UpdateFleetPortSettings
6. 在遊戲會話資源設置下配置以下可選功能：

   a. 開啟或關閉遊戲縮放保護政策。啟用防護後，如果執行個體託管作用中的遊戲工作階段，Amazon GameLift 不會在縮減事件期間關閉執行個體。

   b. 設定最大資源建立限制，以限制玩家在指定時間範圍內可建立的遊戲階段數量。

步驟 5：設定標籤

- (選擇性) 輸入「索引鍵」和「值」配對，將標籤新增至組建。選擇 [下一步] 繼續進行叢集建立檢閱。

步驟 6：檢閱並建立。

- 檢閱您的叢集組態設定。

  無論叢集狀態為何，您都可以隨時更新叢集的中繼資料和組態。如需詳細資訊，請參閱 管理您的亞馬遜GameLift車隊。您可以在叢集達到 ACTIVE 狀態後更新叢集容量。如需詳細資訊，請參閱 擴展亞馬遜GameLift託管容量。您也可以新增或移除遠端位置。

  完成審核後，請選擇 [建立]。

  如果您的要求成功，主控台會顯示新叢集資源的詳細資料頁面。最初的狀態是NEW，因為 Amazon GameLift 啟動了車隊創建過程。您可以在 Fleets (叢集) 頁面追蹤新叢集的狀態。艦隊已準備好在達到狀態時主持遊戲工作階段ACTIVE。

AWS CLI

若要使用建立容器叢集 AWS CLI，請開啟命令列視窗並使用`create-fleet`指令。若要取得有關此指令的更多資訊，請參閱《指AWS CLI 令參考》`create-fleet`中的。

以下顯示的範例`create-fleet`要求會建立具有下列特性的新容器叢集：

- ContainerGroupsConfiguration 指定單一複本容器群組定義：`MegaFrogRaceServer.NA.v2`。複本群組的三個副本將部署到每個叢集執行個體。每個執行個體都有 30 個連線連接埠，可供存取執行個體上的處理序。
- 叢集使用 c5. 大型隨需執行個體。
- 它會將容器群組部署到下列位置：
  - us-west-2（主場地區）
  - ca-central-1 (遠端位置)
- 執行個體上的每個複本容器群組都會同時執行 5 個遊戲伺服器程序，讓每個執行個體一次最多託管 15 個遊戲工作階段。
- 在每個複本容器群組中，Amazon GameLift 允許同時啟用兩個新遊戲工作階段。如果他們還沒有準備好在 300 秒內接待玩家，它也會終止任何啟動遊戲工作階段。
- 在這個機群中的執行個體代管的所有遊戲工作階段已開啟遊戲工作階段保護。
- 個別玩家可以在 15 分鐘期間內建立三個新遊戲工作階段。

```
aws gamelift create-fleet \
    --name SampleFleet123 \
    --description "The sample test fleet" \
    --compute-type "CONTAINER" \
    --container-groups-configuration
 "ContainerGroupDefinitionNames=['MegaFrogRaceServer.NA.v2'],
 DesiredReplicaContainerGroupPerInstance=3,
 ConnectionPortRange={FromPort=1010,ToPort=1040}" \
    --ec2-instance-type c5.large \
    --region us-west-2 \
    --locations "Location=ca-central-1" \
    --fleet-type ON_DEMAND \
    --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
 MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=/local/game/
MegaFrogRace/server.exe,ConcurrentExecutions=5}]" \
    --new-game-session-protection-policy "FullProtection" \
    --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
 PolicyPeriodInMinutes=15" \
    --ec2-inbound-permissions
 "FromPort=1010,ToPort=1040,IpRange=0.0.0.0/0,Protocol=UDP" \
```

如果建立叢集請求成功，Amazon 會 GameLift 傳回一組叢集屬性，其中包括您請求的組態設定和新的叢集 ID。Amazon GameLift 接著會將叢集狀態和位置狀態設定為 [新增]，並啟動叢集啟動程序。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-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

管理您的 Amazon GameLift 容器車隊

本文件適用於公開預覽版本中的功能。內容可能變動。

當您想要取得容器叢集的相關資訊或進行變更時，可以使用下列動作來管理您的容器叢集。

檢視 資源

以下是您可以取得容器叢集中資源相關資訊的一些方法。

- DescribeCompute-傳回有關註冊為運算之容器的詳細資訊。

- DescribeContainerGroupDefinition-傳回有關容器群組定義的詳細資訊。此資源說明群組及其容器的配置方式。
- DescribeFleetAttributes-取得叢集屬性，其中包括連線連接埠範圍和其他屬性。
- DescribeFleetCapacity-取得叢集中複本容器群組的計數及其狀態。
- DescribeRuntimeConfiguration-說明在每個複本容器群組中執行的伺服器處理序。
- GetComputeAccess-提供對託管容器群組的執行個體的遠端存取。
- GetComputeAuthToken- GameLift 為容器叢集中的運算資源請求 Amazon 的身份驗證令牌。
- ListCompute-列出註冊為運算的容器群組。
- ListContainerGroupDefinitions-列出容器群組定義。
- ListFleets-列出使用特定容器群組的叢集。

更新資源

以下是您可以建立和修改容器叢集資源的一些方法。

- CreateContainerGroupDefinition-建立容器群組定義。
- CreateFleet-設定為時`ComputeType`建立容器叢集`CONTAINER`。
- RegisterCompute-使用容器叢集進行登錄計算。
- UpdateFleetAttributes-更新叢集的可變屬性，例如 Anywhere 叢集配置選項。
- UpdateFleetCapacity-更新受管 EC2 叢集或容器叢集的容量設定。
- UpdateRuntimeConfiguration-更新執行階段設定，其中說明要在註冊為計算的每個複本容器群組中執行的伺服器處理序。

刪除資源

以下是您可以移除容器叢集資源的一些方法。

- DeleteContainerGroupDefinition-刪除容器群組定義。
- DeleteFleet-刪除艦隊。
- DeregisterCompute-從容器叢集移除計算資源。

擴展 Amazon GameLift 容器艦隊

本文件適用於公開預覽版本中的功能。內容可能變動。

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，而不會在不需要的資源上浪費金錢。在容器叢集中，您可以透過新增或移除叢集執行個體來擴展叢集容量。

建立新叢集時，Amazon 會將叢 GameLift 集所需的容量設定為一個執行個體，並在叢集的本地區域部署一個執行個體。對於多位置叢集，Amazon GameLift 會將一個執行個體部署到本地區域和每個遠端位置。在叢集狀態達到之後`ACTIVE`，您可以提高所需的容量以擴充，或降低所需的容量以縮減規模。

您可以使用 Amazon GameLift 擴展功能手動變更容量，或根據玩家需求設定自動擴展：

- 使用目標追蹤設定自動縮放。請參閱基於目標的自動縮放。
- 手動變更叢集的容量。請參閱手動設定亞馬遜GameLift叢集的容量。

擴展容器叢集時，請考慮新增或移除執行個體如何影響叢集託管遊戲工作階段和玩家的容量。

- 每個實例的遊戲會話
  - 執行個體上執行的每個遊戲伺服器程序代表主控一個遊戲工作階段的容量。
  - 使用此公式可計算在容器叢集執行個體上同時執行的遊戲工作階段數目：

    ```
    [Game sessions per instance] = [# of processes per replica container group] * [# of
     replica container groups per instance]
    ```

    - 對於每個複本容器群組的處理序，請呼叫 DescribeRuntimeConfiguration並計算遊戲伺服器處理序的同時執行次數。
    - 對於每個執行個體的複本容器群組，DescribeFleetAttributes請呼叫以取得`DesiredReplicaContainerGroupPerInstance`值。如果未設定此值，請使用該`MaxReplicaContainerGroupsPerInstance`值。
- 每個多開的玩家
  - 您可以決定每個遊戲工作階段中允許的玩家插槽數量。根據您的主機解決方案處理遊戲工作階段位置的方式，您可以在配對設定中或在通話中定義每個遊戲工作階段的玩家，以開始遊戲工作階段的位置。
  - 使用此公式計算可以在容器叢集執行個體上同時進行遊戲的玩家人數：

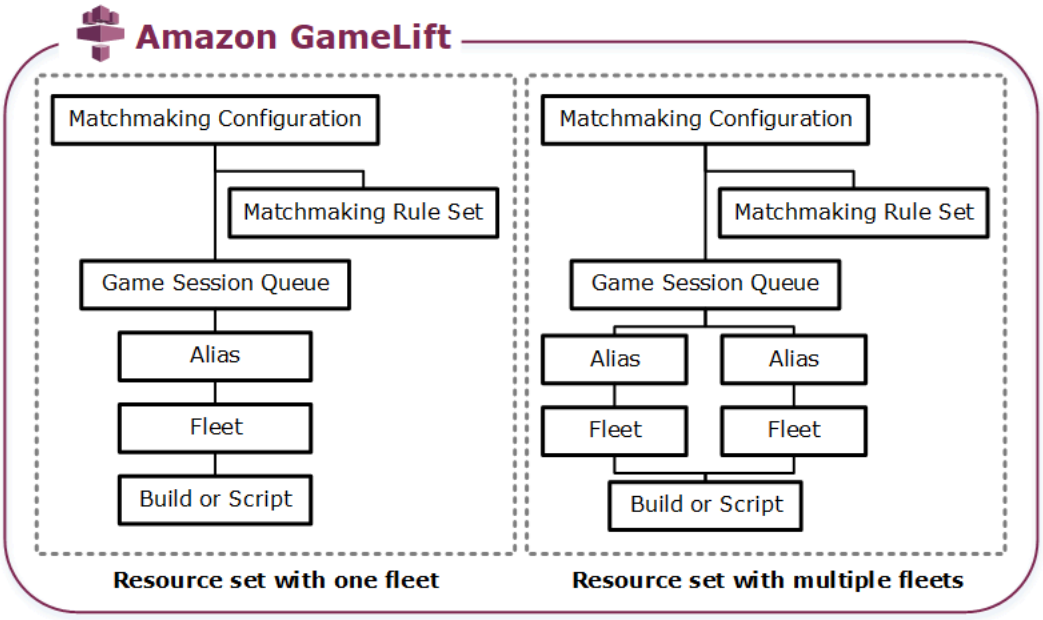
```
[Players per instance] = [# of game sessions per instance] * [# of player slots per
 game session]
```

若要取得容器叢集目前的總容量，請呼叫DescribeFleetCapacity或DescribeFleetLocation 容量以取得叢集中複本容器群組的數目。作用中的群組是指目前主控遊戲工作階段的群組。閒置群組已準備好主持新遊戲工作階段。將這些值乘以每個複本容器群組的伺服器處理作業數目。

管理亞馬遜GameLift託管資源

本節提供有關設定 Amazon GameLift 受管資源以執行遊戲伺服器和為玩家主持遊戲工作階段的詳細資訊。您必須設定和部署資源、擴充容量以符合玩家需求，以及尋找可用的資源來主持遊戲工作階段。

下圖說明 Amazon GameLift 資源物件彼此之間的關聯性。使用組建或指令碼建立叢集、為叢集指定別名，並使用其別名將叢集新增至遊戲工作階段佇列。對於使用FlexMatch配對的遊戲，請使用遊戲工作階段佇列和配對規則集來建立配對設定。



遊戲伺服器代碼

- 建置 — 在 Amazon 上執行GameLift並為您的玩家主持遊戲工作階段的自訂遊戲伺服器軟體。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，而且您必須與 Amazon 整合 GameLift。將遊戲構建文件上傳到計劃設置艦隊的亞馬遜GameLift。AWS 區域如需詳細資訊，請參閱將自訂伺服器組建上傳到亞馬遜 GameLift。
- 腳本 — 與實時服務器一起使用的配置和自定義遊戲邏輯。通過使用創建腳本來為您的遊戲客戶端配置實時服務器JavaScript，並添加自定義遊戲邏輯來為您的玩家託管遊戲會話。如需詳細資訊，請參閱將實時服務器腳本上傳到亞馬遜 GameLift。

機群

一系列運算資源，可執行您的遊戲伺服器並為您的玩家主持遊戲工作階段。如需有關可在何處部署叢集的資訊，請參閱Amazon GameLift 託管地點。如需建立叢集的相關資訊，請參閱建立亞馬遜GameLift車隊。

Alias (別名)

叢集的抽象識別碼，您可以隨時用來變更玩家所連線的艦隊。如需詳細資訊，請參閱在 Amazon GameLift 叢集中新增別名。

遊戲階段佇列

一種遊戲工作階段放置機制，可接收新遊戲工作階段的要求，並搜尋可用的遊戲伺服器以託管新工作階段。如需遊戲工作階段佇列的詳細資訊，請參閱為遊戲會話放置設置亞馬遜GameLift隊列。

將構建和腳本上傳到亞馬遜 GameLift

在部署多人遊戲伺服器以透過 Amazon 託管之前GameLift，您需要上傳遊戲伺服器檔案。本節中的主題提供準備和上傳自訂遊戲伺服器建置檔案或即時伺服器伺服器指令碼檔案的指引。

主題

- 將自訂伺服器組建上傳到亞馬遜 GameLift
- 將實時服務器腳本上傳到亞馬遜 GameLift

將自訂伺服器組建上傳到亞馬遜 GameLift

將遊戲服務器與亞馬遜集成後 GameLift，將構建文件上傳到亞馬遜 GameLift。本主題說明如何封裝遊戲的建置檔案、建立選用的組建安裝指令碼，然後使用 AWS Command Line Interface(AWS CLI) 或 AWS SDK 上傳檔案。

主題

- 封裝遊戲建置檔案
- 創建一個亞馬遜 GameLift 構建
- 更新您的構建文件
- 新增建置安裝指令碼

封裝遊戲建置檔案

將設定好的遊戲伺服器上傳到 Amazon 之前 GameLift，請先將遊戲建置檔案封裝到建置目錄中。此目錄必須包含執行遊戲伺服器和託管遊戲工作階段所需的所有元件，包括下列項目：

- 遊戲伺服器二進位檔案 — 執行遊戲伺服器所需的二進位檔案。一個組建可以包含多個遊戲伺服器的二進位檔案，以便在同一平台上執行。如需支援平台的清單，請參閱Amazon 的開發支持 GameLift。
- 相依性 — 您的遊戲伺服器可執行檔需要執行的任何相依檔案。範例包括資產、設定檔和相依程式庫。

> **Note**
>
> 對於使用 C++ 的亞馬遜 GameLift 伺服器開發套件建立的遊戲組建 (包括使用虛幻外掛程式建立的遊戲組建)，請加入與您建置伺服器 SDK 相同版本的 OpenSSL DLL。有關更多詳細信息，請參閱服務器 SDK 自述文件。

- 安裝指令碼 (選用) — 用於處理在 Amazon GameLift 主機伺服器上安裝遊戲建置的任務的指令碼檔案。將此文件放置在構建目錄的根目錄。Amazon GameLift 會在叢集建立過程中執行安裝指令碼。

您可以在組建中設定任何應用程式 (包括安裝指令碼)，以安全地存取其他AWS服務上的資源。若要取得有關如何執行此操作的資訊，請參閱與您車隊的其他AWS資源進行溝通。

在您封裝建置檔案之後，請確定您的遊戲伺服器可以在目標作業系統的全新安裝上執行。這會驗證您是否在套件中包含所有必要的相依性，以及您的安裝指令碼是否正確。

創建一個亞馬遜 GameLift 構建

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。 這是最簡單和最常用的選項。
- 在亞馬遜簡單存儲服務（亞馬遜 S3）中使用文件創建構建。使用此選項，您可以在 Amazon S3 中管理您的建置版本。

透過這兩種方法，Amazon GameLift 會建立具有唯一組建 ID 和其他中繼資料的新建置資源。組建會以 [初始化] 狀態啟動。亞馬遜 GameLift獲取遊戲服務器文件後，構建將移至「就緒」狀態。

當組建就緒時，您可以將其部署到新的 Amazon GameLift 叢集。如需詳細資訊，請參閱建立亞馬遜 GameLift 受管叢集。Amazon GameLift 設定新叢集時，會將建置檔案下載到每個叢集執行個體並安裝建置檔案。

從檔案目錄建立組建

若要建立儲存在任何位置 (包括本機目錄) 的遊戲組建，請使用upload-buildAWS CLI指令。此命令會在 Amazon 中建立新的組建記錄， GameLift 並從您指定的位置上傳檔案。

傳送上傳請求。在指令行視窗中，輸入下列upload-build指令和參數。

```
aws gamelift upload-build \
    --name user-defined name of build \
    --operating-system supported OS \
    --server-sdk-version Amazon GameLift server SDK version \
    --build-root build path \
    --build-version user-defined build number \
    --region region name
```

- operating-system-遊戲服務器構建的運行時環境。您必須指定作業系統值。您無法稍後更新此項目。
- server-sdk-version— 您的遊戲 GameLift伺服器與之整合的 Amazon 伺服器 SDK 版本。如果你不提供一個值，亞馬遜 GameLift 使用默認值4.0.2。如果您指定了不正確的伺服器 SDK 版本，則在呼叫InitSdk建立與 Amazon GameLift 服務的連線時，遊戲伺服器組建可能會失敗。
- build-root-構建文件的目錄路徑。
- name— 新組建的描述性名稱。
- build-version-構建文件的版本詳細信息。
- region— 您要建立組建的AWS區域。在您計劃部署叢集的區域中建立組建。如果您要在多個地區部署遊戲，請在每個區域建立一個組建。

> **Note**
>
> 使用檢視您目前的預設地區aws configure get region。若要變更預設「地區」，請使用aws configure set region ***region name***指令。

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 提供上傳進度。成功上傳後，Amazon 會傳 GameLift 回新的組建記錄 ID。上傳時間取決於遊戲檔案的大小和連線速度。

使用 Amazon S3 中的文件創建構建

您可以將構建文件存儲在 Amazon S3 中，然後 GameLift 從那裡將其上傳到亞馬遜。建立建置時，您必須指定 S3 儲存貯體位置，Amazon GameLift 會直接從 Amazon S3 擷取建置檔案。

若要建立組建資源

1. 將您的建置檔案存放在 Amazon S3 中。建立包含已封裝建置檔案的 .zip 檔案，並將其上傳到 AWS 帳戶. 記下存儲桶標籤和文件名，創 GameLift 建 Amazon 構建時需要這些標籤。
2. 讓亞馬遜 GameLift 訪問您的構建文件。依照中的指示建立 IAM 角色在 Amazon S3 中訪問遊戲構建文件。建立角色之後，請記下新角色的 Amazon 資源名稱 (ARN)，建立組建時需要此名稱。
3. 建立組建。使用 Amazon GameLift 主控台或AWS CLI建立新的建置記錄。您必須具有`PassRole`權限，如中所述亞馬遜的 IAM 許可示例 GameLift。

Console

1. 在 Amazon 主 GameLift控台的導覽窗格中，選擇託管、組建。
2. 在 [組建] 頁面上，選擇 [建立組建]。
3. 在 [建立組建] 頁面的 [組建設定] 下，執行下列動作：
    a. 在名稱中，輸入指令集名稱。
    b. 在版本中，輸入版本。因為您可以更新組建的內容，因此版本資料可協助您追蹤更新。
    c. 對於操作系統（OS），請選擇遊戲服務器構建的操作系統。您稍後無法更新此值。

d. 對於遊戲伺服器組建，請輸入您上傳到 Amazon S3 之建置物件的 S3 URI，然後選擇物件版本。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. 對於 IAM 角色，請選擇您建立的角色，讓 Amazon 能夠 GameLift 存取 S3 儲存貯體並建立物件。

4. (選用) 在「標籤」下，輸入「金鑰」和「值」配對，將標籤新增至組建。

5. 選擇 建立 。

Amazon GameLift 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

若要定義新組建並上傳伺服器組建檔案，請使用create-build指令。

1. 開啟命令行視窗並切換至您可以使用的目錄AWS CLI。

2. 輸入以下create-build命令：

```
aws gamelift create-build \
    --name user-defined name of build \
    --server-sdk-version Amazon GameLift server SDK 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— 新組建的描述性名稱。
- server-sdk-version— 您用於將遊戲 GameLift 服務器與亞馬遜集成的亞馬遜服務器 SDK 的版本 GameLift。如果你不提供一個值，亞馬遜 GameLift 使用默認值4.0.2。
- operating-system-遊戲服務器構建的運行時環境。您必須指定作業系統值。您無法稍後更新此項目。
- build-version-構建文件的版本詳細信息。這項資訊非常有用，因為每個新版本的遊戲伺服器都需要新的組建資源。
- storage-location
  - Bucket— 包含組建的 S3 儲存貯體的名稱。例如，「我的構建文件」。
  - Key— 包含組建檔案的 .zip 檔案名稱。例如，「我的遊戲建置 7.0.1，7.0.2」。

- RoleARN— 指派給您建立的 IAM 角色的 ARN。例如，「ARN：你：IAM:: 11112222333：角色/」。GameLiftAccess如需政策範例，請參閱 在 Amazon S3 中訪問遊戲構建文件。
- region— 在您計劃部署艦隊的AWS區域中創建構建。如果您要在多個地區部署遊戲，請在每個區域建立一個組建。

> **Note**
>
> 我們建議您使用configure get指令來檢查目前的預設區域。若要變更您的預設「地區」，請使用configure set指令。

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::111
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指令。

更新您的構建文件

您可以使用 Amazon GameLift 主控台或update-buildAWS CLI命令更新建置資源的中繼資料。

建立 Amazon GameLift 組建之後，就無法更新與其相關聯的建置檔案。針對每組新的檔案，建立新的 Amazon 組 GameLift 建。使用該upload-build命令，亞馬遜 GameLift 自動為每個請求創建一個新的構建記錄。如果您使用create-build命令提供建置檔案，請將具有不同名稱的新組建 .zip 檔案上傳到 Amazon S3，並透過參考新檔案名稱來建立組建。

部署更新建置時可嘗試以下訣竅：

- 視需要使用佇列和換出機群。使用 Amazon 設定遊戲用戶端時 GameLift，請指定佇列而不是叢集。使用佇列，您可以將具有新組建的新叢集新增至佇列，並移除舊的叢集。如需詳細資訊，請參閱為遊戲會話放置設置亞馬遜GameLift隊列。

- 使用別名將玩家轉移到新的遊戲版本。將您的遊戲用戶端與 Amazon 整合時 GameLift，請指定叢集別名，而不是叢集 ID。如需詳細資訊，請參閱在 Amazon GameLift 叢集中新增別名。
- 設定自動組建更新。有關將 Amazon GameLift 部署合併到建置系統的範例指令碼和資訊，請參閱 AWS遊戲技術部落格 GameLift上的自動化部署到 Amazon。

新增建置安裝指令碼

為遊戲組建的作業系統 (OS) 建立安裝指令碼：

- 視窗：建立名為`install.bat`.
- Linux：建立一個名為`install.sh`.

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 會在以下位置的託管伺服器上 GameLift 安裝組建並重新建立組建套件中的檔案目錄：
  - 視窗艦隊：`C:\game`
  - 艦隊：`/local/game`
- 在 Linux 叢集的安裝程序期間，執行身分使用者對執行個體檔案結構的存取權限有限。此使用者擁有安裝組建檔案的目錄的完整權限。如果您的安裝指令碼執行需要管理員權限的動作，請使用指定管理員存取權sudo。Windows 叢集的執行身分使用者預設具有系統管理員權限。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，亞馬遜 GameLift 支持常見的外殼解釋器語言，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 執行個體並開啟殼層提示字元。如需詳細資訊，請參閱遠端連線至 Amazon GameLift 叢集執行個體。
- 安裝指令碼無法依賴 VPC 對等連線。在 Amazon 在叢集執行個體上 GameLift 安裝組建之後，才能使用 VPC 對等連線。

Example 視窗安裝 bash 文件

此範例`install.bat`檔案會安裝遊戲伺服器所需的 Visual C++ 執行階段元件，並將結果寫入記錄檔中。指令碼會在根目錄的組建套件中包含元件檔案。

```
vcredist_x64.exe /install /quiet /norestart /log c:\game\vcredist_2013_x64.log
```

Example 安裝外殼腳本

這個範例`install.sh`檔案會在安裝指令碼中使用 bash，並將結果寫入記錄檔。

```
#!/bin/bash
echo 'Hello World' > install.log
```

此範例`install.sh`檔案顯示如何使用 Amazon CloudWatch 代理程式收集系統層級和自訂指標，以及處理日誌輪替。由於 Amazon 在服務 VPC 中 GameLift 執行，因此您必須授予 Amazon GameLift 許可，才能代表您擔任 AWS Identity and Access Management (IAM) 角色。若要允許 Amazon GameLift 擔任角色，請建立包含AWS受管政策的角色`CloudWatchAgentAdminPolicy`，並在建立叢集時使用該角色。

```
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
```

將實時服務器腳本上傳到亞馬遜 GameLift

當您準備好為遊戲部署即時伺服器時，請將已完成的即時伺服器指令碼檔案上傳到 Amazon GameLift。透過建立 Amazon 指GameLift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命令檔資源的新檔案，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，Amazon GameLift 會為其指派唯一的指令碼 ID (例如`script-1111aaaa-22bb-33cc-44dd-5555eeee66ff`)，並上傳指令碼檔案的副本。上傳時間取決於腳本文件的大小和連接速度。

建立指令碼資源後，Amazon GameLift 會使用新的即時伺服器叢集部署指令碼。Amazon GameLift 會將您的伺服器指令碼安裝到叢集中的每個執行個體上，並將指令碼檔案放入中`/local/game`。

若要疑難排解與伺服器指令碼相關的叢集啟動問題，請參閱偵錯亞馬遜GameLift叢集問題。

封裝指令碼檔

您的伺服器指令碼可以包含一個或多個檔案合併為單一 .zip 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 簡單儲存服務 (Amazon S3) 儲存貯體中。

從本地目錄上傳腳本文件

如果您的腳本文件存儲在本地，則可以GameLift從那裡將它們上傳到亞馬遜。若要建立指令碼資源，請使用 Amazon GameLift 主控台或 AWS Command Line Interface(AWS CLI)。

Amazon GameLift console

若要建立命令檔資源

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格中，選擇 [主機]、[指令碼]。
3. 在「命令檔」頁面上，選擇建立命令檔。
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，輸入指令集名稱。
   b. (選擇性) 針對版本，輸入版本資訊。因為您可以更新指令碼的內容，因此版本資料對追蹤更新很有幫助。
   c. 針對「指令碼來源」，選擇「上傳 .zip 檔案」。
   d. 針對指令碼檔案，請選擇 [選擇檔案]，瀏覽找出包含指令碼的 .zip 檔案，然後選擇該檔案。
5. (選擇性) 在標籤下，輸入「鍵」和「值」配對，將標籤新增至指令集。
6. 選擇 建立 。

   Amazon GameLift 會將 ID 指派給新指令碼，並上傳指定的 .zip 檔案。您可以在「指令集」頁面上檢視新命令檔，包括其狀態。

AWS CLI

使用指`create-script`AWS CLI令定義新的指令碼，並上傳伺服器指令碼檔案。

若要建立命令檔資源

1. 將 .zip 檔案放置在您可以使用的AWS CLI目錄中。
2. 開啟命令列視窗並切換至放置 .zip 檔案的目錄。
3. 輸入以下create-script命令和參數。對於--zip-file參數，請務必將字串新增`fileb://`至 .zip 檔案的名稱。它將文件識別為二進製文件，以便亞馬遜GameLift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，Amazon 會GameLift傳回新的指令碼物件。

4. 若要檢視新指令碼，請呼叫describe-script。

從亞馬遜 S3 上傳腳本文件

您可以將腳本文件存儲在 Amazon S3 存儲桶中，然後GameLift從那裡將它們上傳到亞馬遜。建立指令碼時，請指定 S3 儲存貯體位置，Amazon GameLift 會從 Amazon S3 擷取您的指令碼檔案。

若要建立命令檔資源

1. 將您的指令碼檔案存放在 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的 S3 儲存貯體。AWS 帳戶記下物件 URI — 您在建立 Amazon GameLift 指令碼時需要這個動作。

   > **Note**
   >
   > Amazon GameLift 不支援從名稱包含句點 (.) 的 S3 儲存貯體上傳。

2. 讓亞馬遜GameLift訪問您的腳本文件。若要建立允許 Amazon 存取包含伺服器指令碼GameLift的 S3 儲存貯體的 AWS Identity and Access Management (IAM) 角色，請按照中的說明進行操作為 Amazon 設置 IAM 服務角色 GameLift。建立新角色之後，請記下它的名稱，這是您在建立指令碼時所需要的。
3. 建立指令碼。使用 Amazon GameLift 主控台或建AWS CLI立新的指令碼記錄。若要提出此要求，您必須具有 IAM `PassRole` 權限，如中所述亞馬遜的 IAM 許可示例 GameLift。

Amazon GameLift console

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇「主機」、「指令碼」。
2. 在「命令檔」頁面上，選擇建立命令檔。

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，輸入指令集名稱。

   b. (選擇性) 針對版本，輸入版本資訊。因為您可以更新指令碼的內容，因此版本資料對追蹤更新很有幫助。

   c. 對於指令碼來源，請選擇亞馬遜 S3 URI。

   d. 輸入您上傳到 Amazon S3 的指令碼物件的 S3 URI，然後選擇物件版本。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. (選擇性) 在標籤下，輸入「鍵」和「值」配對，將標籤新增至指令集。

5. 選擇 建立 。

   Amazon GameLift 會將 ID 指派給新指令碼，並上傳指定的 .zip 檔案。您可以在「指令集」頁面上檢視新命令檔，包括其狀態。

AWS CLI

使用指create-scriptAWS CLI令定義新的指令碼，並上傳伺服器指令碼檔案。

1. 開啟命令列視窗並切換至您可以使用的目錄AWS CLI。

2. 輸入以下create-script命令和參數。此--storage-location參數會指定指令碼檔案的 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，Amazon 會GameLift傳回新的指令碼物件。

3. 若要檢視新指令碼，請呼叫[describe-script]。

更新腳本文件

您可以使用 Amazon GameLift 主控台或命令更新[update-script]AWS CLI指令碼資源的中繼資料。

您也可以更新指令集資源的指令碼內容。Amazon GameLift 會將指令碼內容部署到使用更新指令碼資源的所有叢集執行個體。部署更新的指令碼後，執行個體會在啟動新遊戲工作階段時使用該指令碼。更新時已經執行的遊戲工作階段不會使用更新的指令碼。

更新腳本檔的步驟

- 對於存放在本機的指令碼檔案，若要上傳更新的指令碼 .zip 檔案，請使用 Amazon GameLift 主控台或命update-script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。Amazon 會GameLift定期檢查更新的指令碼檔案，並直接從 S3 儲存貯體擷取這些檔案。

建立亞馬遜GameLift車隊

本節提供有關設計、建置和維護叢集以搭配 Amazon GameLift 使用的詳細資訊。您可以使用 Amazon機GameLift隊部署自訂遊戲伺服器和即時伺服器。

叢集將您的託管資源代表為一組 Amazon 彈性運算雲端 (Amazon EC2) 執行個體或實體硬體。叢集的位置會決定執行個體或硬體的部署位置，以便為您的玩家主持遊戲工作階段。艦隊的規模，以及它可以支援的遊戲工作階段和玩家數量，取決於執行個體的數量或您提供的硬體數量。您可以手動調整虛擬執行個體，也可以使用自動調整比例

生產中的許多遊戲都使用一個以上的艦隊。例如，您可以使用多個叢集同時執行多個版本的遊戲伺服器、為競價型叢集提供備份容量，或建置備援。

要了解如何創建針對您遊戲需求而設計的艦隊，請從開始亞馬遜GameLift車隊設計指南。在您的叢集執行之後擴展亞馬遜GameLift託管容量，請參閱在 Amazon GameLift 叢集中新增別名、和為遊戲會話放置設置亞馬遜GameLift隊列。

主題

- 亞馬遜GameLift車隊設計指南

- 建立新的亞馬遜GameLift車隊
- 管理您的亞馬遜GameLift車隊
- 在 Amazon GameLift 叢集中新增別名
- 偵錯亞馬遜GameLift叢集問題
- 遠端連線至 Amazon GameLift 叢集執行個體

亞馬遜GameLift車隊設計指南

本設計指南涵蓋建立與 Amazon 搭配使用的託管資源群組的最佳實務GameLift。選擇託管資源的組合，並了解如何配置它們以適合您的遊戲。

主題

- 選擇 Amazon GameLift 運算資源
- 管理 Amazon 如何GameLift啟動遊戲伺服器
- 搭配 Amazon 使用競價型執行個 GameLift

選擇 Amazon GameLift 運算資源

為了為玩家部署遊戲伺服器和託管遊戲工作階段，Amazon GameLift 使用稱為執行個體的 Amazon 彈性運算雲端 (Amazon EC2) 資源或您的實體硬體。使用執行個體設定新叢集時，請決定您需要的執行個體類型，以及如何在這些執行個體上執行遊戲伺服器程序。當受管 EC2 叢集處於作用中狀態並準備好託管遊戲工作階段時，您可以視需要新增或移除執行個體以滿足玩家需求。

您可以在兩種運算類型的組合上部署 Amazon GameLift 遊戲伺服器：

- 受管 EC2 — 受管 EC2 叢集使用 Amazon EC2 執行個體託管您的遊戲伺服器。Amazon 會 GameLift 管理執行個體，並免除託管遊戲時硬體和軟體管理的負擔。
- Amazon GameLift Anywhere — Amazon GameLift Anywhere 車隊使用您現有的基礎設施託管遊戲伺服器，同時 Amazon GameLift 管理您的配對和隊列。

選擇叢集的運算資源時，請考量下列因素：

- 可用硬體
- 車隊位置
- 隨需執行個體與 Spot 執行

- 作業系統
- 執行個體類型
- Service Quotas

可用硬體

考慮實作中的現有基礎結構。將遊戲遷移到 Amazon 時 GameLift，您可以繼續使用基礎設施。透過 Amazon GameLift Anywhere，您可以使用自己的基礎設施以及 Amazon GameLift 受管 EC2 執行個體。您還可以使用現有的基礎設施來託管比支持的 Amazon GameLift 位置更接近玩家的遊戲。如需設定 Amazon GameLift Anywhere 叢集的詳細資訊，請參閱創建一個 Amazon GameLift Anywhere 車隊。

車隊位置

請考慮您計劃部署遊戲伺服器的地理位置。執行個體類型的可用性因本機區域而異。 AWS 區域

對於多地點叢集，執行個體的可用性和配額取決於叢集的本地區域和選定遠端位置的組合。如需叢集位置的詳細資訊，請參閱Amazon GameLift 託管地點。

對於 Amazon GameLift Anywhere 叢集，您可以決定實體硬體的位置。如需自訂位置的詳細資訊，請參閱Amazon GameLift Anywhere。

隨需執行個體與 Spot 執行

Amazon EC2 隨需執行個體和競價型執行個體提供相同的硬體和效能，但可用性和成本不同。

隨需執行個體

您可以在需要時取得隨需執行個體，並視需要保留該執行個體。隨需執行個體有固定的成本，這表示您需要支付使用時間的費用，而且無需簽訂長期合約。

Spot 執行個體

Spot 執行個體可利用未使用的 AWS 運算容量，提供符合成本效益的隨需執行個體替代 競價型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變動。 AWS 可以在需要重新容量時中斷 Spot 執行個體。Amazon GameLift 使用佇列和 FleetIQ 演算法判斷 AWS 是否要中斷競價型執行個體，並將執行個體置於回收狀態。然後，當執行個體上沒有使用中的遊戲工作階段時，Amazon 會 GameLift 嘗試取代它。

如需如何使用 Spot 執行個體的詳細資訊，請參閱搭配 Amazon 使用競價型執行個 GameLift。

作業系統

Amazon GameLift 執行個體支援在 Microsoft 視窗或 Amazon Linux 上執行的遊戲伺服器組建。當您將遊戲組建上傳到 Amazon 時 GameLift，請指定遊戲的作業系統。當您建立 Amazon EC2 叢集來部署遊戲組建時，Amazon GameLift 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Amazon 的開發支持 GameLift。

使用 Amazon GameLift Anywhere 叢集時，您可以使用硬體支援的任何作業系統。Amazon GameLift Anywhere 叢集要求您將遊戲組建部署到硬體，同時使用 Amazon GameLift 在單一位置管理資源。

執行個體類型

Amazon EC2 叢集的執行個體類型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、儲存和網路功能組合。

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 重力子)。

  > **Note**
  >
  > 重力臂執行個體需要在 Linux 作業系統上建置 Amazon GameLift 伺服器。C ++ 和 C # 需要服務器 SDK 5.1.1 或更新版本。轉到需要服務器 SDK 5.0 或更新版本。這些執行個體不 out-of-the-box 支援 Amazon Linux 2023 (AL2023) 或 Amazon Linux 2 (AL2) 上的單聲道安裝。

- 遊戲伺服器組建的運算、記憶體和儲存需求。
- 您計劃每個執行個體執行的伺服器處理序數目。

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處理序。這樣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 Amazon 如何GameLift啟動遊戲伺服器。

Service Quotas

若要查看 Amazon GameLift 的預設服務配額以及您目前的配額 AWS 帳戶，請執行下列動作：

- 如需 Amazon 的一般服務配額資訊 GameLift，請參閱 AWS 一般參考. GameLift
- 如需帳戶每個位置的可用執行個體類型清單，請開啟 Amazon GameLift 主控台的服務配額頁面。此頁面也會顯示每個位置中每個執行個體類型的帳戶目前使用量。
- 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 (或您指定的其他區域) 中擁有的作用中執行個體數目。

準備啟動遊戲時，請在 Amazon GameLift 主控台填寫啟動問卷。Amazon GameLift 團隊會使用啟動問卷來判斷您遊戲的正確配額和限制。

管理 Amazon 如何GameLift啟動遊戲伺服器

您可以設定受管 EC2 叢集的執行階段組態，以便在每個執行個體執行多個遊戲伺服器處理序。這樣可以更有效地使用您的託管資源。

機群如何管理多個程序

Amazon GameLift 使用叢集的執行階段組態來判斷每個執行個體上要執行的程序類型和數目。執行階段組態至少包含一個代表一個遊戲伺服器可執行檔的伺服器處理序組態。您可以定義其他伺服器處理序組態，以執行與您的遊戲相關的其他類型的處理序。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。
- 要同時執行的處理序數目。

當叢集中的執行個體啟動時，會啟動執行階段設定中定義的一組伺服器處理序。通過多個流程，Amazon GameLift 錯開了每個流程的啟動。伺服器處理序的壽命有限。當它們結束時，Amazon會GameLift啟動新程序，以維護執行時期組態中定義的伺服器處理序數量和類型。

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叢集執行階段設定的更新，以實作變更。以下是亞馬遜如何GameLift採用運行時配置更改：

1. 執行個體會傳送要求給 Amazon，以GameLift取得最新版本的執行階段組態。
2. 執行處理會比較其使用中處理作業與最新的執行階段組態，然後執行下列動作：
   - 如果更新的執行階段組態移除伺服器處理作業類型，則此類型的作用中伺服器處理作業會繼續執行，直到結束為止。執行個體不會取代這些伺服器處理序。

- 如果更新的程式實際執行組態減少伺服器處理作業類型的並行處理作業數目，則此類型的多餘伺服器處理作業會繼續執行，直到結束為止。執行個體不會取代這些多餘的伺服器處理序。
- 如果更新的執行階段設定新增伺服器處理序類型或增加現有類型的並行處理程序，則執行個體會啟動新的伺服器處理序，最多可達 Amazon 的GameLift最大值。在此情況下，執行個體會在現有處理序結束時啟動新的伺服器處理序。

針對多個程序最佳化叢集

若要在叢集上使用多個程序，請執行下列動作：

- 建立包含您要部署到叢集的遊戲伺服器可執行檔的組建，然後將組建上傳到 Amazon GameLift。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 Amazon GameLift Server SDK。
- 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 版本整合。

若要最佳化叢集效能，建議您執行下列動作：

- 處理伺服器處理序關閉案例，讓 Amazon GameLift 能有效率地回收處理程序。例如：
  - 在調用伺服器 API 的遊戲伺服器程式碼中新增關閉程序`ProcessEnding()`。
  - 在遊戲伺服器程式碼`OnProcessTerminate()`中實作回呼函式，以處理來自 Amazon 的終止請求GameLift。
- 確保 Amazon GameLift 關閉並重新啟動狀態不良的伺服器處理序。GameLift透過在遊戲伺服器程式碼中實作`OnHealthCheck()`回呼函數，將健康狀態回報給 Amazon。Amazon GameLift 會自動關閉連續三個報告中報告狀況不良的伺服器處理序。如果您沒有實作`OnHealthCheck()`，Amazon 會GameLift假設伺服器處理序正常，除非程序無法回應通訊。

選擇每個執行個體的處理序數

決定要在執行環境上執行的並行處理作業數目時，請記住下列事項：

- Amazon 將每個執行個體GameLift限制為同時處理序的最大數量。叢集伺服器處理序組態的所有並行處理序總和不得超過此配額。
- 為了維持可接受的效能等級，Amazon EC2 執行個體類型可能會限制可同時執行的程序數量。針對您的遊戲測試不同的設定，找出適合您偏好執行個體類型的正確處理程序數目。
- Amazon 執行的並行處理程序不GameLift會超過設定的總數。這意味著從先前的運行時配置轉換到新配置可能會逐漸發生。

搭配 Amazon 使用競價型執行個 GameLift

設定 GameLift 受管 EC2 叢集時，您可以使用競價型執行個體、隨需執行個體或組合。進一步了解 Amazon 如何在中 GameLift 使用競價型執行個體隨需執行個體與 Spot 執行。若要使用競價型艦隊，您的遊戲整合需要此頁面上列出的調整。

您是否正在使 FlexMatch 用配對？ 您可以將 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. 設計 Spot 執行個體的遊戲工作階段佇列。

   使用佇列管理遊戲工作階段配置是最佳做法，使用 Spot 執行個體時也是必要的。若要設計佇列，請考慮下列事項：

   - 位置 — 為了獲得最佳的玩家體驗，請選擇地理位置靠近您的玩家。
   - 執行個體類型 — 考量您的遊戲伺服器硬體需求，以及所選位置的執行個體可用性。

   若要嘗試最佳化 Spot 可用性和恢復能力的佇列，請參閱。教學課程：設定 Spot 執行個體的遊戲工作階段佇列
2. 針對 Spot 最佳化佇列建立機群。

   根據您的佇列設計，建立叢集，將遊戲伺服器部署到您想要的位置和執行個體類型。請參閱 建立亞馬遜 GameLift 受管叢集 協助建立及設定新機群。
3. 建立您的遊戲工作階段佇列。

   新增艦隊目的地、設定遊戲工作階段放置程序，以及定義放置優先順序。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，佇列可避免資源中斷的機率很高，並選擇符合您定義的優先順序的位置。如需協助在遊戲用戶端實作遊戲工作階段配置，請參閱 建立遊戲階段。
5. 更新您的遊戲伺服器以處理 Spot 中斷。

   AWS 當需要恢復容量時，可以通過 2 分鐘的通知中斷 Spot 執行個體。設定您的遊戲伺服器以處理中斷，以盡量減少玩家的影響。

   在 AWS 回收 Spot 執行個體之前，它會傳送終止通知。Amazon GameLift 通過調用 Amazon 服務器 SDK 回調函數`onProcessTerminate()`將通知傳遞給所有受影響的 GameLift服務器進程。實

施此回調以結束遊戲會話或將遊戲會話和玩家移動到新的實例。請參閱 回應伺服器處理序關閉通知 協助實作 onProcessTerminate()。

> **Note**
>
> AWS 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 AWS 前回收 Spot 執行個體。準備好遊戲伺服器以處理意外中斷。

6. 檢閱 Spot 叢集和佇列的效能。

   在 Amazon GameLift 控制台或使用 Amazon 查看 Amazon GameLift 指標 CloudWatch 以查看性能。如需 Amazon GameLift 指標的詳細資訊，請參閱監控亞馬遜GameLift與亞馬遜CloudWatch。關鍵指標包括：

   - 中斷率 — 使用InstanceInterruptions和指GameSessionInterruptions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。由回收的遊戲工作階段 AWS 具有狀態TERMINATED和狀態原因。INTERRUPTED
   - 佇列有效性 — 追蹤放置成功率、平均等待時間和佇列深度，以確認 Spot 叢集不會影響佇列效能。
   - 車隊使用量 — 監控執行個體、遊戲工作階段和玩家工作階段的資料 隨需叢集的使用量可能是佇列避免放置到 Spot 叢集以避免中斷的指標。

建立新的亞馬遜GameLift車隊

創建一個新的車隊並部署您的自定義遊戲服務器構建或實時服務器進行託管。您可以部署任何上傳到 Amazon 的遊戲組建或指令碼資源GameLift。

主題

- 亞馬遜GameLift車隊創建如何運作
- 建立亞馬遜 GameLift 受管叢集
- 創建一個 Amazon GameLift Anywhere 車隊

亞馬遜GameLift車隊創建如何運作

當您建立新叢集時，Amazon 會GameLift啟動一個工作流程，在每個叢集位置使用一個 Amazon 彈性運算雲端 (Amazon EC2) 執行個體建立叢集。當 Amazon GameLift 完成工作流程的每個步驟時，叢集會發出事件，Amazon GameLift 更新車隊的狀態。您可以使用 Amazon GameLift 主控台或呼叫

Amazon GameLift API 操作來追蹤所有事件DescribeFleetEvents。您也可以使用追蹤個別位置的狀態DescribeFleetLocationAttributes。

EC2 叢集建立工作流程：

- Amazon GameLift 會在叢集的本地區域和叢集中定義的每個遠端位置建立叢集資源。
- Amazon 將所需的容量GameLift設定為一個執行個體。
- 亞馬遜GameLift將車隊和位置狀態設置為新。
- Amazon GameLift 開始將事件寫入叢集事件日誌。
- Amazon GameLift 會在每個叢集位置為一個新執行個體配置請求的運算資源。
- Amazon 會將遊戲伺服器檔案下GameLift載到每個執行個體，並將叢集狀態設定為 [下載中]。
- Amazon 會GameLift驗證每個執行個體上下載的遊戲伺服器檔案，以確認下載期間沒有發生錯誤。亞馬遜GameLift將車隊狀態設置為驗證。
- Amazon 在每個執行個體上GameLift建置遊戲伺服器，並將叢集狀態設定為 [建置]。
- Amazon 依照叢集執行階段組態中的指示，GameLift開始在每個執行個體上啟動伺服器處理序。如果您將叢集設定為每個執行個體執行多個並行伺服器處理序，則 Amazon 會將程序啟動GameLift交錯幾秒鐘。當每個流程都在線時，它會向亞馬遜報告準備情況GameLift。亞馬遜GameLift將車隊狀態設置為激活。
- Amazon 會將叢GameLift集狀態和位置狀態設定為「作用中」，因為伺服器處理程序報告就緒。

Amazon GameLift Anywhere fleet creation

- 亞馬遜GameLift創建了一個車隊資源。對於車隊的本地區域和叢集中定義的每個自訂位置，Amazon 會將叢GameLift集和位置狀態設定為 [新增]。
- Amazon GameLift 開始將事件寫入叢集事件日誌。
- 叢集中的一個伺服器程序通知 Amazon 已GameLift就緒後，Amazon 就會將叢GameLift集狀態和位置狀態設為 Active。當其他叢集位置的伺服器處理序報告準備就緒時，Amazon 會將每個叢GameLift集位置的狀態設為 Active。

如需疑難排解叢集建立問題的說明，請參閱偵錯亞馬遜GameLift叢集問題。

建立亞馬遜 GameLift 受管叢集

使用 Amazon 主 GameLift 控台或 AWS Command Line Interface (AWS CLI) 建立受管叢集。

建立新的受管 EC2 叢集之後，Amazon GameLift 部署叢集並安裝和啟動遊戲伺服器時，叢集的狀態會經過數個階段。艦隊已準備好在達到ACTIVE狀態後主持遊戲工作階段。如需建立叢集問題的協助，請參閱偵錯亞馬遜GameLift叢集問題。

Console

若要建立受管 EC2 叢集

1. 在 Amazon 主 GameLift 控台的導覽窗格中，選擇「叢集」。
2. 在 Fleets (機群) 頁面上，選擇 Create fleet (建立機群)。
3. 選擇受管 EC2。
4. 在 [叢集詳細資料] 頁面上執行下列動作：
   a. 在名稱中，輸入叢集名稱。我們建議在您的叢集名稱中加入叢集類型 (Spot 或隨選)。這使得在檢視叢集清單時更容易識別叢集類型。
   b. 在「描述」中，提供叢集的簡短描述。
   c. 對於二進位類型，請選取建立或指令碼以定義 Amazon GameLift 部署到此叢集的遊戲伺服器類型。
   d. 從上傳的指令碼或組建的下拉式清單中選取「指令碼」或「建置」。
5. (選擇性) 在 [其他詳細資訊] 下，以取得下列
   a. 針對執行個體角色，指定 IAM 角色，以授權遊戲組建中的應用程式存取帳戶中的其他AWS資源。如需詳細資訊，請參閱與您車隊的其他AWS資源進行溝通。若要建立具有執行個體角色的叢集，您的帳戶必須具有 IAM `PassRole` 權限。如需詳細資訊，請參閱亞馬遜的 IAM 許可示例 GameLift。

      如果您要授權非伺服器可執行檔的應用程式 (例如 CloudWatch 代理程式)，請啟用共用認證選項。

      您無法在建立叢集之後更新這些設定。
   b. 對於產生認證，請選擇讓 Amazon 為叢集 GameLift 產生 TLS 憑證。您可以使用機群 TLS 憑證，讓遊戲用戶端在連線時驗證遊戲伺服器，以及加密所有用戶端/伺服器通訊。對於啟用 TLS 的叢集中的每個執行個體，Amazon GameLift 也會使用憑證建立新的 DNS 項目。使用這些資源來設定遊戲的身分驗證和加密。
   c. 針對測量結果群組，輸入新的或現有叢集測量結果群組的名稱。您可以將多個叢集的指標新增至相同的量度群組，來彙總這些指標。

建立叢集之後，您無法更新量度群組。

6. 選擇下一步。
7. 在 [選取位置] 頁面上，選取一或多個要部署執行個體的其他遠端位置。系統會根據您存取主控台的地區自動選取主要區域。如果您選取其他位置，叢集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AWS 帳戶.
>
> - 您在 2022 年 2 月 28 日之前建立的未啟用區域的艦隊將不會受到影響。
> - 若要建立新的多位置叢集或更新現有的多位置叢集，請先啟用您選擇使用的任何區域。
>
> 如需有關預設未啟用的區域以及如何啟用的詳細資AWS 區域訊，請參閱 AWS 一般參考.

8. 選擇下一步。
9. 在 [定義執行個體詳細資訊] 頁面上，

   a. 此叢集的隨需或 Spot 執行個體。如需叢集類型的詳細資訊，請參閱隨需執行個體與 Spot 執行。

   b. 從「篩選器架構」功能表中選擇 x64 或「Arm」。

   > **Note**
   >
   > 重力臂執行個體需要在 Linux 作業系統上建置亞馬遜 GameLift 伺服器。C ++ 和 C # 需要服務器 SDK 5.1.1 或更新版本。轉到需要服務器 SDK 5.0 或更新版本。這些執行個體不 out-of-the-box 支援亞馬遜 Linux 2023 (AL2023) 或亞馬遜 Linux 2 (AL2) 上的單聲道安裝。

   如需 Amazon EC2 Arm 架構的相關資訊，請參閱AWS重力發處理器和 Amazon EC2 執行個體類型。

有關 Amazon 支援的執行個體類型的資訊 GameLift，請參閱 CreateFleet() 請求參數下的`EC2InstanceType`值。

10. 從清單中選取一個 Amazon EC2 執行個體類型。如需選擇執行個體類型的詳細資訊，請參閱執行個體類型。建立叢集之後，就無法變更執行個體類型。
11. 選擇下一步。
12. 在 [設定執行階段] 頁面的 [執行階段設定] 底下，執行下列

    a. 針對 Launch 路徑，請在您的組建或指令碼中輸入遊戲執行檔的路徑。在 Windows 執行個體、遊戲伺服器路徑 `C:\game`。在 Linux 執行個體上，遊戲伺服器是針對`/local/game`. 範例：**`C:\game\MyGame\server.exe`**`/local/game/MyGame/server.exe`、或**`MyRealtimeLaunchScript.js`**。

    b. (選擇性) 對於 Launch 參數，請輸入要作為一組命令列參數傳遞至遊戲執行檔的資訊。範例：**`+sv_port 33435 +start_lobby`**。

    c. 針對並行處理，請選擇叢集中每個執行處理要同時執行的伺服器處理作業數目。查看 Amazon 同時伺服器處理序數量的 GameLift 限制。

    對每個執行個體之並行伺服器程序的限制適用於所有組態的並行程序總數。如果您將叢集設定為超出限制，則無法啟動叢集。
13. 在「遊戲工作階段啟動」下方，提供限制以在此艦隊中的執行個體上啟動新遊戲工作階段

    a. 對於最大同時啟動遊戲工作階段，請輸入同時啟動的執行個體上的遊戲工作階段數目。啟動多個新遊戲工作階段時，此限制很有用，可能影響在執行個體上執行之其他遊戲工作階段的效能。

    b. 在 [新啟動逾時] 中，輸入等待工作階段啟動的時間長度。如果遊戲工作階段未在逾時前移至`ACTIVE`狀態，Amazon 會 GameLift 終止遊戲工作階段啟用。
14. (選用) 在 EC2 連接埠設定下，執行下列動作：

    a. 選擇 [新增連接埠設定]，為連線至叢集上部署之伺服器處理序的輸入流量定義存取權限。

    b. 在「類型」中，選擇「自訂 TCP」或「自訂 UDP」。

    c. 在通訊埠範圍中，輸入允許輸入連線的通訊埠號碼範圍。連接埠範圍必須使用格式`nnnnn[-nnnnn]`，其值在 1026 到 60000 之間。範例：**`1500`** 或 **`1500-20000`**。

    d. 對於 IP 位址範圍，請輸入 IP 位址範圍。使用 CIDR 符號。範例：**`0.0.0.0/0`** (此範例允許任何嘗試連接的人員存取。)
15. (選擇性) 在遊戲工作階段資源設定下，執行下列動作：

    a. 對於遊戲縮放保護政策，請打開或關閉擴展保護。如果 Amazon 託管活動中的遊戲工作階段，Amazon GameLift 不會在縮減事件期間終止具有保護的執行個體。

    b. 針對資源建立限制，請輸入玩家在政策期間可建立的遊戲工作階段數目上限。

16. 選擇下一步。
17. (選擇性) 輸入「索引鍵」和「值」配對，將標籤新增至組建。選擇 [下一步] 繼續進行叢集建立檢閱。
18. 選擇 建立 。Amazon GameLift 會將 ID 指派給新叢集，並開始叢集啟用程序。您可以在 Fleets (叢集) 頁面追蹤新叢集的狀態。

無論叢集狀態為何，您都可以隨時更新叢集的中繼資料和組態。如需詳細資訊，請參閱管理您的亞馬遜GameLift車隊。您可以在叢集達到 ACTIVE 狀態後更新叢集容量。如需詳細資訊，請參閱擴展亞馬遜GameLift託管容量。您也可以新增或移除遠端位置。

AWS CLI

若要使用建立叢集AWS CLI，請開啟命令列視窗並使用`create-fleet`指令。如需 `create-fleet` 命令的詳細資訊，請參閱《AWS CLI 命令參考》中的 `create-fleet`。

以下顯示的範例 `create-fleet` 請求會建立含以下特點的新機群：

- 叢集使用 c5.large 隨需執行個體搭配適用於所選遊戲組建的作業系統。
- 它會將指定的遊戲伺服器組建部署到下列位置，而該組建必須處於 [就緒] 狀態：
  - 我們-西 2（主場地區）
  - SA-東 -1（遠程位置）
- TLS 憑證產生已啟用。
- 機群中的每個執行個體將同時執行 10 個相同的遊戲伺服器程序，讓每個執行個體可同時代管高達 10 個遊戲工作階段。
- 在每個執行個體上，Amazon GameLift 允許同時啟用兩個新的遊戲工作階段。如果他們還沒有準備好在 300 秒內接待玩家，它也會終止任何啟動遊戲工作階段。
- 在這個機群中的執行個體代管的所有遊戲工作階段已開啟遊戲工作階段保護。
- 個別玩家可以在 15 分鐘期間內建立三個新遊戲工作階段。
- 此叢集上託管的每個遊戲工作階段都有一個連線點，該連線點位於指定的 IP 位址和連接埠範圍內。
- Amazon GameLift 將此叢集的指標新增至指`EMEAfleets`標群組 (在本範例中) 結合 EMEA 區域中所有叢集的指標。

```
aws gamelift create-fleet \
    --name SampleFleet123 \
```

```
    --description "The sample test fleet" \
    --ec2-instance-type c5.large \
    --region us-west-2 \
    --locations "Location=sa-east-1" \
    --fleet-type ON_DEMAND \
    --build-id build-92f061ed-27c9-4a02-b1f4-6f85b2385620 \
    --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=33435,IpRange=0.0.0.0/0,Protocol=UDP"
 "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" \
    --metric-groups  "EMEAfleets"
```

如果建立叢集請求成功，Amazon 會 GameLift 傳回一組叢集屬性，其中包括您請求的組態設定和新的叢集 ID。Amazon GameLift 接著會啟動叢集啟用程序，並將叢集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

創建一個 Amazon GameLift Anywhere 車隊

使用 Amazon GameLift 將您環境中的硬體整合到您的 Amazon GameLift 遊戲託管中。Amazon GameLift 在一個Anywhere叢集中向 Amazon GameLift Anywhere 註冊您的硬件。您可以在分房系統Anywhere和遊戲工作階段佇列中整合和管理 EC2 叢集，以管理配對和遊戲配置。

如需使用 Amazon 測試遊戲伺服器的詳細資訊 GameLift Anywhere，請參閱使用亞馬遜機GameLiftAnywhere隊測試您的整合。

若要開始使用，請參閱第 5 Amazon 的開發支持 GameLift 版或更新版本，並檢閱下列使用 Amazon GameLift Anywhere 叢集的概念。

自訂位置

Amazon GameLift Anywhere 叢集使用自訂位置來代表基礎設施的實體位置。

裝置註冊

若要讓 Amazon GameLift Anywhere 叢集與您的運算資源通訊，請先註冊您的裝置。您可以使用此`RegisterCompute`操作從 Amazon 開 GameLift AWS 發套件完成裝置註冊。此作業會使用裝置的 IP 位址將其與叢集位置建立關聯，並與 Amazon 通訊 GameLift。

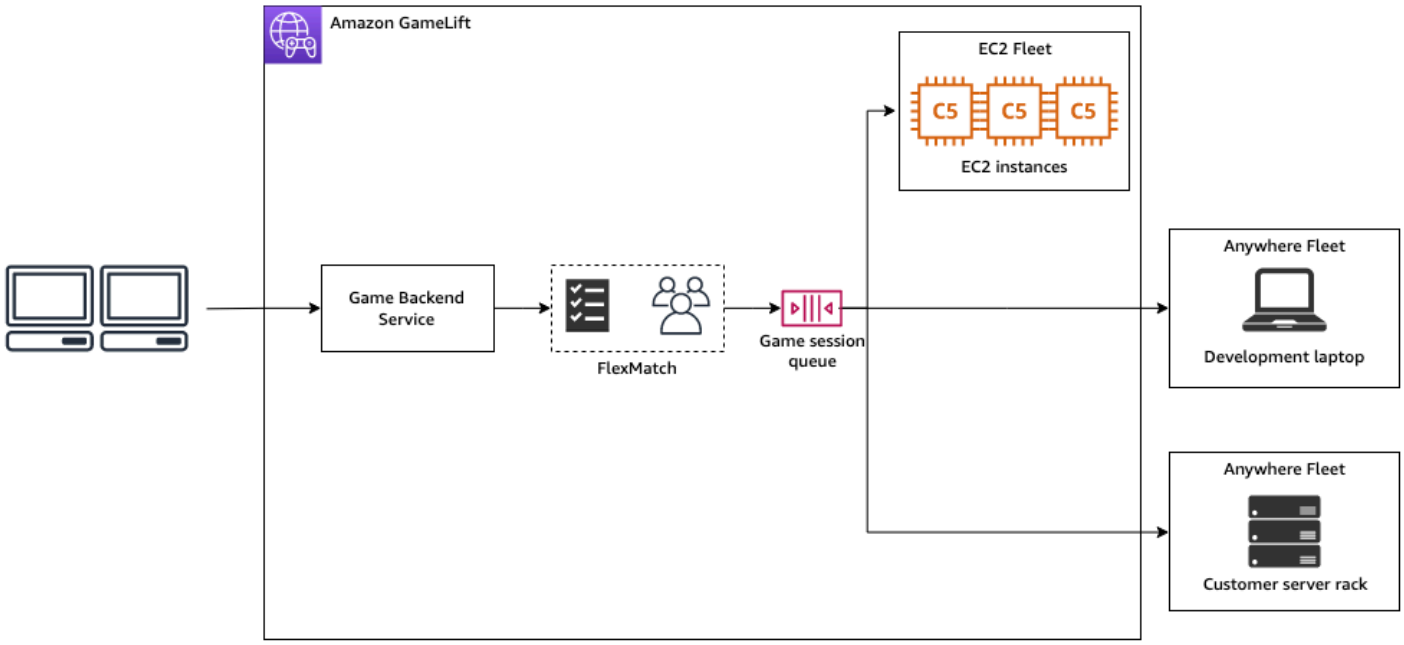
驗證令牌

當您在運算上初始化遊戲伺服器時，Amazon GameLift Server SDK 會使用身份驗證令牌向 Amazon 驗證您的遊戲伺服器 GameLift。您可以為同一計算上的所有遊戲服務器重複使用相同的身份驗證令牌，直到身份驗證令牌到期時間為止。要檢索身份驗證令牌，請調用 `get-compute-auth-token` AWS Command Line Interface （ AWS CLI ）命令。根據需要將令牌傳遞給每個遊戲服務器。

遊戲工作階段

計算上的每個遊戲會話使用在將計算註冊到叢集位置時創建的相同身份驗證令牌。

下圖顯示使用 FlexMatch 配對和多個艦隊的遊戲工作階段佇列。這些機隊包括一個帶有 C5 執行個體的 EC2 叢集、具有開發筆記型電腦的Anywhere機群，以及具有客戶託管伺服器機架的Anywhere叢集。



主題

- 建立自訂位置
- 建立叢集
- 註冊您的運算
- 執行伺服器處理序
- 建立遊戲階段
- 遷移到受管 EC2

建立自訂位置

若要開始在您的運算資源上託管遊戲，請建立描述運算所在位置的自訂位置。

Console

建立自訂位置

1. 打開 Amazon GameLift 控制台。
2. 在功能窗格的 [主機] 下，選擇 [位置]。
3. 在 Locations (位置) 頁面上，選擇 Create location (建立位置)。
4. 在「建立位置」對話方塊中，執行下列操作：

a. 輸入「地點」名稱。這會標示 Amazon GameLift 用來在Anywhere叢集中執行遊戲的硬體位置。Amazon GameLift 將您的自定義位置的名稱附加到自定義位置。

b. (選擇性) 將標籤作為鍵值配對新增至您的自訂位置。針對您要新增的每個標籤選擇「新增標籤」。

c. 選擇建立。

AWS CLI

使用create-location指令建立自訂位置。location-name標記了 Amazon GameLift 用於在Anywhere車隊中運行遊戲的硬件位置。建立自訂位置時，地點名稱的開頭必須為custom-。

```
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"
    }
}
```

建立叢集

使用 Amazon GameLift 主控台或 AWS CLI 建立Anywhere叢集。

建立新的艦隊後，Anywhere艦隊的狀態會從移NEW至ACTIVE。當它達到ACTIVE狀態時，艦隊已準備好主持遊戲工作階段。如需建立叢集問題的協助，請參閱偵錯亞馬遜GameLift叢集問題。

Console

若要建立Anywhere叢集

1. 打開 Amazon GameLift 控制台。
2. 在功能窗格的 [主機] 底下，選擇 [叢集]。
3. 在 Fleets (機群) 頁面上，選擇 Create fleet (建立機群)。

4. 在 [運算類型] 步驟上，選擇 Anywhere，然後選擇 [下一步]。
5. 在 [叢集詳細資料] 步驟中，定義詳細資料，然後選擇 [下一步]。
6. 在 [自訂位置] 步驟中，選取您建立的自訂位置，然後選擇 [下一步]。Amazon GameLift 會自動選取房屋 AWS 區域 做為您要建立叢集的區域。您可以使用首頁「地區」來存取和使用您的資源。
7. 完成剩餘的叢集建立步驟，然後選擇「提交」以建立Anywhere叢集。

AWS CLI

使用`create-fleet`指令建立Anywhere叢集。在中包含您的自訂位置`locations`。Amazon GameLift 會在您的本地區域和您提供的自訂位置建立叢集。在下列範例中，取代*`FleetName`*並*`custom-location-1`*使用您自己的資訊。變數`custom-location-1`是在建立自訂位置步驟中建立的位置名稱。

```
aws gamelift create-fleet \
--name FleetName \
--compute-type ANYWHERE \
--locations "Location=custom-location-1"
```

範例輸出

```
{
    "FleetAttributes": {
        "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
        "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
        "Name": "HardwareAnywhere",
        "CreationTime": "2023-02-23T17:57:42.293000+00:00",
        "Status": "ACTIVE",
        "MetricGroups": [
            "default"
        ],
        "CertificateConfiguration": {
            "CertificateType": "DISABLED"
        },
        "ComputeType": "ANYWHERE"
    }
}
```

註冊您的運算

若要在您建立的叢集中註冊計算資源，請使用register-compute指令。請將*fleet-id*之前步驟中fleet-id傳回的或主控台叢集詳細資料頁面中的叢集 ARN 取代。將和*ip-address*取*compute-name*代為計算資源的 IP 位址。

> **Note**
>
> 我們建議您使用register-compute與遊戲伺服器分開的get-compute-auth-token指令碼或程序管理員呼叫和指令。

```
aws gamelift register-compute \
    --compute-name HardwareAnywhere \
    --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --ip-address 10.1.2.3 \
    --location custom-location-1
```

範例輸出

```
{
    "Compute": {
        "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
        "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
        "ComputeName": "HardwareAnywhere",
        "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333: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-east-1.api.amazongamelift.com"
    }
}
```

執行伺服器處理序

1. 從您建立的叢集取得運算資源的驗證 Token。

您的遊戲伺服器使用身份驗證令牌在 Amazon 進行驗證 GameLift。每個身份驗證令牌都有到期時間。若要繼續使用運算資源來託管您的遊戲伺服器，請在到期前擷取新的驗證權杖。

> **Note**
>
> Amazon GameLift 建議使用`register-compute`與遊戲服務器分開的腳本或進程管理器調用和`get-compute-auth-token`命令。

在下列範例中，請以先前步驟中建立之叢集的 ARN 或叢集識別碼取代。`fleet-id`以您在上一個步驟中使用`register-compute`指令建立的計算名稱取代。`compute-name`

```
aws gamelift get-compute-auth-token \
    --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --compute-name HardwareAnywhere
```

輸出範例：

```
{
    "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
    "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
    "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"
}
```

2. 執行遊戲伺服器執行檔的執行個體。

若要執行您的遊戲伺服器，請呼叫`InitSDK()`並傳遞您的伺服器參數來初始化您的遊戲伺服器。如需詳細資訊，請參閱 ServerParameters。

伺服器 SDK 輸入：

```
//Define the server parameters
ServerParameters serverParameters = new ServerParameters(
    webSocketUrl=wss://us-east-1.api.amazongamelift.com,
```

```
    processId=PID1234,
    hostId=HardwareAnywhere,
    fleetId=arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445,
    authToken=0c728041-3e84-4aaa-b927-a0fb202684c0);

//InitSDK establishes a connection with GameLift's websocket server for
 communication.
var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

3. 伺服器程序準備好主持遊戲工作階段後，請ProcessReady()從遊戲伺服器呼叫 Amazon GameLift。如需處理參數的詳細資訊，請參閱 ProcessParameters

```
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
   this.OnStartGameSession,
   this.OnProcessTerminate,
   this.OnHealthCheck,
   this.OnUpdateGameSession,
   port=1024,
   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);
```

建立遊戲階段

1. 將邏輯添加到您的遊戲服務器，以便您的服務器進程響應onStartGameSession()消息ActivateGameSession()。此操作沒有參數，但它會向 Amazon GameLift 發送確認，表明您的服務器收到並接受了創建遊戲會話消息。

```
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();
}
```

2. 在遊戲用戶端後端服務中，使用start-matchmaking、start-game-session-placement或create-game-session指令開始遊戲工作階段。

```
aws gamelift create-game-session \
    --fleet-id arn:aws:gamelift:us-east-1:682428703967:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --name GameSession1 \
    --maximum-player-session-count 2 \
    --location custom-location-1
```

輸出範例：

```
GameSession {
  FleetId = arn:aws:gamelift:us-east-1:682428703967:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445,
  GameSessionId = 4444-4444,
  Name = GameSession1,
  Location = custom-location-1,
  IpAddress = 10.2.3.4,
  Port = 1024,
  ...
}
```

Amazon GameLift 會將onStartGameSession()訊息傳送到您註冊的伺服器程序。訊息包含上一步的GameSession物件，其中包含遊戲屬性、遊戲工作階段資料、分房系統資料，以及有關遊戲工作階段的更多資訊。

3. 遊戲工作階段完成後，結束遊戲伺服器程序。

伺服器 SDK 輸入：

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
   Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

4. 通過調用啟動另一個遊戲服務器進程ProcessReady(processParams)。

遷移到受管 EC2

在您開發遊戲伺服器並準備好開始生產之後，您就可以讓 Amazon GameLift 管理您的硬體。若要遷移到受管 EC2 叢集，請將您的組建上傳到 Amazon， GameLift 然後建立受管 EC2 叢集。如需有關上傳組建和設定叢集的詳細資訊，請參閱將自訂伺服器組建上傳到亞馬遜 GameLift和建立亞馬遜 GameLift 受管叢集。

管理您的亞馬遜GameLift車隊

使用 Amazon GameLift 主控台或 AWS CLI 更新叢集設定、變更遠端位置或刪除叢集。

更新叢集組態

您可以使用 Amazon GameLift 主控台或 AWS CLI 更新可變叢集屬性、連接埠設定和執行階段組態。若要變更縮放限制，請參閱使用亞馬遜自動擴展車隊容量 GameLift。

Amazon GameLift console

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇「叢集」。
2. 選擇您要更新的機隊。叢集必須處於ACTIVE狀態，才能進行編輯。
3. 在 [叢集詳細資料] 頁面的下列任一區段中，選擇 [編輯]。
   - 車隊設定
     - 變更機群屬性，如 Name (名稱) 和 Description (說明) 等。
     - 新增或移除指標群組，Amazon CloudWatch 用來追蹤多個叢集的彙總 Amazon GameLift 指標。
     - 更新資源建立限制設定。
     - 開啟或關閉遊戲工作階段的防護。
   - 執行階段組態 — 您可以變更執行階段組態的下列任何設定，並新增或移除執行階段組態。
     - 變更遊戲伺服器的啟動路徑。
     - 新增、移除或變更選用的 Launch 參數。
     - 變更遊戲伺服器執行的並行處理序數目。
   - 遊戲工作階段啟動 — 透過更新最大並行遊戲工作階段啟動次數和新啟動逾時，變更您希望伺服器處理序執行和主控遊戲工作階段的方式
   - EC2 連接埠設定 — 更新允許對叢集輸入存取的 IP 位址和連接埠範圍。

4. 選擇「確認」以儲存變更。

AWS CLI

使用下列 AWS CLI 指令更新叢集：

- update-fleet-attributes
- update-fleet-port-settings
- update-runtime-configuration

更新車隊位置

您可以使用 Amazon GameLift 主控台或 AWS CLI 新增或移除叢集的遠端位置。您無法變更艦隊的所在地區。

Amazon GameLift console

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇「叢集」。
2. 選擇您要更新的機隊。叢集必須處於ACTIVE狀態，才能進行編輯。
3. 在 [機隊詳細資訊] 頁面上，選擇 [位置] 索引標籤以檢視車隊的位置。
4. 若要新增遠端位置，請選擇 [新增]，然後選取要部署執行個體的目標位置。此清單不包含叢集執行個體類型無法使用的執行個體。
5. 選取新地點後，選擇 [新增]。Amazon GameLift 將新地點新增至清單，狀態設定為NEW。GameLift然後，Amazon 會在每個新增的位置開始佈建執行個體，並準備託管遊戲工作階段。
6. 若要從叢集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。
7. 選取一或多個叢集後，選擇 [移除]。移除的位置會保留在清單中，狀態設定為DELETING。GameLift然後，亞馬遜開始終止已刪除位置的活動的過程。如果有主控遊戲工作階段的作用中執行個體，Amazon 會GameLift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 指令更新叢集位置：

- create-fleet-locations
- delete-fleet-locations

刪除叢集

您可以在不再需要的時候刪除叢集。刪除叢集會永久移除與遊戲工作階段和玩家工作階段相關的所有資料，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**
>
> 如果叢集具有 VPC 對等連線，請先透過呼叫來要求授權。CreateVpcPeeringAuthorizationAmazon GameLift 會在刪除叢集期間刪除 VPC 對等連線。

您可以使用 Amazon GameLift 主控台或 AWS CLI 工具刪除叢集。

Amazon GameLift console

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇「叢集」。
2. 選擇您要刪除的叢集。您只能刪除處於ACTIVE或ERROR狀態的叢集。
3. 選擇 刪除 。
4. 在 [刪除叢集] 對話方塊中，輸入以確認刪除**delete**。
5. 選擇 刪除 。

AWS CLI

使用下列 AWS CLI 命令刪除叢集：

- delete-fleet

在 Amazon GameLift 叢集中新增別名

Amazon GameLift 別名用於抽象叢集指定。艦隊名稱告訴 Amazon GameLift 在為玩家創建新的遊戲會話時，在哪裡搜索可用資源。使用別名而非特定的叢集 ID，藉由變更別名的目標位置，順暢地將玩家流量從一個叢集切換到另一個叢集。

別名的路由策略有兩種類型：

- 簡單 — 將玩家流量路由至指定的叢集 ID。您可以隨時更新別名的機群 ID。
- 終端機 — 將訊息傳回給用戶端。例如，您可以將正在使用 out-of-date 客戶端的玩家引導到可以獲得升級的位置。

艦隊的使用壽命有限，在遊戲生命週期中更換艦隊的原因有很多。您無法更新艦隊的遊戲伺服器組建或變更現有叢集上的特定運算資源屬性。相反，使用更改創建新艦隊，然後將玩家切換到新艦隊。使用別名切換機群對遊戲影響不大，對玩家更是沒有影響。

別名在不使用佇列的遊戲中很有用。切換佇列機群很簡單，只需建立新機群，將此機群加入佇列，並移除舊機群，而這些玩家都不會看到。相反地，不使用佇列的遊戲用戶端必須指定與 Amazon GameLift 服務通訊時要使用的叢集。如果沒有別名，艦隊交換器需要更新您的遊戲代碼，並可能將更新的遊戲客戶端分發給玩家。

更新別名指向的艦隊 ID 時，最多需要 2 分鐘的轉換期間，別名上的遊戲工作階段可能會出現在舊艦隊中。

建立新的別名

您可以使用 Amazon GameLift 主控台 (如此處所述) 建立別名，或使用 AWS CLI 命令建立別名。

1. 在 Amazon 主 GameLift 控台的導覽窗格中，選擇別名。
2. 在別名索引標籤中，選擇建立別名。我們建議在您的別名中加入叢集類型。這使得在檢視別名清單時更容易識別叢集類型。
3. 在 [建立別名] 頁面的 [別名詳細資訊] 底下，執行下列動作：
   a. 在名稱中，輸入別名名稱。
   b. 對於說明，請輸入簡短描述以進行識別。
   c. 選擇簡單或端子佈線類型。
4. (選擇性) 在「標籤」下，輸入「鍵」和「值」配對，將標籤新增至別名。

5. 選擇建立。

編輯別名

您可以使用 Amazon GameLift 主控台或 AWS CLI 命令更新- 別名來編輯別名。

1. 在 Amazon 主 GameLift 控台的導覽窗格中，選擇別名。
2. 在「別名」頁面上，選擇要編輯的別名。
3. 在別名頁面上選擇編輯。
4. 您可在 Edit alias (編輯別名) 頁面編輯下列項目：
   - 別名名稱 — 別名的易記名稱。
   - 說明 — 別名的簡短說明。
   - 類型 — 玩家流量的路由策略。請選擇 Simple (簡單) 變更相關機群，或選擇 Terminal (終止) 編輯終止訊息。
5. 選擇儲存變更。

偵錯亞馬遜GameLift叢集問題

本主題針對 Amazon GameLift 受管主機解決方案提供叢集組態問題的指導。如需進一步的疑難排解，您可以在機群啟用後遠端存取機群執行個體。請參閱 遠端連線至 Amazon GameLift 叢集執行個體。

機群建立問題

建立叢集後，Amazon GameLift 服務會啟動工作流程，在每個叢集的位置部署新執行個體，並準備執行您的遊戲伺服器。如需詳細描述，請參閱亞馬遜GameLift車隊創建如何運作。艦隊無法主持遊戲工作階段和玩家，直到它達到作用中狀態。本節討論阻止艦隊活躍的最常見問題。

下載和驗證

在此階段，如果解壓縮的建置檔案發生問題、安裝指令碼無法執行，或是執行階段組態中指定的可執行檔未包含在建置檔案中，叢集建立可能會失敗。Amazon GameLift 提供與這些問題相關的日誌。

如果日誌未顯示出問題，問題有可能是因為內部服務錯誤。在這種情況下，請再次嘗試建立機群。如果問題仍存在，請考慮重新上傳遊戲建置 (以防檔案已損毀)。您也可以聯繫亞馬遜GameLift支持或在論壇上發布問題。

建置

在建置階段期間導致失敗的問題，幾乎絕對是因為遊戲建置檔案和/或安裝指令碼的問題。確認上傳到 Amazon GameLift 的遊戲建置檔案可以安裝在執行適當作業系統的機器上。請務必使用全新的作業系統安裝，而不是現有的開發環境。

啟用

最常見的叢集建立問題會在啟用階段期間發生。在此階段，會測試許多元素，包括遊戲伺服器的可行性、執行階段組態設定，以及遊戲伺服器使用 Server SDK 與 Amazon GameLift 服務互動的能力。叢集啟用期間會發生的常見問題包括：

伺服器程序無法啟動。

首先，請檢查您是否已在叢集執行時間組態中正確設定啟動路徑和選用的啟動參數。您可以使用 [叢集詳細資料] 頁面詳細資訊) 區段或呼叫AWS CLI指令來檢視叢集目前的執行階段組態describe-runtime-configuration。如果執行時間組態看起來正確，請檢查遊戲組建檔案和/或安裝指令碼的問題。

伺服器程序已啟動，但叢集無法啟用。

如果伺服器處理序啟動並成功執行，但叢集未移至作用中狀態，則可能的原因是伺服器處理序無法通知 Amazon GameLift 已準備好主持遊戲工作階段。檢查您的遊戲伺服器是否正確呼叫伺服器 API 動作 `ProcessReady()` (請參閱初始化伺服器處理序)。

VPC 對等互連請求失敗。

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describe-vpc-peering-connections叫來追蹤對等連線要求的成功或失敗。請務必檢查是否存在有效的 VPC 對等授權 (describe-vpc-peering-authorizations因為授權僅在 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 主控台 (編輯叢集的容量分配設定) 或呼叫AWS CLI指令來變更叢集的執行階段組態update-runtime-configuration。

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是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其建立 VPC 對等連線的叢集 (請參閱。適用於亞馬遜的 VPC 對等互連 GameLift 之所以發生這種情況，是因為刪除叢集的程序也包括刪除叢集的 VPC 和任何 VPC 對等連線。 您必須先透過呼叫亞馬遜GameLift服務 API CreateVpcPeeringAuthorization() 或使用 AWS CLI 命令來取得授權`create-vpc-peering-authorization`。一旦您擁有授權，您便可以刪除叢集。

實時服務器車隊問題

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：

- 車隊的實時服務器不會拾取腳本更新。
- 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。

這幾乎可以肯定是無法在實時腳本`processEnding`中成功調用的結果。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，運行遊戲會話的實時服務器永遠不會釋放以啟動新會話，並且只有在新的實時服務器運行時才能啟動新的遊戲會話。此外，Realtime 指令碼的更新不會影響已經執行的遊戲工作階段，只會影響遊戲工作階段。

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。如實時服務器腳本示例中所示，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，如果您確實陷入這種情況，則有幾種解決方法可以解決您的實時服務器。訣竅是觸發實時服務器進程（或基礎艦隊實例）重新啟動。在此情況下，會GameLift自動為您關閉遊戲工作階段。釋放實時服務器後，他們可以使用最新版本的實時腳本開始新的遊戲會話。

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。這將清除所有現有的遊戲會話，並讓您從最近更新的實時腳本重新開始。
- 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 叢集執行個體。

如果您選擇不包括在實時腳本`processEnding`中調用的方式，即使艦隊處於活動狀態並且遊戲會話啟動時，也可能會發生一些棘手的情況。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，實時服務器不會獲取任何腳本更新。

遠端連線至 Amazon GameLift 叢集執行個體

您可以連線到作用中 Amazon GameLift 受管 EC2 叢集中的任何執行個體。存取執行個體的常見原因包括：

- 解決遊戲伺服器整合的問題
- 微調您的執行階段設定和其他叢集特定設定
- 獲取實時遊戲服務器活動，例如日誌跟踪。
- 使用實際的玩家流量運行基準測試工具。
- 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：

- 您可以連線至作用中叢集中的執行個體。非作用中的叢集 (啟動或處於錯誤狀態) 可能會在短時間內存取。如需叢集啟用問題的說明，請參閱偵錯亞馬遜GameLift叢集問題。
- 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階段組態繼續啟動和停止伺服器處理序。它激活並主持遊戲會話。它可能會因應縮減事件或其他事件而關閉。
- 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使用中遊戲工作階段和連線的玩家。

下列指示說明如何使用AWS命令列介面 (CLI) 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 Amazon GameLift 服務 API 參考中所述。

收集實例數據

收集下列資訊：

- 您要連線的執行個體 ID。您可以使用執行個體識別碼或 ARN。
- 在執行個體上使用的 Amazon GameLift 伺服器 SDK 版本。伺服器 SDK 與執行個體上執行的遊戲組建整合。

若要擷取例證資料

下列步驟假設您有想要連線的執行個體的受管 EC2 叢集 ID。

1. 取得運算名稱。

   呼叫受管 EC2 叢集的清單運算，以取得叢集中所有使用中運算的清單。若為單一位置叢集，請指定叢集識別碼或 ARN。若為多位置叢集，請指定叢集識別碼或 ARN 以及位置。對於受管 EC2 叢集，計算是 EC2 執行個體，傳回的屬性`ComputeName`為執行個體 ID。例如：

   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.uwp57060n1k6dnlnw49b78hg1
   west-2.amazongamelift.com",
         "ComputeStatus": "Active",
         "Location": "sa-east-1",
         "CreationTime": "2023-07-09T22:51:45.931000-07:00",
         "OperatingSystem": "AMAZON_LINUX",
   ```

```
      "Type": "c4.large"
    }
  ]
}
```

2. 尋找伺服器 SDK 版本。

   服務器 SDK 版本是構建資源的屬性。

   a. describe-fleet-attributes使用叢集 ID 呼叫以取得叢集的組建 ID 和 ARN。

   b. 使用構建 ID 或 ARN 調用描述構建以獲取構建的服務器 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_2",
  "ServerSdkVersion": "5.1.1",
  . . .
}
```

Connect 至執行個體 (伺服器 SDK 5)

如果您要連接的執行個體是使用伺服器 SDK 5.x 版執行遊戲組建，請使用下列指示使用 Amazon EC2 Systems Manager (SSM) 連接到執行個體。您可存取執行 Windows 或 Linux 的遠端執行個體。

1. 要求存取執行個體登入資料。當您擁有要連線的執行個體的運算名稱和叢集 ID 時，請呼叫get-compute-access。如果成功，Amazon 會 GameLift 傳回一組用於存取執行個體的臨時登入資料。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. 匯出存取認證。您可以選擇將認證匯出至環境變數，並使用它們為預設使用者設定 AWS CLI。如需詳細資訊，請參閱AWS Command Line Interface使用指南中的用於設定 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. Connect 至叢集執行個體。使用您要連線的執行個體啟動 SSM 工作階段。包括例證的AWS區域或位置。如需詳細資訊，請參閱 Amazon EC2 Systems Manager 使用者指南中的啟動工作階段 (AWSCLI)。使用您在步驟 1 中取得的認證。例如：

```
aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2
```

Connect 至執行個體 (伺服器 SDK 4.x 或更早版本)

如果您要連線的執行個體使用伺服器 SDK 第 4 版或更早版本執行遊戲組建，請遵循下列指示。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 Connect 線至 Windows 執行個體。使用安全殼層用戶端 Connect 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，請使用指令get-instance-access要求存取認證。如果成功，Amazon 會 GameLift 傳回執行個體的作業系統、IP 地址和一組登入資料 (使用者名稱和密鑰)。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 — 若要連線到 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 — 若要連線到 Linux 執行個體，安全殼層需要使用者名稱和私密金鑰。Amazon GameLift 發出 RSA 私鑰並將其作為單個字符串返回，換行符（\n）表示換行符。要使私鑰可用，請執行以下步驟：（1）將字符串轉換為.pem文件，並（2）為新文件設置權限。傳回的登入資料範例：

     ```
     "Credentials": {
     ```

```
    "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEAy7WZhaDsrA1W3mRlQtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7iO5dSrvC7dQkW2duV5QuUdE0QW
\nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
\nG50TCFeOzfl8dqqCP6GzbPaIjiU19xX/azOR9V+tpUOzEL+wmXnZt3/nHPQ5xvD2OJH67km6SuPW
\noPzev/D8V+x4+bHthfSjR9Y7DvQFjfBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrqu
\n/uler7vgIn5m7lN5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
ufGxbL1\nmb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
BOsShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA3OzdXzMQexXVJ1TLZVEH0E7bhlY9d8O1ozR
\noQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfql
+lIp1\nYkriL0DbLXlvRAH+yHPRit2hHOjtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh
+GTfbjsXsOxQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ5OBEKzet9XcKwONBYELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWVOEihvm+xTtmaZlSp//lkq75XDwnU
\nWA8gkn6O3QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
\ngYBjbO+OZk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\noMo7yykABY7Ozd5wQewBQ4AdSlWSX4nGDtsiFxWiI5sKuAAeOCbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vvwKBgF+09VI/1wJBirsDGz9whVWfFPrTkJNvJZzYt69qezxlsjgFKshy
\nWBhd4xHZtmCqpBPlAymEjr/TOlbxyARmXMnIOWIAnNXMGB4KGSyl1mzSVAoQ+fqR+cJ3d0dyPl1j
\njjb0Ed/NY8frlNDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0iOegLDa
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjkpUfVTrW0YFjXkfcrR/V+QFL5OndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
 RSA PRIVATE KEY-----",
    "UserName": "gl-user-remote"
}
```

使用 AWS CLI 時，您可以將-query 和- output 參數納入要求，以自動產生.pem檔案。`get-instance-access`

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 GameLift 叢集組態中授權的任何連接埠存取 Amazon 叢集中的執行個體。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。建立叢集之後，但在啟用之前，您無法更新連接埠設定。如果卡住，請在開啟連接埠設定的情況下重新建立叢集。

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"
```

下列範例會在視窗叢集上開啟連接埠 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
   ```

查看遠程實例上的文件

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。如果執行個體與活躍玩家一起託管遊戲，您可能會冒著遊戲工作階段當機和玩家丟棄的風險，或是中斷遊戲關機程序並造成儲存的遊戲資料和記錄檔發生錯誤的風險。

在託管實例上查找以下資源：

- 遊戲建置檔案。這些文件是您上傳到 Amazon 的遊戲構建 GameLift。它們包括一個或多個遊戲伺服器可執行檔、資產和相依性。遊戲構建文件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。在您指定的任何目錄路徑的game根目錄中找到遊戲伺服器產生的記錄檔。

- Amazon GameLift 託管資源。根目錄`Whitewater`包含 Amazon GameLift 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階段設定。若要變更執行階段組態屬性，請更新叢集的執行階段設定 (請參閱 AWS SDK 作業UpdateRuntimeConfiguration或 AWS CLI update-runtime-configuration)。
- 車隊資料。JSON 檔案包含執行個體所屬叢集的相關資訊，供執行個體上執行的伺服器處理序使用。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 憑證產生的叢集上，請在下列位置尋找憑證檔案，包括憑證、憑證鏈結、私密金鑰和根憑證：
  - 在 Windows 上： `c:\\GameMetadata\Certificates`
  - 在 Linux 上：`/local/gamemetadata/certificates/`

擴展亞馬遜GameLift託管容量

託管容量 (以執行個體計算) 代表 Amazon GameLift 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同時玩家數量。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，而不會在不需要的資源上浪費金錢。如需詳細資訊，請參閱擴展機群容量的規模。

容量會在車隊位置層級進行調整。所有艦隊至少都有一個位置：艦隊的所在地AWS區。檢視或擴充容量時，資訊會依據位置列出，包括叢集的本地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，也可以設定自動調整規模，以在玩家需求變更時動態調整容量。建議您先開啟以目標為基礎的自動縮放選項。基於目標的自動擴展的目標是保持足夠的託管資源以容納當前玩家，再加上一點額外的東西來處理玩家需求的意外高峰。對於大多數遊戲，基於目標的自動縮放提供了高效的擴展解決方案。

本區段中的主題提供有關下列任務的詳細說明資訊：

- 設定容量擴展的最低和最高限制
- 手動設定容量級別
- 使用基於目標的自動縮放
- 管理以規則為基礎的自動調整比例 (進階功能)
- 暫時停用自動縮放

您可以使用 Amazon GameLift 主控台執行大部分的叢集擴展活動。您也可以使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 搭配亞馬遜GameLift服務 API。

在主控台中管理叢集容量

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格中，選擇 [主機]、[叢集]。
3. 在「艦隊」頁面上，選擇作用中叢集的名稱以開啟叢集的詳細資料頁面。
4. 選擇「縮放」頁籤。在此標籤上，您可以：
   - 檢視整個叢集的歷史擴展指標。
   - 檢視和更新每個叢集位置的容量設定，包括擴展限制和目前的容量設定。
   - 更新以目標為基礎的自動調整規模、檢視套用至整個叢集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。

主題

- 設置亞馬遜GameLift容量限制
- 手動設定亞馬遜GameLift叢集的容量
- 使用亞馬遜自動擴展車隊容量 GameLift

設置亞馬遜GameLift容量限制

手動或透過自動擴展擴展 Amazon GameLift 叢集位置的託管容量時，請考慮該位置的擴展限制。所有車隊位置都有定義位置容量允許範圍的最小和最大限制。根據預設，叢集位置的限制至少有 0 個執行個體和最多 1 個執行個體。在擴展叢集位置之前，請先調整限制。

如果您使用的是自動擴展，則最大限制允許 Amazon GameLift 擴展叢集位置以滿足玩家需求，但可以防止失控的託管成本，例如 DDOS 攻擊期間。設定 Amazon CloudWatch 警示以在容量接近最大限制時通知您，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，最低限制也能維持主機的可用性。

您可以在 Amazon GameLift 主控台中設定叢集位置的容量限制，或使用 AWS Command Line Interface (AWS CLI)。

設定容量限制

Console

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格中，選擇 [主機]、[叢集]。
3. 在「艦隊」頁面上，選擇作用中叢集的名稱以開啟叢集的詳細資料頁面。
4. 在 [擴展] 索引標籤的 [擴展容量] 下，選取叢集位置，然後選擇 [編輯]。
5. 在 [編輯擴展容量] 對話方塊中，設定 [最小大小]、[所需執行個體] 和 [最大大小] 的執行個體數
6. 選擇 Confirm (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指令搭配您要變更容量的叢集 ID 和位置。此指令會傳回包FleetCapacity含該位置目前容量設定的物件。判斷新執行個體限制是否可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。在指令行視窗中，使用具有下列參數的update-fleet-capacity指令。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，Amazon 會GameLift傳回叢集 ID。如果新的`max-size`或`min-size`值與目前的`desired-instances`設定衝突，Amazon 會GameLift傳回錯誤訊息。

手動設定亞馬遜GameLift叢集的容量

當您建立新叢集時，Amazon GameLift 會自動將所需的執行個體設定為每個叢集位置的一個執行個體。然後，Amazon GameLift 會在每個位置部署一個新執行個體。若要變更叢集容量，您可以新增以目標為基礎的自動擴展政策，也可以手動設定要用於某個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。

當您不需要自動調整規模或需要將容量保留在指定層級時，手動設定叢集的容量會很有用。手動設定容量只有在您未使用以目標為基礎的自動擴展政策時才有作用。如果您有以目標為基礎的自動擴展政策，它會立即根據自己的擴展規則重設所需的容量。

您可以在 Amazon GameLift 主控台中手動設定容量，或使用 AWS Command Line Interface (AWS CLI)。艦隊的狀態必須處於作用中狀態。

暫停自動縮放

您可以暫停每個叢集位置的所有自動擴展活動。自動擴展暫停後，除非手動變更，否則叢集位置中所需的執行個體數量保持不變。當您暫停某個位置的自動資源調整時，會影響叢集目前的原則，以及您未來可能定義的任何原則。

手動設定機群容量

Console

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格中，選擇 [主機]、[叢集]。
3. 在「艦隊」頁面上，選擇作用中叢集的名稱以開啟叢集的詳細資料頁面。
4. 在 [縮放] 索引標籤的 [暫停自動縮放位置] 下，選取您要暫停自動縮放的每個位置，然後選擇 [暫停]。
5. 在 [擴充容量] 底下，選取您要手動設定的位置，然後選擇 [編輯]。
6. 在 [編輯擴展容量] 對話方塊中，為 [所需的執行個體] 設定偏好的值，然後選擇 [確認]。這會告訴 Amazon 處GameLift於作用中狀態且準備好託管遊戲工作階段的執行個體數量。

Amazon 透過部署其他執行個體或關閉不需要的執行個體來GameLift回應變更。Amazon GameLift 完成此程序後，位置中的作用中執行個體數量會隨之變更，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指令搭配您要變更容量的叢集 ID 和位置。此指令會傳回包FleetCapacity含該位置目前容量設定的物件。判斷執行個體限制是否可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。使用指update-fleet-capacity令搭配叢集 ID、位置和所需執行個體的新值。如果此值落在目前限制範圍之外，您可以使用相同的指令調整極限值。

```
--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，Amazon 會GameLift傳回叢集 ID。如果新的所需執行個體設定超出最小和最大限制，Amazon 會GameLift傳回錯誤訊息。

使用亞馬遜自動擴展車隊容量 GameLift

在 Amazon 中使用自動擴展功能GameLift來動態擴展叢集容量，以回應遊戲伺服器活動。當玩家到達並開始遊戲工作階段時，自動調整功能可以新增更多執行個體；隨著玩家需求減弱，自動擴展可能會終

止不需要的執行個體 Auto Scaling 是最大限度地減少託管資源和成本的有效方法，同時仍然可以提供流暢，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 何時擴展或縮小規模。擴展政策有兩種類型：以目標為基礎和以規則為基礎。基於目標的方法-目標跟踪-是一個完整的解決方案。我們建議將其作為最簡單，最有效的選擇。基於規則的擴展政策要求您定義自動擴展決策過程的各個方面，這對於解決特定問題非常有用。此解決方案最適合作為基於目標的自動縮放的補充。

您可以使用 Amazon GameLift 主控台、AWS Command Line Interface (AWS CLI) 或AWS開發套件來管理以目標為基礎的自動擴展。您只能使用AWS CLI或 AWS SDK 管理以規則為基礎的自動擴展，不過您可以在主控台中檢視以規則為基礎的擴展政策。

主題

基於目標的自動縮放

Amazon 的目標型自動擴展功能會根據叢集指`PercentAvailableGameSessions`標GameLift調整容量層級。此指標代表車隊在玩家需求突然增加時的可用緩衝區。

維持容量緩衝的主要原因，是玩家的等待時間。當遊戲工作階段已準備就緒並等待時，需要幾秒鐘的時間才能讓新玩家進入遊戲工作階段。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。啟動新執行個體和伺服器處理序可能需要幾分鐘的時間。

設定以目標為基礎的自動調整規模時，請指定叢集要維護的緩衝區大小。由於`PercentAvailableGameSessions`測量可用資源的百分比，因此實際緩衝區大小是叢集總容量的百分比。Amazon GameLift 新增或移除執行個體以維持目標緩衝區大小。使用較大的緩衝區，您可以最大限度地減少等待時間，但您也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。

設定以目標為基礎的自動縮放

Console

1. 打開亞馬遜GameLift控制台。
2. 在功能窗格中，選擇 [主機]、[叢集]。
3. 在「艦隊」頁面上，選擇作用中叢集的名稱以開啟叢集的詳細資料頁面。

4. 選擇「縮放」頁籤。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 [擴充容量] 底下，檢查 [最小大小] 和 [最大大小限制] 是否適用於叢集。啟用自動調整規模後，容量會在這兩個限制之間進行調整
6. 在目標型自動調整規模政策中，選擇編輯。
7. 在 [編輯以目標為基礎的自動調整規模政策] 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]。確認設定後，Amazon 會在以目標為基礎的自動擴展政策下新GameLift增以目標為基礎的新政策。

AWS CLI

1. 設定容量限制。使用update-fleet-capacity指令設定限制值。如需詳細資訊，請參閱設置亞馬遜GameLift容量限制。
2. 建立新政策。開啟命put-scaling-policy令列視窗，並搭配原則的參數設定使用命令。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"
```

使用以規則為基礎的原則自動調整

Amazon 中基於規則的擴展政策在自動擴展叢集容量以回應玩家活動時，GameLift提供精細的控制。針對每個原則，您可以將擴展連結至數個叢集指標之一、識別觸發點，以及自訂回應向上擴充或縮減事件。規則型政策對於補充以目標為基礎的擴展以處理特殊情況非常有用。

以規則為基礎的原則會指出下列情況：「如果叢集量度在特定時間長度內達到或超過臨界值，則會依指定的數量變更叢集的容量。」 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開發套件或 AWS Command Line Interface (AWS CLI) 搭配 Amazon GameLift 服務 API 建立、更新或刪除以規則為基礎的政策。您可以在 Amazon GameLift 主控台中檢視所有使用中政策。

若要暫時停止叢集的所有擴展政策，請使用AWS CLI命令stop-fleet-actions。

若要建立或更新以規則為基礎的資源調整政策 ()AWS CLI：

1. 設定容量限制。使用update-fleet-capacity指令設定其中一個或兩個限制值。如需詳細資訊，請參閱設置亞馬遜GameLift容量限制。
2. 建立新政策。開啟命令列視窗，並搭配原則的參數設定使用put-scaling-policy命令。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
```

若要刪除以規則為基礎的資源調整政策，請使用：AWS CLI

- 開啟命令列視窗，並使用具有叢集 ID 和原則名稱的delete-scaling-policy命令。

  範例：

```
aws gamelift delete-scaling-policy \
    --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --name "Scale up when AGS<50"
```

自動縮放規則的語法

若要建構以規則為基礎的資源調整政策陳述式，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，只要叢集的額外容量少於處理 50 個新遊戲工作階段所需的容量，本政策聲明就會啟動擴展事件：

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 Scaling 事件，請將自動擴展政策連結至下列其中一個叢集特定量度。如需完整的量度說明，請參閱適用於艦隊的亞馬遜GameLift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

如果叢集位於遊戲工作階段佇列中，您可以使用下列指標：

- 佇列深度 — 此叢集的待處理遊戲工作階段要求數量是可用的最佳託管位置。
- 等待時間 — 特定於車隊的等待時間。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

告訴 Amazon GameLift 如何將指標資料與臨界值進行比較。有效的比較運算子包括大於 (>)、小於 = <), greater than or equal (>) 以及小於或等於 (<=)。

閾值

當指定的量度值達到或超過閾值時，就會啟動縮放事件。此值一律為正整數。

評估期間

在開始擴展事件之前，量度必須符合或超過評估期間的完整長度臨界值。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

這組變數共同運作，指定 Amazon 在擴展事件開始時GameLift應如何調整叢集的容量。從三種可能的調整類型中選擇：

- 容量變更 — 依指定的執行個體數目增加或減少目前的容量。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，值「-10」會將叢集縮減 10 個執行個體，無論叢集的總大小為何。
- 容量變更百分比 — 依指定的百分比增加或減少目前的容量。將調整值設定為您要增加或減少叢集容量的百分比。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 個執行個體的叢集，變更百分比為「20」會將 10 個執行個體新增至叢集。
- 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。

以規則為基礎的自動縮放提示

下列建議可協助您利用以規則為基礎的原則，充分利用自動調整規模。

使用多個政策

您可以同時為一個叢集建立多個自動擴展政策。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。使用多個策略沒有限制。

對於多個策略，每個策略都會獨立運作。沒有辦法控制縮放事件的順序。例如，如果您有多個推動擴展規模的政策，則玩家活動可能會同時開始多個擴展事件。避免彼此啟動的策略。例如，如果您建立將容量設定在彼此臨界值之外的向上擴充和縮減政策，則可以建立無限迴圈。

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。使用自動縮放時，此功能非常重要。自動調整規模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。為了讓您的自動擴展政策按預期影響容量，請增加最大值。

叢集容量也受到叢集執行個體類型的限制以及AWS 帳戶. 在這些限制和帳戶配額之外，您無法設定最小和最大值。

在容量更改後追蹤指標

在因應自動擴展政策而變更容量後，Amazon 會GameLift等待 10 分鐘，然後再從相同政策回應觸發器。這項等待讓 Amazon 有GameLift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，Amazon 會GameLift根據指標評估政策，並追蹤政策的評估期，該期間會在擴展事件發生後重新啟動。這意味著擴展政策可以在等待時間結束後立即啟動另一個擴展事件。

不同自動擴展政策啟動的擴展事件之間沒有等待時間。

為遊戲會話放置設置亞馬遜GameLift隊列

遊戲工作階段佇列是處理新遊戲工作階段要求以及尋找可用遊戲伺服器來託管這些要求的主要機制。隊列為遊戲開發人員和玩家提供了顯著的好處。其中包含：

- 佇列提供最佳的放置位置。處理遊戲工作階段放置請求時，佇列會使用 Amazon GameLift 演算法根據一組定義的偏好設定來排定佇列位置的優先順序。
- 在價格較低的 Spot 艦隊上託管遊戲。使用佇列最佳化 AWS Spot 叢集的使用，進而大幅降低託管成本。依預設，佇列一律會嘗試在 Spot 叢集中放置新的遊戲工作階段。
- 在高需求期間更快地放置新遊戲。佇列會使用多個可能的位置來放置。這表示如果偏好的放置位置無法使用，則永遠會有後援容量。
- 讓遊戲可用性更具彈性。中斷可能發生。使用多位置佇列時，速度降低或中斷時間不會影響玩家存取您的遊戲。
- 更有效率地運用額外機群容量。為了處理玩家需求的意外激增，佇列可讓您快速存取額外的主機容量。佇列中的叢集位置提供彼此的備份容量。位置根據玩家需求擴展或縮小。

- 取得有關遊戲工作階段位置和佇列效能的指標。Amazon 會GameLift發出佇列指標，包括放置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 主控台或中檢視這些指標CloudWatch。

若要開始使用佇列，請參閱設計遊戲工作階段佇列。

設計遊戲工作階段佇列

本主題說明如何設計佇列，以最小的延遲提供玩家體驗，並有效使用主機資源。如需遊戲工作階段佇列及其運作方式的詳細資訊，請參閱為遊戲會話放置設置亞馬遜GameLift隊列。

這些 Amazon GameLift 功能需要佇列：

- 以 FlexMatch 進行配對
- 搭配 Amazon 使用競價型執行個 GameLift

定義佇列的範圍

您遊戲的玩家人口可能有一群不應該一起玩的玩家。例如，如果您以兩種語言發行遊戲，則每種語言都應該擁有自己的遊戲伺服器。

若要為您的玩家族群設定遊戲階段位置，請為每個玩家區段建立個別的佇列。調整每個佇列的範圍，將玩家置於正確的遊戲伺服器。範圍佇列的一些常見方法包括：

- 按地理位置。在多個地理區域部署遊戲伺服器時，您可能會為每個位置的玩家建立佇列，以減少玩家延遲。
- 通過構建或腳本的變化。如果您的遊戲伺服器有多個版本，您可能會支援無法在相同遊戲工作階段中遊玩的玩家群組。例如，遊戲伺服器組建或指令碼可能支援不同的語言或裝置類型。
- 按事件類型。您可以創建一個特殊的隊列來管理比賽或其他特殊事件的參與者的遊戲。

建立玩家延遲政策

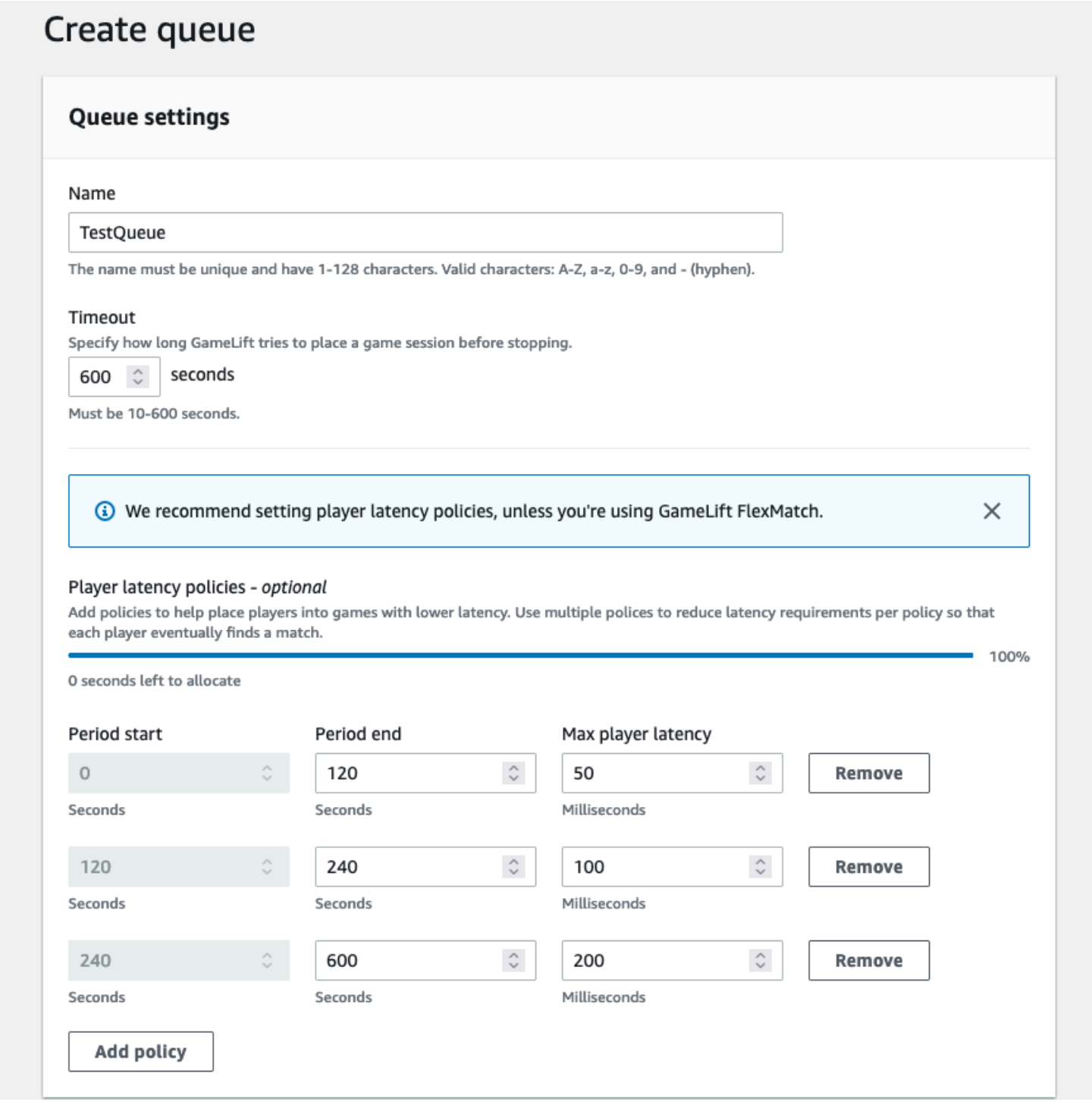
如果您的刊登位置要求包含玩家延遲資料，演算法會在所有玩家平均延遲最低的位置尋找遊戲工作階段。根據平均玩家延遲放置遊戲工作階段，可防GameLift止 Amazon 將大多數玩家置於高延遲的遊戲中。但是，亞馬遜GameLift仍然會使玩家處於極端延遲狀態。若要容納這些玩家，請建立玩家延遲政策。

玩家延遲政策可防GameLift止 Amazon 將要求的遊戲工作階段放置在任何地方，讓請求中的玩家可能會遇到超過最大值的延遲。玩家延遲政策還可以防止 Amazon GameLift 將遊戲工作階段請求與更高延遲的玩家進行比對。

> **Tip**
>
> 若要管理特定延遲規則，例如需要群組中所有玩家的類似延遲，您可以使用 Amazon 建立以延遲GameLiftFlexMatch為基礎的配對規則。

例如，假設此佇列的逾時時間為 5 分鐘，以及下列播放程式延遲政策：

1. 花費 120 秒搜尋所有玩家延遲少於 50 毫秒的位置。
2. 花費 120 秒搜尋所有玩家延遲少於 100 毫秒的位置。
3. 花剩餘的佇列時間，直到逾時搜尋所有玩家延遲少於 200 毫秒的位置。



建立多位置佇列

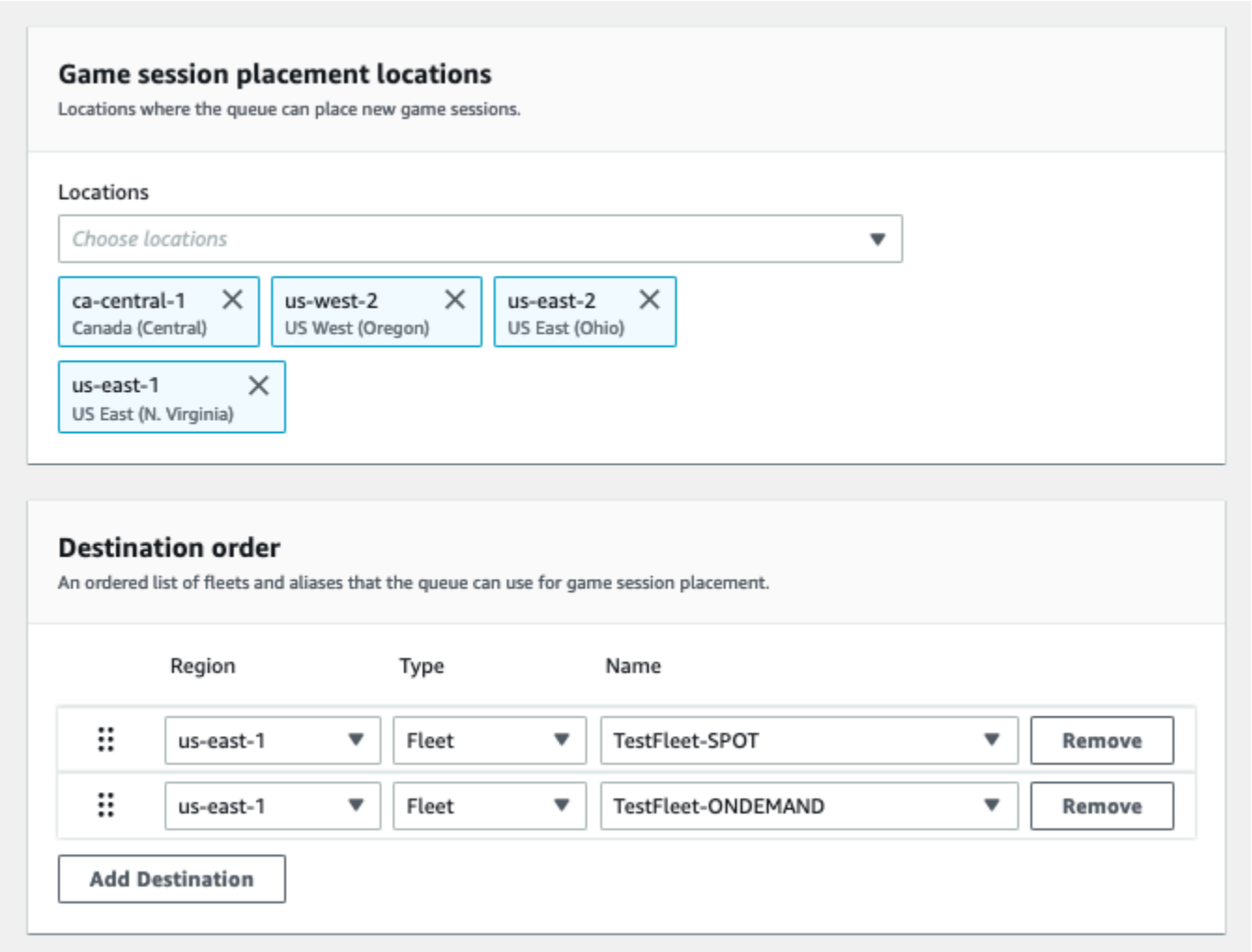
我們建議所有佇列採用多位置設計。這種設計可以提高放置速度和託管恢復性。需要多位置設計才能使用玩家延遲資料，以最小的延遲將玩家投入遊戲工作階段。如果您要建立使用 Spot 執行個體叢集的多位置佇列，請按照中的指示進行。教學課程：設定 Spot 執行個體的遊戲工作階段佇列

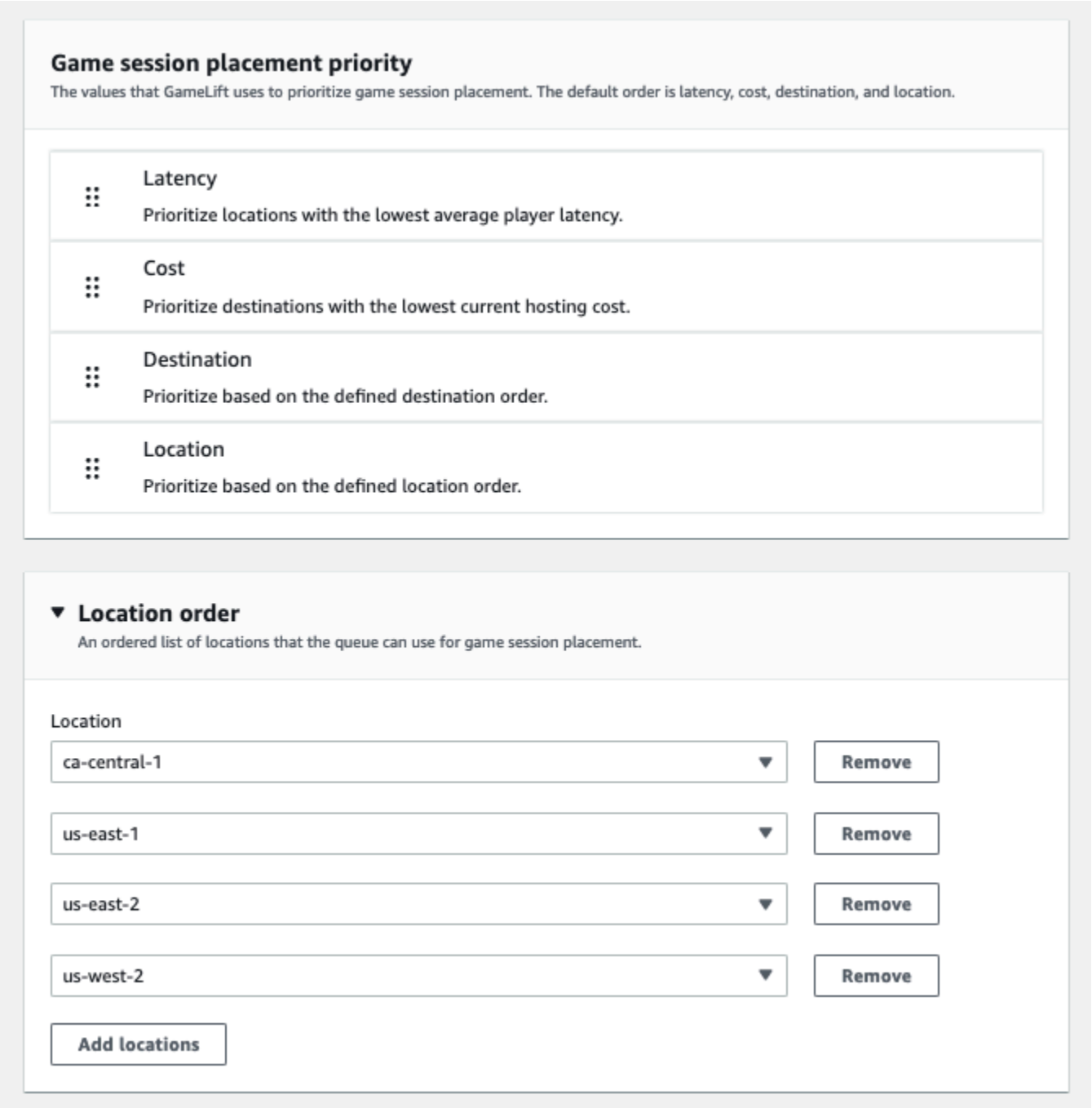
建立多位置佇列的一種方法是將多位置叢集新增至佇列。如此一來，佇列就可以將遊戲工作階段放置在艦隊的任何位置。您還可以添加具有不同配置的其他車隊或主位置以進行冗餘。如果您使用的是多地點 Spot 執行個體叢集，請遵循最佳做法，並包含具有相同位置的隨需執行個體叢集。

下列範例概述設計基本多位置佇列的程序。在此範例中，我們使用兩個叢集：一個 Spot 執行個體叢集和一個隨需執行個體叢集。每個叢集AWS 區域對放置位置都有下列項目：`us-east-1`、`us-east-2`、`ca-central-1`和`us-west-2`。

使用多位置叢集建立基本的多位置佇列

1. 選擇要在其中建立佇列的位置。您可以將佇列放置在靠近部署用戶端服務的位置，將要求延遲降到最低。在此範例中，我們在中建立佇列`us-east-1`。
2. 建立新佇列，並將多位置叢集新增為佇列目的地。目的地順序決定了亞馬遜如何GameLift放置遊戲會話。在此範例中，我們會先列出 Spot 執行個體叢集，第二個列出隨需執行個體叢集。
3. 定義佇列的遊戲工作階段放置優先順序。此順序決定佇列先搜尋可用遊戲伺服器的位置。在此範例中，我們使用預設的優先順序。
4. 定義位置順序。如果您沒有定義位置順序，亞馬遜GameLift會按字母順序使用位置。


Game session placement locations
Locations where the queue can place new game sessions.
Locations
Choose locations
ca-central-1
Canada (Central)
us-west-2
US West (Oregon)
us-east-2
US East (Ohio)
us-east-1
US East (N. Virginia)
Destination order
An ordered list of fleets and aliases that the queue can use for game session placement.
Region
Type
Name
us-east-1
Fleet
TestFleet-SPOT
Remove
us-east-1
Fleet
TestFleet-ONDEMAND
Remove
Add Destination



排定遊戲工作階段位

Amazon GameLift 使用 FleetIQ 演算法，根據一組已排序的準則來決定要放置新遊戲工作階段的位置。您可以使用預設的優先順序，也可以自訂順序。

預設優先順序

對於包含玩家延遲資料的位置要求，FleetIQ 會依照下列預設順序排列遊戲工作階段放置標準的優先順序：

1. 延遲 — 請求中所有玩家的最低平均延遲。
2. 成本 — 如果多個位置的延遲相等，則託管成本最低。託管成本主要取決於執行個體類型和位置的組合。
3. 目的地 — 目的地順序，如果多個位置的延遲和成本相等。FleetIQ 會根據佇列組態中列出的順序排列目的地優先順序。
4. 位置 — 位置順序 (如果延遲、成本和目的地在多個位置相等)。FleetIQ 會根據佇列組態中列出的順序來排定位置的優先順序。

自訂優先順序

若要在 Amazon GameLift 主控台自訂佇列的優先順序，請將優先順序值拖曳到您想要的位置。若要使用 AWS Command Line Interface (AWS CLI) 自訂佇列的優先順序，請搭配--priority-configuration選項使用create-game-session-queue指令。您可以使用此指令來建立新佇列或更新現有佇列。

FleetIQ 演算法會根據預設順序，將未明確提及的任何條件附加到清單末尾。如果您在優先順序組態中包含位置條件，您也必須提供排序的位置清單。

視需要設計多個佇列

視您的遊戲和玩家而定，您可能會想要建立多個遊戲工作階段佇列。當您的遊戲用戶端服務要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。若要協助您判斷是否要使用多個佇列，請考慮：

- 遊戲伺服器的變化。您可以為遊戲伺服器的每個變體建立單獨的佇列。佇列中的所有艦隊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佇列的任何遊戲伺服器上進行遊戲。
- 不同的玩家組。您可以根據玩家群組自訂 Amazon GameLift 放置遊戲工作階段的方式。例如，您可能需要針對需要特殊執行個體類型或執行階段設定的特定遊戲模式自訂佇列。或者，您可能需要一個特殊的隊列來管理錦標賽或其他賽事的位置。
- 遊戲工作階段佇列指標。您可以根據收集遊戲工作階段放置指標的方式來設定佇列。如需詳細資訊，請參閱亞馬遜隊列GameLift指標。

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 主控台或 Amazon 中檢視與佇列相關的指標CloudWatch。如需佇列測量結果的清單和說明，請參閱亞馬遜隊列GameLift指標。

佇列測量結果可提供下列項目的深入解析：

- 整體佇列效能 — 佇列測量結果指出佇列回應放置要求的順利程度。這些指標也可協助您識別刊登位置失敗的時間和原因。對於具有手動調整叢集的佇列，`AverageWaitTime`和指`QueueDepth`標可指示何時應調整佇列容量。
- FleetIQ 演算法效能 — 對於使用 FleetIQ 演算法的放置要求，指標會顯示演算法找到理想遊戲工作階段放置的頻率。刊登位置可以使用玩家延遲最低的資源或成本最低的資源來排定優先順序。還有一些錯誤指標可GameLift以識別 Amazon 找不到理想位置的常見原因。如需指標的詳細資訊，請參閱 監控亞馬遜GameLift與亞馬遜 CloudWatch。
- 地點特定刊登位置 — 針對多地點佇列，量度會依地點顯示成功的刊登位置。對於使用 FleetIQ 演算法的佇列，此資料可提供有用的深入資訊，瞭解玩家活動發生的位置。

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放置位置的速率，請將指`PlacementsSucceeded`標與 FleetIQ 指標結合使用，以獲得最低延遲和最低價格。
- 若要提高佇列尋找理想放置位置的速率，請檢閱下列錯誤指標：
  - 如果高`FirstChoiceOutOfCapacity`，請調整佇列叢集的容量調整。
  - 如果`FirstChoiceNotViable`錯誤指標很高，請查看您的競價型執行個體叢集。當特定執行個體類型的中斷率過高時，Spot 執行個體叢集被視為不可行。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 執行個體叢集。建議您在每個位置加入具有不同執行個體類型的 Spot 執行個體叢集。

Amazon GameLift 遊戲工作階段佇列的最佳實務

以下是一些最佳做法，可協助您建立有效的遊戲工作階段佇列以放置遊戲工作階段。

適用於任何叢集類型之佇列的最佳作法

佇列包含可放置新遊戲工作階段的艦隊目的地清單。每個叢集都可以在多個地理位置部署執行個體。選擇放置時，佇列會選取叢集和叢集位置的組合。您可以為佇列提供一組優先順序，以便在選擇放置時使用。

請考慮下列準則和最佳做法：

- 在涵蓋您玩家的位置新增車隊。您可以在任何可用位置新增叢集和別名。如果您根據回報的玩家延遲進行刊登位置，位置就很重要。
- 為所有艦隊使用別名。為佇列中的每個叢集指派別名，並在佇列中設定目的地時使用別名。
- 對所有艦隊使用相同或類似的遊戲構建或腳本。佇列可能會讓玩家進入佇列中任何艦隊的遊戲階段。玩家必須能夠在任何艦隊的任何遊戲階段中進行遊戲。
- 在至少兩個位置建立車隊。透過將遊戲伺服器託管在至少一個其他位置，您可以減輕地區服務中斷對玩家的影響。您可以縮減備份叢集，並在使用量增加時使用自動擴展來增加容量。
- 排定遊戲工作階段的優先順序 佇列會根據數個元素 (包括目標清單順序) 來排列放置選項的優先順序。
- 在與用戶端服務相同的位置建立佇列。將佇列放在靠近用戶端服務的位置，可以將通訊延遲降到最低。
- 使用具有多個位置的車隊。使用佇列篩選器設定可避免佇列將遊戲工作階段放置在指定的位置。您可以使用至少兩個具有不同主地位置的多地點艦隊，以減輕區域停機期間遊戲放置的影響。
- 對所有叢集使用相同的 TLS 憑證設定。連線至遊戲階段作業的遊戲用戶端必須具有相容的通訊協定。

Spot 叢集佇列的最佳做法

如果您的佇列包含 Spot 叢集，請設定彈性佇列。這可利用 Spot 叢集節省成本，同時將遊戲工作階段中斷的影響降到最低。如需正確建置叢集和遊戲工作階段佇列以搭配 Spot 叢集使用的說明，請參閱。教學課程：設定 Spot 執行個體的遊戲工作階段佇列如需 Spot 執行個體的詳細資訊，請參閱搭配 Amazon 使用競價型執行個 GameLift。

除了上一節中的一般最佳作法之外，請考慮以下特定於 Spot 的最佳作法：

- 在每個位置建立至少一個隨需叢集。隨選艦隊為您的玩家提供備份遊戲伺服器。您可以保持備份叢集縮小到需要為止，並在 Spot 叢集無法使用時，使用自動擴充功能來增加隨需容量。
- 在一個位置的多個 Spot 叢集中選取不同的執行個體類型。如果有一個 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 叢集。最佳做法是選擇廣泛使用的執行個體類型，並在相同的系列中使用執行個體類型 (例如 m5.large、m5.xlarge、m5.2xlarge)。使用 Amazon 主GameLift 控台檢視執行個體類型的歷史定價資料。

建立遊戲工作階段佇列

佇列的用途是使用最佳可用的託管資源，將新遊戲工作階段放置橫跨多個機群和區域。若要進一步了解如何建置遊戲的佇列，請參閱 設計遊戲工作階段佇列。

在遊戲用戶端，新遊戲工作階段是透過使用放置請求經由佇列啟動。進一步了解中的遊戲工作階段位置建立遊戲階段。

更新佇列中的佇列目的地時，會有短暫的轉換期間 (最多 30 秒)，在此期間放置在佇列目的地的遊戲工作階段仍可能會出現在舊艦隊中。

Console

1. 在 Amazon 主GameLift控台的導覽頁面中，選擇 [佇列]。
2. 在 Queues (佇列) 頁面上，選擇 Create queue (建立新佇列)。
3. 在 [建立佇列] 頁面的 [佇列設定] 下，執行下列動作：

   a. 在名稱中，輸入佇列名稱。

   b. 對於「超時」，請輸入您希GameLift望 Amazon 在停止之前嘗試放置遊戲會話的長時間。Amazon 會GameLift搜尋任何叢集上的可用資源，直到請求逾時為止。

   c. (選擇性) 對於玩家延遲政策，請輸入 Amazon GameLift 在定義的最大延遲內尋找資源的時間長度。新增其他原則以逐步放鬆最大延遲。若要新增其他原則，請選擇 [新增原則]。
4. 在「遊戲工作階段放置位置」下，選取要包含在佇列中的位置。默認情況下，包括所有位置。佇列中的所有叢集都必須具有相同的憑證組態。所有艦隊都應執行與使用佇列的遊戲用戶端相容的遊戲組建。
5. 在「目的地順序」下，將一或多個目的地新增至佇列。

   a. 選擇 Add destination (新增目的地)。

   b. 選取目的地所在的位置。

   c. 選擇目的地的類型。

   d. 在產生的機群或別名名稱清單中，選取要新增者。

   e. 如果您有多個目的地，請將六個點圖示拖曳至目的地左側，以設定預設順序。Amazon 在搜尋目的地以尋找可GameLift用資源以放置新遊戲工作階段時，會使用此順序。
6. 對於遊戲工作階段放置優先順GameLift序，請新增並拖曳「延遲」、「成本」、「目的地」和「位置」值，以定義 Amazon 如何設定佇列中叢集的優先順序。如需有關設定叢集優先順序的詳細資訊，請參閱。排定遊戲工作階段位

7. 將位置新增至您的位置順序，並將其拖曳至佇列應使用的優先順序。如果位置是遊戲會話放置的最後優先級，亞馬遜將其GameLift用作決勝者。
8. (選擇性) 在事件通知設定下，執行下列動作：
   a. 選取或建立 SNS 主題以接收與放置相關的事件通知。如需事件通知的詳細資訊，請參閱設定遊戲工作階段位置的事件通知。
   b. 新增自訂事件資料以附加至此佇列所建立的事件。
9. (選擇性) 新增標籤。有關標記的詳細資訊，請參閱標記AWS資源。
10. 選擇 建立 。

AWS CLI

Example 建立佇列

下列範例會使用這些設定建立遊戲工作階段佇列：

- 超時五分鐘
- 兩個車隊目的地
- 篩選條件以僅允許us-east-1、中的位置us-east-2。 us-west-2，以及 ca-central-1
- 根據成本排定目的地的優先順序，然後根據定義的順序排列位置。

```
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="LOCATION","DESTINATION",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**
>
> 您可以使用叢集或別名 ID 呼叫describe-fleet-attributes或描述別名來取得叢集和別名 ARN 值。

如果`create-game-session-queue`請求成功，Amazon 會GameLift傳回具有新佇列組態的GameSessionQueue物件。您現在可以使用將請求提交到佇列StartGameSessionPlacement。

Example 使用玩家延遲政策建立佇列

下列範例會使用這些設定建立遊戲工作階段佇列：

- 十分鐘逾時
- 三個艦隊目的地
- 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 \
                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 會GameLift傳回具有新佇列組態的GameSessionQueue物件。

設定遊戲工作階段位置的事件通知

您可以使用事件通知來監視個別放置請求的狀態。我們建議您為所有具有大量位置活動的遊戲設定事件通知。

設定事件通知有兩種選項。

- 讓亞馬遜使用隊列將事件通知GameLift發佈到亞馬遜簡單通知服務（亞馬遜 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 Amazon 發出的遊戲工作階段放置事件清單GameLift，請參閱遊戲工作階段安置事。

設定 SNS 主題

GameLift若要讓 Amazon 將遊戲工作階段佇列產生的所有事件發佈到主題，請將通知目標欄位設定為主題。

設定亞馬遜GameLift事件通知的 SNS 主題

1. 登入 AWS Management Console，並在 https://console.aws.amazon.com/sns/v3/home 開啟 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 [存取原則] 底下，執行下列動作：

    a. 選擇高級方法。

    b. 將 JSON 物件的下列粗體區段新增至現有政策。

```
{
  "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:your_region:your_account: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:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
        }
      }
    }
  ]
}
```

c. (選擇性) 將條件新增至資源策略，將其他存取控制新增至主題。

4. 請選擇 建立主題 。

5. 建立 SNS 主題後，請在佇列建立期間將其新增至佇列，或編輯現有佇列以新增它。

設定具有伺服器端加密的 SNS 主題

使用伺服器端加密 (SSE)，您可以將機密資料儲存在加密主題中。SSE 會使用 AWS Key Management Service (AWS KMS) 中管理的金鑰來保護 Amazon SNS 主題中訊息的內容。如需使用 Amazon SNS 進行伺服器端加密的詳細資訊，請參閱 Amazon 簡單通知服務開發人員指南中的靜態加密。

若要設定具有伺服器端加密的 SNS 主題，請檢閱下列主題：

- 在AWS Key Management Service開發人員指南中建立金鑰
- 在 Amazon 簡易通知服務開發人員指南中的主題啟用 SSE

建立 KMS 金鑰時，請使用下列 KMS 金鑰原則：

```
{
    "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 會GameLift自動將所有遊戲工作階段放置事件發佈到EventBridge。EventBridge您可以使用設定規則，將事件路由至目標進行處理。例如，您可以設定規則，將事件路由`PlacementFulfilled`至處理連線至遊戲工作階段之前的工作的AWS Lambda函數。如需詳細資訊EventBridge，請參閱什麼是AmazonEventBridge？ 在亞馬遜EventBridge用戶指南。

以下是與 Amazon GameLift 佇列搭配使用的一些EventBridge規則範例：

匹配所有亞馬遜GameLift隊列中的事件

```
{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ]
```

```
}
```

匹配來自特定隊列的事件

```
{
    "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：設定 Spot 執行個體的遊戲工作階段佇列

簡介

本教學課程說明如何針對低成本 Spot 叢集上部署的遊戲設定遊戲工作階段配置。競價型艦隊需要額外的步驟來維持玩家持續的遊戲伺服器可用性。

目標對象

本教程適用於希望使用 Spot 艦隊託管自定義遊戲服務器或實時服務器的遊戲開發人員。

您將學到什麼

- 定義您的遊戲工作階段佇列服務的玩家群組。
- 建立艦隊基礎結構以支援遊戲工作階段佇列的範圍。
- 為每個叢集指派別名以抽象叢集 ID。
- 建立佇列、新增叢集，並排定 Amazon GameLift 放置遊戲工作階段的位置優先順序。
- 新增玩家延遲政策，協助將延遲問題降到最低。

先決條件

在建立遊戲工作階段放置的叢集和佇列之前，請先完成下列工作：

- 檢閱 亞馬遜如何GameLift工作。
- 將您的遊戲服務器與亞馬遜集成GameLift。

- 將您的遊戲服務器構建或實時腳本上傳到亞馬遜GameLift。
- 規劃您的車隊配置。

步驟 1：定義佇列的範圍

在本教學課程中，我們會為具有一個遊戲伺服器組建變化的遊戲設計佇列。遊戲發行時，我們將在兩個地點推出：亞太區域 (首爾) 和亞太區域 (新加坡)。由於這些位置彼此很接近，因此延遲對我們的玩家來說並不是問題。

在這個例子中，有一個玩家段，這意味著我們創建一個隊列。未來，當我們在北美發行遊戲時，我們可以建立第二個隊列，以北美玩家為範圍。

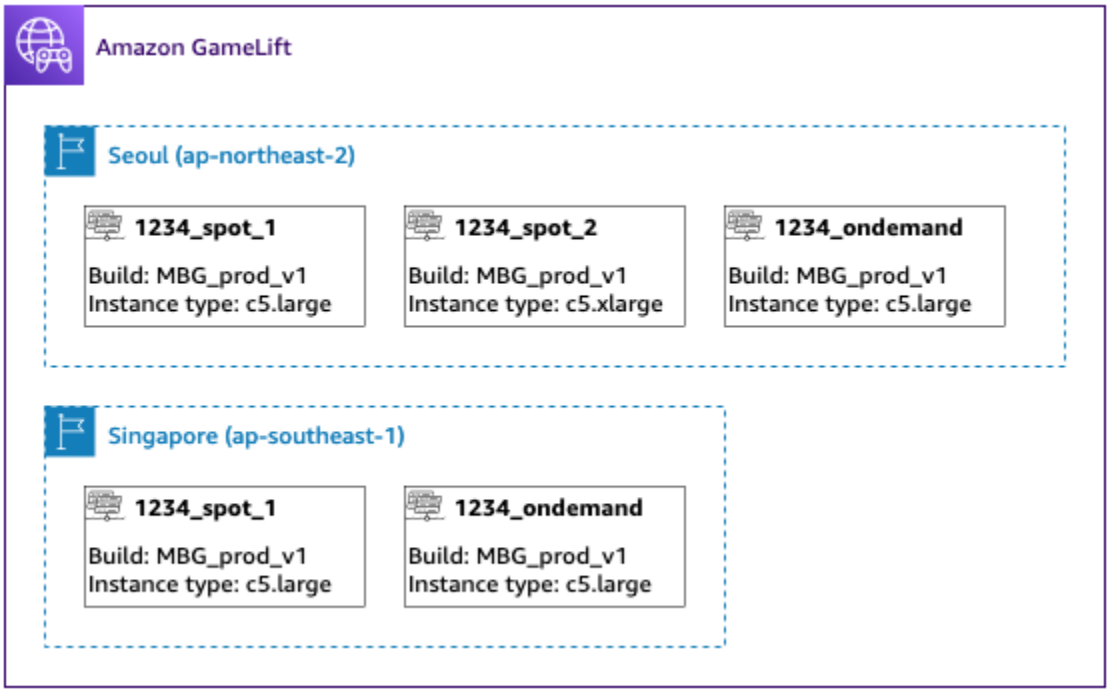
如需詳細資訊，請參閱定義佇列的範圍。

步驟 2：建立競價型叢集基礎架構

使用符合您在中定義範圍的遊戲伺服器組建或指令碼，在步驟 1：定義佇列的範圍位置建立車隊。

在本教學中，我們建立了兩個位置基礎結構，每個位置至少包含一個 Spot 叢集和一個隨需叢集。每個艦隊都會部署相同的遊戲伺服器組建。此外，我們預計首爾地區的玩家流量會更重，因此我們在那裡增加了更多 Spot 車隊。

下圖顯示 Spot 叢集基礎架構範例，在 ap-Northest-2 (首爾) 位置有 3 個車隊，在 ap-Spot-東南 -1 (新加坡) 位置有 2 個車隊。兩個叢集中的所有執行個體都使用組建 MBG_prod_v1。AP-North-2 中的叢集包含下列叢集組態：執行個體類型為 c5.large 的叢集 1234_spot_1、執行個體類型為 c5.xlarge 的叢集 1234_spot_2，以及執行個體類型為 c5.large 的叢集 1234_ondemand。AP 東南 -1 中的叢集包含下列叢集組態：執行個體類型為 c5.large 的叢集 1234_spot_1，以及執行個體類型為 c5.large 的叢集 1234_ondemand。

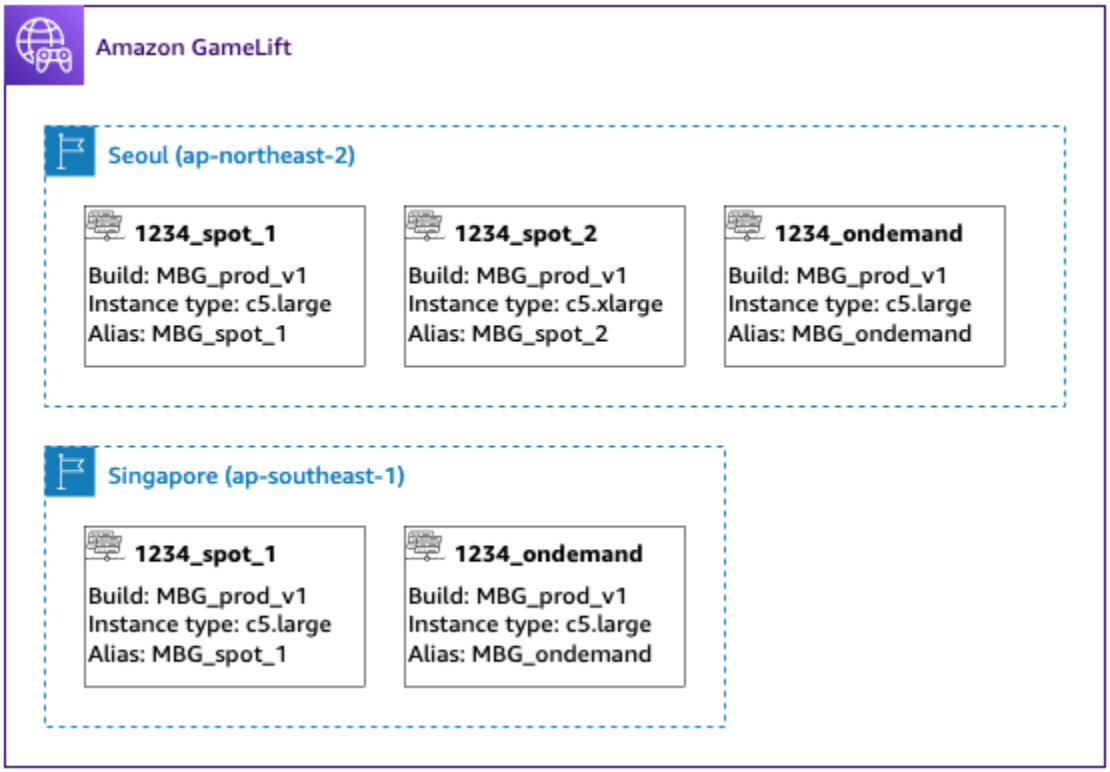


步驟 3：為每個叢集指派別名

為基礎結構中的每個叢集建立新的別名。別名抽象車隊身份，使定期更換車隊更換效率。如需建立別名的詳細資訊，請參閱在 Amazon GameLift 叢集中新增別名。

我們的車隊基礎設施有五個車隊，因此我們使用路由策略創建五個別名。我們在亞太區域 (首爾) 地點需要三個別名，在亞太區域 (新加坡) 地點需要兩個別名。

下圖顯示步驟二中所述的競價型叢集基礎結構，並在每個叢集中新增了別名。叢集 1234_1 具有別名 MBG 點 _1，叢集 1234_SPOT 2 具有別名 MBG 點 _2，而叢集 1234 按需求具有別名 MBG。



如需詳細資訊，請參閱建立多位置佇列。

步驟 4：建立包含目的地的佇列

建立遊戲工作階段佇列並新增您的艦隊目的地。如需建立佇列的詳細資訊，請參閱建立遊戲工作階段佇列。

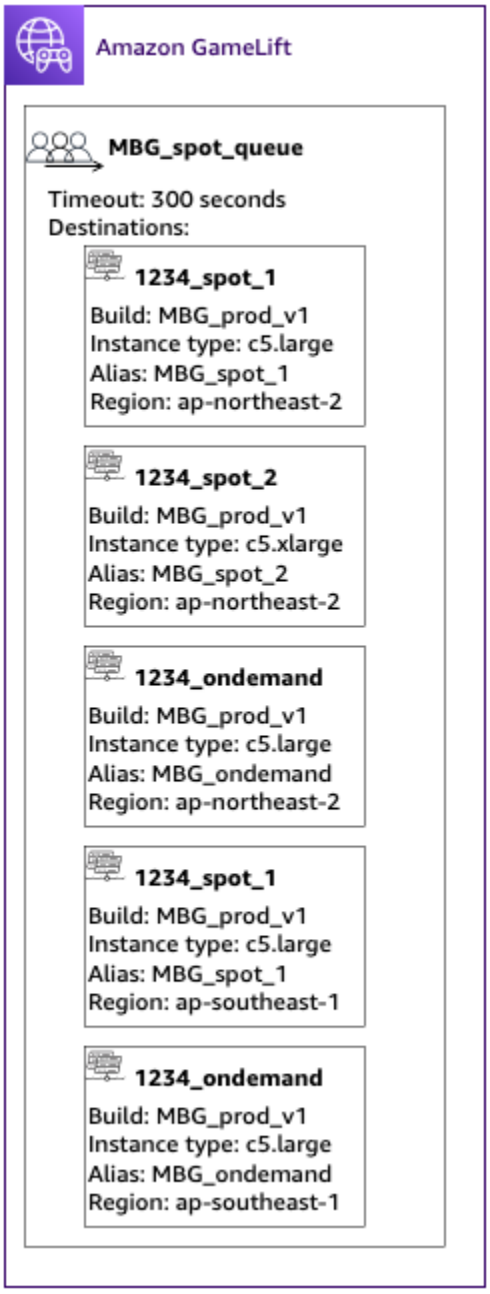
建立佇列時：

- 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。
- 現在跳過有關玩家延遲政策的部分。我們將在下一步中介紹此內容。
- 排定佇列中的艦隊優先順序。使用 Spot 叢集時，我們建議採用下列其中一種方法：
  - 如果您的基礎架構使用的是主要位置，且叢集位於第二個位置進行備份，請先依據位置排定叢集的優先順序，然後依機群類型排定
  - 如果您的基礎架構同樣使用多個位置，請依叢集類型排定叢集的優先順序，並將 Spot 叢集置於佇列的頂端。

在本教程中，我們創建一個名稱的新隊列**MBG_spot_queue**，並添加所有五個艦隊的別名。然後，我們首先按位置排定位置的優先順序，然後按車隊類型排序

根據此配置，此隊列總是嘗試將新的遊戲會話放入首爾的 Spot 艦隊中。當這些艦隊已滿時，佇列會使用首爾隨需叢集上的可用容量作為備份。如果三支首爾艦隊都無法使用，亞馬遜將在新加坡艦隊上GameLift放置遊戲會議。

下圖顯示逾時 300 秒和已排定優先順序目的地的佇列。目的地的順序如下：在東北-2 中的 1234_spot_1，在 AP-東北 -2 中為 1234_spot_2，在 AP-東北-2 中的 1234 點 _ 點 1，在 AP-東南 1 中為 1234 點 _ 點 1，以及在 AP-東南 1 的需求為 1234。



步驟 5：將延遲限制新增至佇列

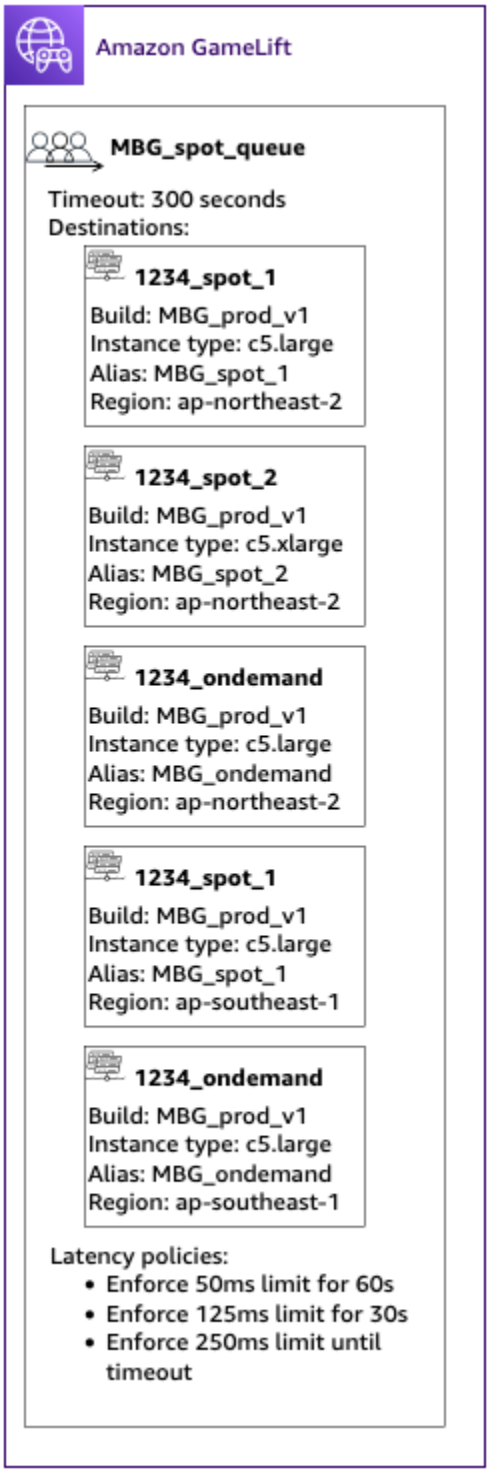
我們的遊戲會在遊戲會話放置請求中包含延遲信息。我們還有一個玩家派對功能，可以為一群玩家創建遊戲會話。我們可以讓玩家等待更長時間才能進入具有理想遊戲體驗的遊戲。我們的遊戲測試顯示了以下觀察結果：

- 低於 50 毫秒的延遲是理想的。
- 遊戲在延遲超過 250 毫秒的情況下無法遊玩。
- 玩家會在大約一分鐘內變得不耐煩。

對於我們的佇列，我們會在 300 秒的逾時時間內新增政策陳述式，以限制允許的延遲。原則陳述式會逐漸允許較大的延遲值，最多可達 250 毫秒。

透過此原則，我們的佇列會在第一分鐘尋找具有理想延遲 (低於 50 毫秒) 的放置位置，然後放寬限制。在玩家延遲為 250 毫秒或更高的情況下，佇列不會放置位置。

下圖顯示步驟 4 中新增了玩家延遲政策的佇列。播放器延遲政策狀態，強制執行 60 毫秒的 50 毫秒限制，在 30 秒強制執行 125 毫秒限制，並強制執行 250 毫秒限制，直到超時。



總結

恭喜您！以下是您完成的事情：

- 您有一個遊戲工作階段佇列的範圍是您的玩家人口。

- 您的佇列有效地使用 Spot 叢集，並且在 Spot 中斷發生時具有彈性。
- 您的隊列會優先考慮艦隊，以獲得頂級玩家體驗。
- 隊列有延遲限制，以保護玩家免受不良遊戲體驗的影響。

您現在可以使用佇列為其服務的玩家安排遊戲階段。向這些玩家提出遊戲工作階段放置要求時，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放置要求的詳細資訊，請參閱建立遊戲階段、或為即時伺服器整合遊戲用戶端。

接下來的步驟：

- 設計您自己的佇列。
- 建立佇列。
- 在您的遊戲用戶端中使用佇列。

使用管理資源 AWS CloudFormation

您可以使用AWS CloudFormation來管理您的亞馬遜GameLift資源。在 AWS CloudFormation 中，您可以一個範本來製作每個資源的模型，然後使用此範本來建立資源。若要更新資源，您可以對範本進行變更，並使用 AWS CloudFormation 來實作更新。您可以將資源組織成邏輯群組，稱為堆疊和堆疊集。

使AWS CloudFormation用維護您的 Amazon GameLift 託管資源可提供更有效的方式來管理資AWS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。

如需有關 AWS CloudFormation 的詳細資訊，請參閱《使用者指南》AWS CloudFormation。若要檢視 Amazon GameLift 資源的範本資訊，請參閱 Amazon 資GameLift源類型參考資料。

最佳實務

如需使用的詳細指引AWS CloudFormation，請參閱《使AWS CloudFormation用指南》中的AWS CloudFormation最佳做法。此外，這些最佳實務與 Amazon 具有特殊關聯性GameLift。

- 始終如一地管理您的資源AWS CloudFormation。如果您在資源之外更改AWS CloudFormation資源將與資源模板不同步。
- 使用 AWS CloudFormation 堆疊和堆疊集來有效地管理多個資源。

  - 使用堆疊來管理已連線資源的群組。例如，包含組建的堆疊、參考組建的叢集，以及參考叢集的別名。如果您更新範本以取代組建，請AWS CloudFormation取代連接至組建的叢集。 AWS CloudFormation然後更新現有的別名以指向新叢集。如需詳細資訊，請參閱《使用指南》中的〈AWS CloudFormation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。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudFormation使用堆疊組合〉。
- 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使用兩個叢集設定範本，一個使用 Spot 執行個體的叢集，以及一個使用隨需執行個體的叢集。
- 當您管理多個地點的資源時，請將特定位置的資源和全球資源分組到單獨的堆疊中。
- 將您的全球資源放在使用該資源的服務附近。佇列和配對配置等資源往往會收到來自特定來源的大量請求。將您的資源放在靠近這些請求的來源之處，您可以將請求傳送時間降至最低，並改善整體效能。
- 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

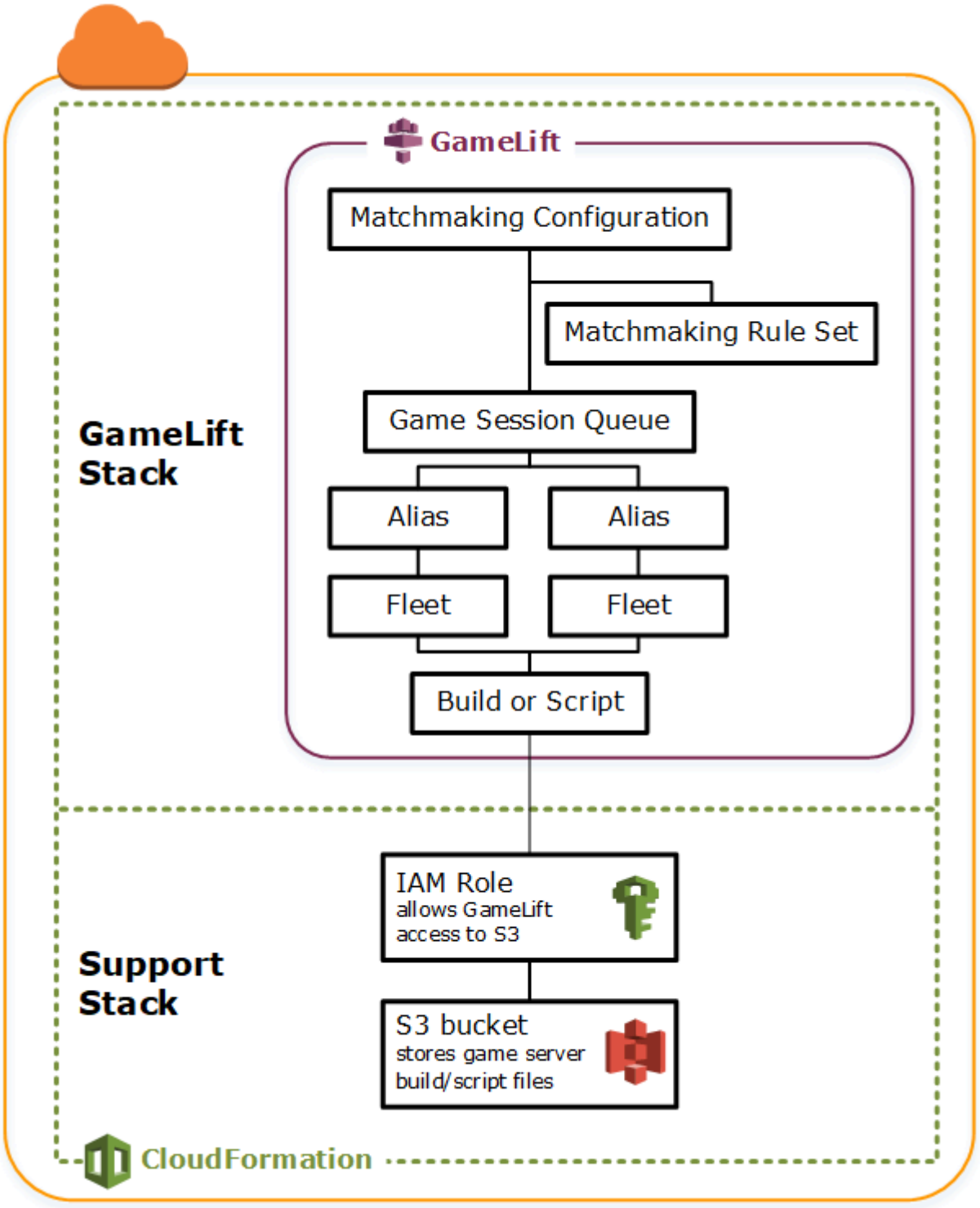
我們建議您在為 Amazon GameLift 資源設定AWS CloudFormation堆疊時使用下列結構。根據您是在一個位置還是多個位置部署遊戲，您的最佳堆疊結構會有所不同。

堆疊單一位置

若要在單一位置管理 Amazon GameLift 資源，我們建議使用雙堆疊結構：

- 支援堆疊 — 此堆疊包含 Amazon 資GameLift源所依賴的資源。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。該堆棧還應包括 IAM 角色，該角色在創建 Amazon GameLift 構建或腳本資源時GameLift授予 Amazon 存儲桶從 S3 存儲桶檢索文件的權限。此堆疊也可能包含與您的遊戲搭配使用的其他AWS資源，例如 DynamoDB 資料表、亞馬遜紅移叢集和 Lambda 函數。
- Amazon GameLift 堆疊 — 此堆疊包含所有 Amazon 資GameLift源，包括建置或指令碼、一組叢集、別名和遊戲工作階段佇列。 AWS CloudFormation使用存放在 S3 儲存貯體位置的檔案建立組建或指令碼資源，並將組建或指令碼部署到一或多個叢集資源。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。如果您使用的是配FlexMatch對，此堆疊還包含配對配對配置和規則集。

下圖說明在單一 AWS 區域中部署資源的雙堆疊結構。

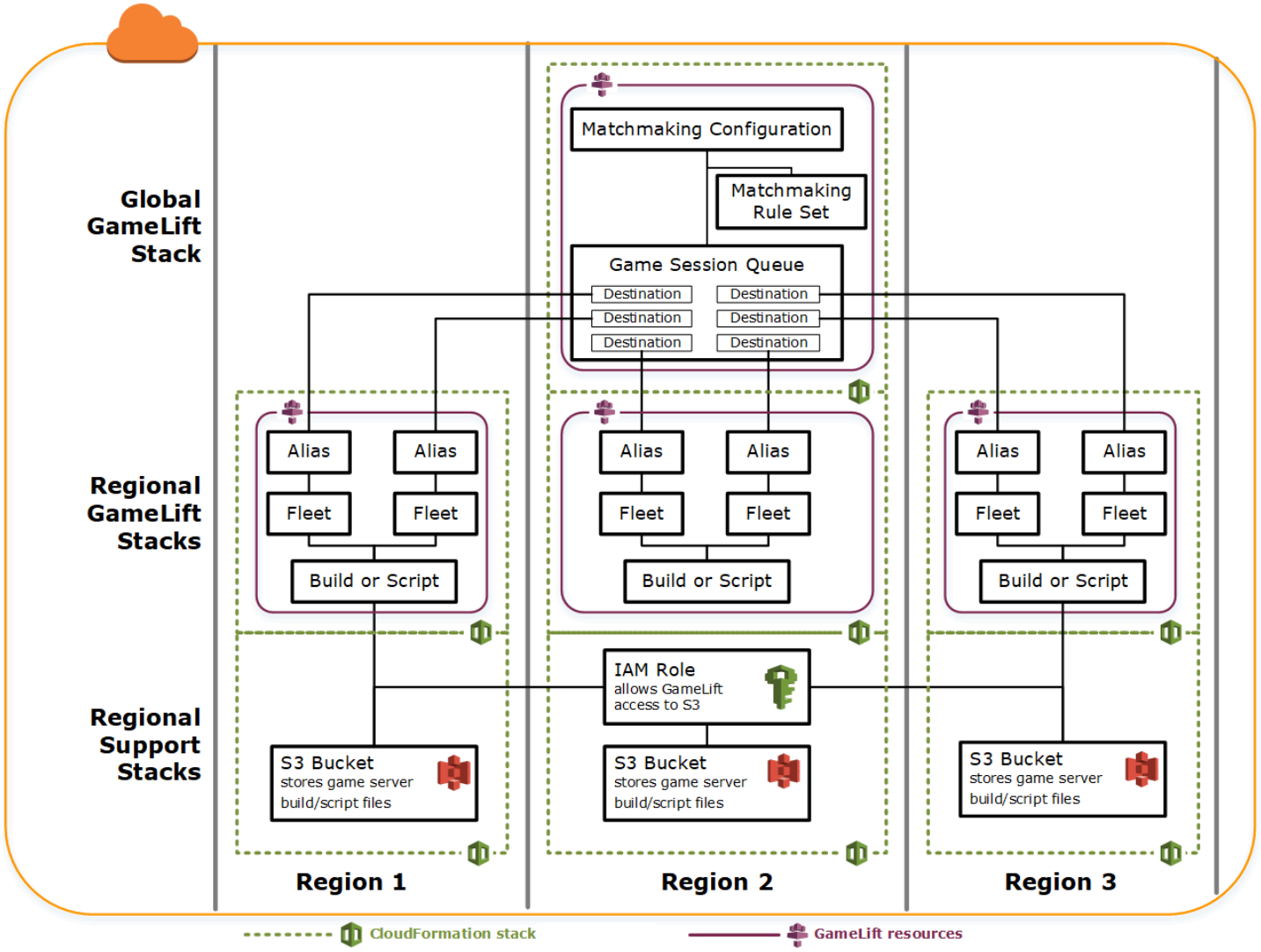


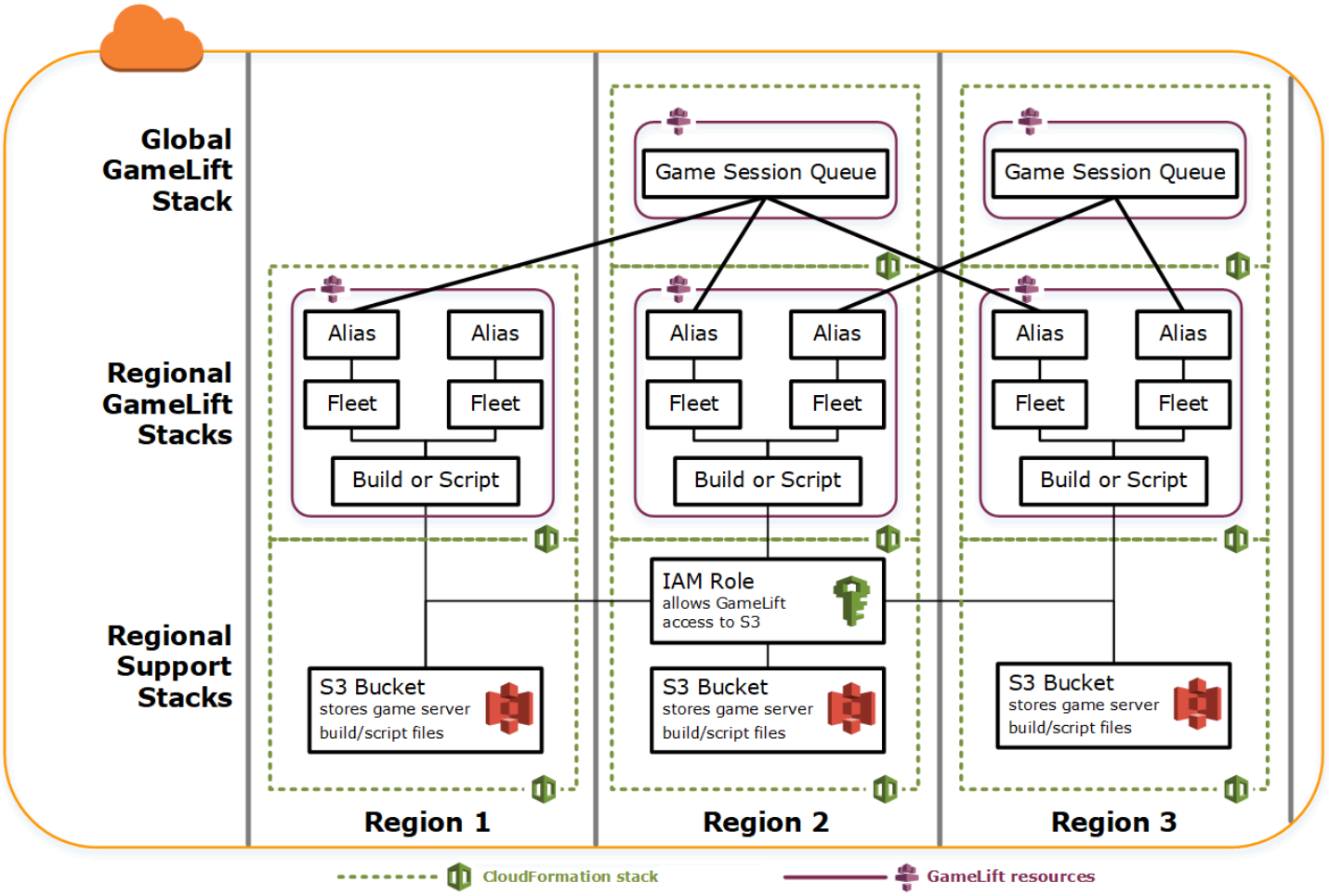
多個區域的堆疊

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 叢集) 只能參考相同區域中的其他資源。其他資源 (例如 Amazon GameLift 佇列) 不受區域限制。若要管理多個區域的 Amazon GameLift 資源，我們建議使用以下結構。

- 區域支援堆疊 — 這些堆疊包含 Amazon 資GameLift源所依賴的資源。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。它也可能包含適用於您遊戲的其他AWS資源，例如 DynamoDB 表、亞馬遜紅移叢集和 Lambda 函數。這些資源中有許多是區域特定的，因此您必須在每個區域中建立這些資源。Amazon GameLift 也需要能夠存取這些支援資源的 IAM 角色。由於 IAM 角色不受區域限制，因此您只需要一個角色資源，放置在任何區域中，並在所有其他支援堆疊中參照。
- 區域 Amazon GameLift 堆疊 — 此堆疊包含必須存在於部署遊戲的每個區域中的 Amazon GameLift 資源，包括建置或指令碼、一組叢集和別名。 AWS CloudFormation使用 S3 儲存貯體位置的檔案建立組建或指令碼資源，並將組建或指令碼部署到一或多個叢集資源。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。
- 全球 Amazon GameLift 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。

下圖說明在多個 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。


Global GameLift Stack
Regional GameLift Stacks
Regional Support Stacks
Matchmaking Configuration
Matchmaking Rule Set
Game Session Queue
Destination
Alias
Fleet
Build or Script
IAM Role
allows GameLift access to S3
S3 Bucket
stores game server build/script files
Region 1
Region 2
Region 3
CloudFormation stack
GameLift resources



更新組建

Amazon GameLift 組建是不可變的，組建和叢集之間的關係也是如此。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使用指南》中的〈更新堆疊資源的行為〉。

自動部署組建更新

更新包含相關建置、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，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、驗證檔案，並建立新的 Amazon GameLift 組建。
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。

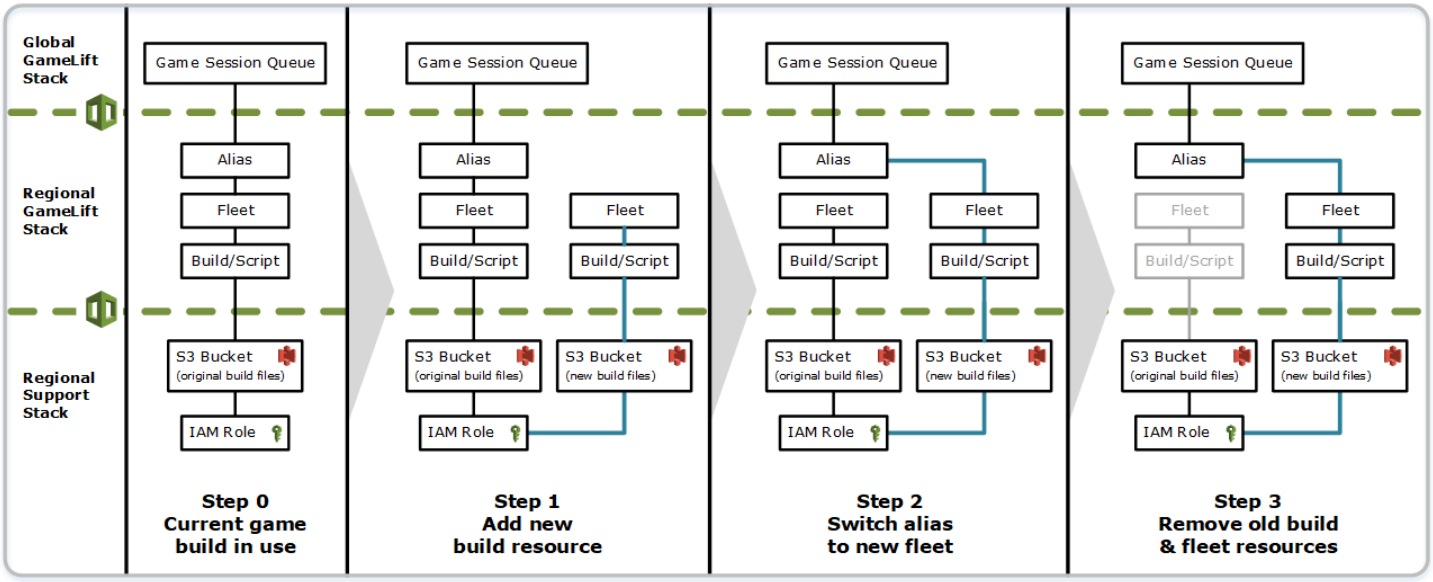
您也可以使 AWS CloudFormation 保留資源，而不是刪除它們。如需詳細資訊，請參閱《AWS CloudFormation API 參考》 中的 RetainResources。

手動部署組建更新

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 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，AWS CloudFormation 會建立第二個建置資源和對應機群。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。



復原的運作方式

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。建立新的 Amazon GameLift 組建和叢集，而且在叢集啟用後，別名會切換到新的叢集。如果您是個別管理別名，則需要切換它們以指向新機群。

如需有關如何處理失敗或卡住的復原的詳細資訊，請參閱AWS CloudFormation使用者指南中的繼續復原更新。

適用於亞馬遜的 VPC 對等互連 GameLift

本主題提供有關如何在 Amazon GameLift 託管的遊戲伺服器與其他非 Amazon 資源之間設定 VPC 對等連接的指引。GameLift使用 Amazon 虛擬私有雲 (VPC) 對等連線，讓遊戲伺服器能夠直接和私密地與其他AWS資源 (例如 Web 服務或存放庫) 進行通訊。您可以使用任何在上執行的資源建立 VPC 對等互連，AWS並由您有權存取的AWS帳戶管理。

> **Note**
>
> VPC 對等互連是進階功能。若要瞭解啟用遊戲伺服器與其他AWS資源直接和私密通訊的偏好選項，請參閱與您車隊的其他AWS資源進行溝通。

如果您已經熟悉 Amazon VPC 和 VPC 對等互連，請瞭解設定與 Amazon GameLift 遊戲伺服器的對等互連會有所不同。您無法存取包含遊戲伺服器的 VPC (由 Amazon GameLift 服務控制)，因此您無法直接請求 VPC 對等互連。相反地，您必須先使用非 Amazon GameLift 資源預先授權 VPC，以接受來自 Amazon 服務的對等請求。GameLift然後，您會觸發 Amazon GameLift 要求您剛才授權的 VPC 對等互連。Amazon 會GameLift處理建立對等連線、設定路由表和設定連線的任務。

為現有機群設定 VPC 對等互連

1. 獲取AWS帳戶 ID 和憑據。

   您需要下列AWS帳戶的 ID 和登入認證。您可以登入AWS Management Console並檢視您的AWS帳戶設定，以尋找帳戶 ID。若要取得登入資料，請前往 IAM 主控台。

   - AWS您用來管理 Amazon GameLift 遊戲伺服器的帳戶。
   - AWS您用來管理非 Amazon GameLift 資源的帳戶。

   如果您對 Amazon GameLift 和非 Amazon GameLift 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：

   - 適用於您的亞馬遜GameLift遊戲伺服器的 VPC — 這是您的亞馬遜GameLift車隊 ID。您的遊戲伺服器是在 EC2 執行個體叢集GameLift上部署在 Amazon 中。叢集會自動放置在自己的 VPC 中，該 VPC 由 Amazon GameLift 服務管理。您無法直接存取 VPC，因此可由叢集 ID 識別。
   - 適用於非 Amazon GameLift AWS 資源的 VPC — 您可以使用任何在其上執行AWS並由您可存取的AWS帳戶管理的資源建立 VPC 對等互連。如果您尚未為這些資源建立 VPC，請參閱開始使用 Amazon V PC。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。Amazon GameLift 叢集遊戲伺服器的虛擬私人雲端與叢集位於同一個區域。

3. 授權 VPC 對等。

   在此步驟中，您將預先授權來自 Amazon 的未來請求，GameLift以將 VPC 與您的遊戲伺服器與非 Amazon 資源的 VPC 對等。GameLift此動作會更新 VPC 的安全群組。

   若要授權 VPC 對等互連，請呼叫亞馬遜GameLift服務 API CreateVpcPeeringAuthorization() 或使用 AWS CLI 命令。`create-vpc-peering-authorization`使用管理非 Amazon GameLift 資源的帳戶進行此呼叫。請識別以下資訊：

   - 對等虛擬私人雲端 ID — 這適用於包含您非 GameLift Amazon 資源的虛擬私人雲端。
   - 亞馬遜GameLiftAWS帳戶 ID — 這是您用來管理亞馬遜GameLift車隊的帳戶。

   授權 VPC 對等互連後，除非已撤銷，否則授權在 24 小時內有效。您可以使用以下操作來管理 VPC 對等授權：

   - DescribeVpcPeeringAuthorizations() (AWSCLI`describe-vpc-peering-authorizations`)。
   - DeleteVpcPeeringAuthorization() (AWSCLI`delete-vpc-peering-authorization`)。

4. 要求對等連線。

   透過有效的授權，您可以要求 Amazon GameLift 建立對等連線。

   若要請求 VPC 對等互連，請呼叫亞馬遜GameLift服務 API CreateVpcPeeringConnection() 或使用 AWS CLI 命令。`create-vpc-peering-connection`使用管理 Amazon GameLift 遊戲伺服器的帳戶撥打此電話。使用下列資訊來識別您要建立對等互連的兩個 VPC：

   - 對等虛擬私人雲端 ID 和AWS帳戶 ID — 這是您非 Amazon GameLift 資源的虛擬私人雲端，以及您用來管理這些資源的帳戶。VPC ID 必須與有效對等授權上的 ID 相符。
   - 叢集 ID — 這可識別您 Amazon GameLift 遊戲伺服器的虛擬私人雲端。

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 對等連線相關聯的叢集事件，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() (AWSCLI`describe-vpc-peering-connections`)。
- DeleteVpcPeeringConnection() (AWSCLI`delete-vpc-peering-connection`)。

使用新機群設定 VPC 對等

您可以建立新的 Amazon GameLift 叢集，並同時要求 VPC 對等連線。

1. 獲取AWS帳戶 ID 和憑據。

   您需要下列兩個AWS帳戶的 ID 和登入認證。您可以登入AWS Management Console並檢視您的AWS帳戶設定，以尋找帳戶 ID。若要取得登入資料，請前往 IAM 主控台。

   - AWS您用來管理 Amazon GameLift 遊戲伺服器的帳戶。
   - AWS您用來管理非 Amazon GameLift 資源的帳戶。

   如果您對 Amazon GameLift 和非 Amazon GameLift 資源使用相同的帳戶，則只需要該帳戶的 ID 和登入資料。
2. 取得您非亞馬遜GameLiftAWS資源的虛擬私人雲端識別碼。

   如果您尚未為這些資源建立 VPC，請立即執行 (請參閱開始使用 Amazon VPC)。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 資源在與 Amazon 使用的AWS帳戶或使用者/使用者群組不同的帳戶或使用者/使用者群組下進行管理GameLift，則在下一步要求授權時，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. 使用非 GameLift Amazon 資源授權 VPC 對等互連。

   當 Amazon GameLift 建立新叢集和對應的 VPC 時，它也會針對您的非 GameLift Amazon 資源向虛擬私人雲端傳送請求給對等。您需要預先授權該請求。此步驟會更新您 VPC 的安全群組。

   使用管理非亞馬遜GameLift資源的帳戶登入資料，呼叫亞馬遜GameLift服務 API CreateVpcPeeringAuthorization() 或使用 AWS CLI 命令`create-vpc-peering-authorization`。請識別以下資訊：

   - 對等虛擬私人雲端 ID — 虛擬私人雲端的識別碼與您的非 GameLift Amazon 資源一起使用。
   - 亞馬遜GameLiftAWS帳戶 ID — 您用來管理 Amazon GameLift 叢集的帳戶識別碼。

授權 VPC 對等互連後，除非已撤銷，否則授權在 24 小時內有效。您可以使用以下操作來管理 VPC 對等授權：

- DescribeVpcPeeringAuthorizations() (AWSCLIdescribe-vpc-peering-authorizations)。
- DeleteVpcPeeringAuthorization() (AWSCLIdelete-vpc-peering-authorization)。

4. 遵循使用 AWSCLI 建立新叢集的指示。包含下列其他參數：

- peer-vpc-aws-account-id — 您用來管理虛擬私人雲端 (包含非 GameLift Amazon 資源) 的帳戶識別碼。
- peer-vpc-id— VPC 的 ID 與您的非GameLift帳戶.

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describe-vpc-peering-connections叫來追蹤對等連線要求的成功或失敗。

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 會透過您的非 Amazon GameLift AWS 帳戶 ID 和虛擬私人雲端 ID 的組合進行唯一識別。

```
$ 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 遊戲伺服器建立 VPC 對等連線，請考慮以下常見的根本原因：

- 找不到所請求連線的授權：
  - 檢查非 Amazon GameLift VPC 的虛擬私人雲端授權狀態。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 對等連線組態) 會重疊。指派給對等 VPC 的 IPv4 CIDR 區塊不能重疊。您 Amazon GameLift 叢集的 VPC CIDR 區塊會自動指派且無法變更，因此您必須針對非 Amazon 資源變更 VPC 的 CIDR 區塊。GameLift若要解決此問題：

  - 通過電話`DescribeVpcPeeringConnections()`查找您的亞馬遜GameLift車隊的這個 CIDR 塊。
  - 前往 Amazon VPC 主控台，尋找非 Amazon GameLift 資源適用的 VPC，然後變更 CIDR 區塊，使其不會重疊。
- 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。

在主機中檢視遊戲資料

受管 Amazon GameLift 服務會持續收集使用中遊戲的資料，以協助您瞭解玩家行為和效能。使用 Amazon GameLift 主控台，您可以檢視、管理和分析組建、叢集、遊戲工作階段和玩家工作階段的這些資訊。

主題

查看您當前的 Amazon GameLift 狀態

Amazon GameLift 儀表板提供了以下內容的視圖：

- [就緒]、[初始化] 和 [失敗] 狀態的組建數目。如需目前區域中組建的詳細資訊，請選擇 [檢視組建]。
- 所有狀態的叢集數目。選擇 [檢視車隊] 以瞭解目前區域中車隊的詳細資訊。
- 您目前的資源。
- 新功能和服務公告。

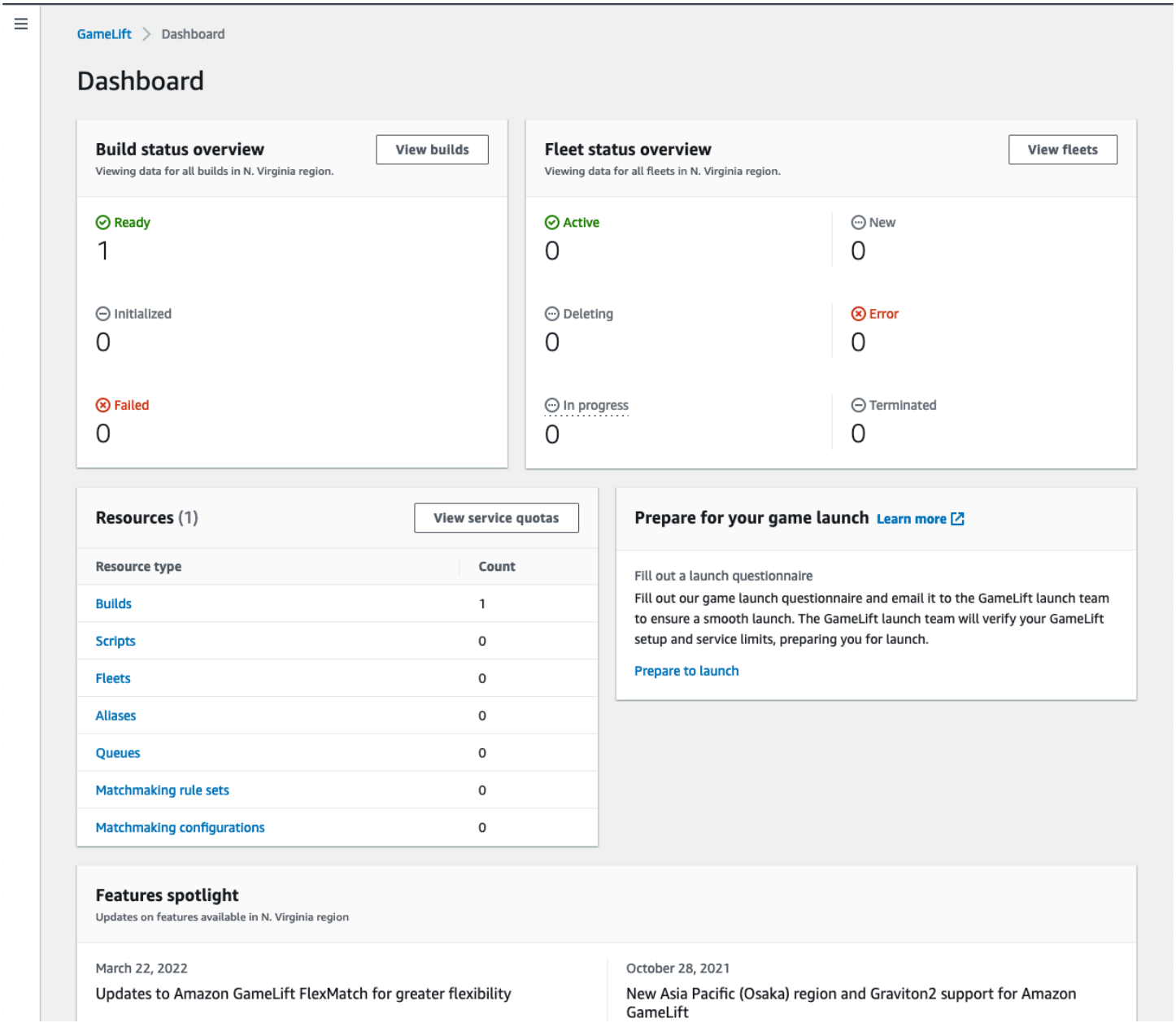
打開 Amazon GameLift 儀表板

- 在 Amazon 主 GameLift 控台的導覽窗格中，選擇「儀表板」。

在儀表板中，您可以：

- 選擇 [準備上市] 並填寫對應的啟動問卷，讓您的遊戲準備上市。
- 選擇檢視服務配額，以準備啟動或回應啟動時，請求服務配額增加。

- 通過選擇功能焦點中的鏈接，查看博客文章和有關新功能的詳細信息。



檢視您的組建

在 Amazon GameLift 主控台的 [組建] 頁面上，您可以檢視和管理已上傳到 Amazon 的所有遊戲伺服器組建的相關資訊GameLift。在功能窗格中，選擇 [主機]、[組建]。

[組建] 頁面會顯示每個組建的下列資訊：

> **Note**
>
> [組建] 頁面只會顯示您目前AWS區域中的組建。

- 名稱 — 與已上載組建相關聯的名稱。
- 狀態 — 組建的狀況。顯示三種狀態訊息之一：
  - 已初始化 — 上傳尚未開始或仍在進行中。
  - 準備就緒-組建已準備就緒，可以建立車隊。
  - 失敗-在 Amazon GameLift 收到二進製文件之前構建超時。
- 建立時間 — 您將組建上傳到 Amazon 的日期和時間GameLift。
- 組建 ID — 上傳時指派給組建的唯一 ID。
- 版本-與已上載版次相關聯的版本標籤。
- 作業系統 — 執行組建的作業系統。建置作業系統會決定 Amazon 在叢集執行個體上GameLift安裝的作業系統。
- 大小 — 上傳到亞馬遜的構建文件的大小（以兆字節（MB）為單位。GameLift
- 艦隊 — 與組建一起部署的艦隊數量。

您可以在此頁面進行下列任一操作：

- 查看建置的詳細資訊。選擇組建名稱以開啟其組建詳細資料頁面。
- 從建置建立新機群。選取組建，然後選擇 [建立叢集]。
- 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]。

構建細節

在 [組建] 頁面上，選擇組建名稱以開啟其詳細資料頁面。詳細資料頁面的 [概觀] 區段會顯示與 [組建] 頁面相同的組建摘要資訊。「艦隊」部分顯示了使用構建創建的艦隊列表，其中包括與「艦隊」頁面相同的摘要信息。

檢視您的指令碼

在 Amazon GameLift 主控台的指令碼頁面上，您可以檢視和管理已上傳到 Amazon 的所有即時伺服器指令碼的相關資訊GameLift。在功能窗格中，選擇 [主機]、[指令碼]。

「命令檔」頁面會顯示每個命令檔的下列資訊：

> **Note**
>
> 「指令碼」頁面只會顯示您目前AWS區域中的指令碼。

- 名稱 — 與上載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 — 與上載的指令碼相關聯的版本標籤。
- 大小 — 上傳到 Amazon 的指令碼檔案大小 (以 MB 為單位)。GameLift
- 建立時間 — 您將指令碼上傳到 Amazon 的日期和時間GameLift。
- 「艦隊」— 使用指令碼部署的艦隊數量。

您可以在此頁面進行下列任一操作：

- 檢視指令集詳細資訊 選擇組建名稱以開啟其指令碼詳細資料頁面。
- 從指令碼建立新叢集。選取指令碼，然後選擇 [建立叢集]。
- 篩選並排序指令碼清單。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

在「指令碼」頁面上，選擇指令碼的名稱以開啟其詳細資料頁面。詳細資訊頁面的 [總覽] 區段會顯示與 [組建] 頁面相同的指令碼摘要資訊。「艦隊」區段會顯示使用指令碼建立的艦隊清單，其中包括與「艦隊」頁面相同的摘要資訊。

檢視您的車隊

您可以在您的AWS帳戶GameLift下查看有關為在 Amazon 上託管遊戲而創建的所有艦隊的信息。該列表顯示您當前區域創建的艦隊。在「艦隊」頁面中，您可以建立新的叢集或檢視叢集的其他詳細資訊。

艦隊的詳細資訊頁面包含使用情況資訊、指標、遊戲工作階段資料和玩家工作階段資料。您也可以編輯叢集記錄或刪除叢集。

若要檢視 [叢集] 頁面，請從導覽窗格中選擇 [叢集]。

Fleets (機群) 頁面預設會顯示以下摘要資訊。您可以通過選擇設置（齒輪）按鈕來自定義顯示的信息。

- 名稱 — 指定給叢集的易記名稱。
- 狀態 — 叢集的狀態，可以是下列其中一種狀態：[新增]、[下載中]、[建立中] 和 [作用中]。
- 建立時間 — 建立叢集的日期和時間。
- 運算類型 — 用來託管遊戲的運算類型。叢集可以是受管 EC2 叢集或任何地方叢集。
- 執行個體類型 — Amazon EC2 執行個體類型，可決定叢集執行個體的運算容量。
- 作用中執行個體 — 叢集正在使用的 EC2 執行個體數目。
- 所需執行個體 — 要保持作用中的 EC2 執行個體數量。
- 遊戲工作階段 — 在艦隊中執行的作用中遊戲工作階段數目。資料延遲五分鐘。

檢視車隊詳情

透過選擇叢集名稱，從儀表板或「艦隊」頁面存取「機隊」詳細資訊頁面。

在叢集詳細資訊頁面上，您可以執行下列動作：

- 更新叢集的屬性、連接埠設定和執行階段設定。
- 新增或移除叢集位置。
- 變更機群容量設定。
- 設定或變更目標追蹤自動縮放比例。
- 刪除機群。

詳細資訊

車隊設定

- 「叢集 ID」— 指派給叢集的唯一識別碼。
- 名稱 — 叢集的名稱。
- ARN — 指派給此叢集的識別碼。叢集的 ARN 會將其識別為 Amazon GameLift 資源，並指定區域和 AWS帳戶。

- 描述 — 叢集的簡短可識別描述。
- 狀態 — 叢集的目前狀態，可能是 [新增]、[下載中]、[建立中] 和 [使用中]。
- 建立時間 — 建立叢集的日期和時間。
- 終止時間 — 叢集終止的日期和時間。如果叢集仍處於作用中狀態，則此欄位為空白。
- 叢集類型 — 指出叢集使用隨需執行個體還是 Spot 執行個體。
- EC2 類型 — 為叢集建立時選取的 Amazon EC2 執行個體類型。
- 執行個體角色 — 管理其他AWS資源存取權的 AWS IAM 角色 (如果叢集建立期間已提供)。
- TLS 憑證 — 是否啟用或停用叢集以使用 TLS 憑證來驗證遊戲伺服器並加密所有用戶端/伺服器通訊。
- 量度群組 — 用來彙總多個叢集量度的群組。
- 遊戲縮放保護政策 — 艦隊遊戲工作階段保護的目前設定。
- 每位玩家的遊戲工作階段上限 — 玩家在政策期間可建立的工作階段數上限。
- 政策期間 — 等待多長時間才能重設玩家已建立的工作階段數量。

構建細節

[組建詳細資料] 區段會顯示裝載在叢集上的組建。選取組建名稱以查看完整的組建詳細資料頁面。

運行時配置

程式實際執行組態段落會顯示要在每個執行處理上啟動的伺服器處 它包含遊戲伺服器可執行檔的路徑和選用的啟動參數。

遊戲會話激活

「遊戲工作階段啟動」區段會顯示同時啟動的伺服器處理序數目，以及在終止程序之前等待程序啟動的時間長度。

EC2 連接埠設定

[連接埠] 區段會顯示叢集的連線權限，包括 IP 位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 Amazon 中使用指標的詳細資訊 GameLift，請參閱監控亞馬遜GameLift與亞馬遜 CloudWatch。

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。請參閱亞馬遜 GameLift API 參考中的事件說明。

擴展

S caling 索引標籤包含叢集容量的相關資訊，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。

擴充容量

檢視每個叢集位置目前的叢集容量設定。如需變更限制和容量的詳細資訊，請參閱擴展亞馬遜 GameLift託管容量。

- AWS位置 — 部署叢集執行個體的位置名稱。
- 狀態 — 叢集位置的託管狀態。位置狀態必ACTIVE須是能夠主持遊戲。
- 最小大小 — 必須在該位置部署的執行個體數目下限。
- 所需執行個體 — 用於維護位置的作用中執行個體的目標數目。當作用中執行個體和所需執行個體具有相同時，資源調整事件會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的執行個體為止。
- 大小上限 — 可在該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目。此值告訴您可以新增至位置的執行個體數目上限。

自動擴展政策

本節涵蓋套用至叢集之自動調整規模原則的相關資訊。您可以設定或更新以目標為基礎的政策。叢集的規則型原則 (必須使用 AWS SDK 或 CLI 定義) 會顯示在此處。如需縮放的詳細資訊，請參閱使用亞馬遜自動擴展車隊容量 GameLift。

縮放歷史

檢視容量隨時間變更的圖表。

Locations

[位置] 索引標籤會列出所有部署叢集執行個體的位置。位置包括車隊的所在地區域和任何已新增的遠端位置。您可以直接在此標籤中新增或移除位置。

- 位置 — 部署叢集執行個體的位置名稱。
- 狀態 — 叢集位置的託管狀態。位置狀態會追蹤啟動該位置中第一個例項的程序。位置狀態必ACTIVE須是能夠主持遊戲。
- 作用中執行個體 — 在叢集位置執行伺服器處理作業的執行個體數目。
- 作用中伺服器 — 可在叢集位置託管遊戲工作階段的遊戲伺服器處理序數目。
- 遊戲工作階段 — 在叢集位置中執行個體上啟用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，參與在艦隊位置活躍的遊戲工作階段。

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，請參閱查看有關遊戲和玩家會話的數據。

查看有關遊戲和玩家會話的數據

您可以查看有關遊戲會話和單個玩家的信息。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。

檢視遊戲工作階段和玩家資料

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇「叢集」。
2. 從代管遊戲工作階段的「艦隊」清單中選擇艦隊。
3. 選擇「遊戲工作階段」標籤。此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。
4. 選擇一個遊戲工作階段以檢視關於遊戲工作階段的其他資訊，以及連線到遊戲的玩家清單。

詳細資訊

概要

本節會顯示您的遊戲工作階段資訊摘要。

- 狀態 — 遊戲工作階段狀態。
  - 啟動 — 執行個體正在啟動遊戲工作階段。
  - 有效 — 根據工作階段的玩家建立政策，遊戲工作階段正在執行中，並且可以接收玩家。

  - 終止 — 遊戲工作階段已結束。
- ARN — 遊戲工作階段的亞馬遜資源名稱。
- 名稱 — 為遊戲工作階段產生的名稱。
- 位置 — Amazon GameLift 託管遊戲工作階段的位置。
- 建立時間 — Amazon GameLift 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 位址 — 為遊戲工作階段指定的 IP 位址。
- 連接埠 — 用來連線到遊戲工作階段的連接埠號碼。
- 創作者 ID — 啟動遊戲工作階段之玩家的唯一識別碼。
- 玩家工作階段建立政策 — 指出遊戲工作階段是否接受新玩家。
- 遊戲擴展保護政策 — 在 Amazon 在叢集中GameLift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

格式良好的資料，可在開始時傳送至您的遊戲工作階段。

遊戲屬性

會影響您遊戲工作階段的金鑰和值配對屬性。

配對資料

FlexMatch分房系統的 JSON 資訊。若要檢視和編輯分房系統，請選擇 [檢視配對設定]。如需有關FlexMatch配對的詳細資訊，請參閱建立分房系統。

玩家工作階段

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碼。
- 玩家 ID — 玩家的唯一識別碼。選擇此 ID 以獲取其他玩家信息。
- 狀態 — 玩家工作階段的狀態。以下是可能的狀態：
  - 保留 — 玩家會話已保留，但玩家沒有連接。

- 作用中 — 玩家工作階段已連線至遊戲伺服器。
- 已完成 — 玩家工作階段已結束；玩家不再連線。
- 逾時 — 玩家無法連線。

- 創建時間-玩家連接到遊戲會話的時間。
- 結束時間 — 玩家從遊戲工作階段中斷連線的時間。
- 玩家資料 — 玩家工作階段建立期間提供的玩家相關資訊。

玩家資訊

檢視選取玩家的詳細資訊，包括玩家連接到目前區域中所有機群的所有遊戲清單。此資訊包括每個玩家工作階段的狀態、開始時間、結束時間和總連線時間。您可以選擇檢視相關遊戲工作階段和車隊的資料。

檢視您的別名

「別名」頁面會顯示您在目前「區域」中建立之叢集別名的相關資訊。若要檢視別名頁面，請在導覽窗格中選擇「別名」。

您可以在別名頁面上執行下列動作：

- 創建一個新的別名。選擇 [建立別名]。
- 篩選並排序別名表格。使用表格上方的控制項。
- 檢視別名詳細資訊。選擇別名以開啟別名詳細資訊頁面。
- 刪除別名。選擇別名，然後選擇 [刪除]。

別名詳情

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 。
- 檢視與別名相關聯的叢集。
- 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編號。
- 「描述」— 別名的描述。
- ARN — 別名的亞馬遜資源名稱。
- 建立 — 建立別名的日期和時間。
- 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，可以是下列其中一種：
  - 簡單 — 將玩家流量路由至指定的叢集 ID。您可以隨時更新別名的機群 ID。
  - 終端機 — 將訊息傳回給用戶端。例如，您可以將正在使用out-of-date客戶端的玩家引導到可以獲得升級的位置。
- 標籤 — 用來識別別名的索引鍵和值配對。

檢視您的佇列

您可以針對所有現有的遊戲工作階段配置佇列，檢視相關的資訊。佇列頁面會顯示在您目前區域中建立的佇列。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測量結果資料。如需佇列的詳細資訊，請參閱為遊戲會話放置設置亞馬遜GameLift隊列。

「佇列」頁面會顯示每個佇列的下列摘要資訊：

- 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 — 遊戲工作階段放置要求在逾時前保留在佇列中的時間上限 (以秒為單位)。
- 佇列中的目的地 — 佇列組態中列出的叢集數目。Amazon 會在佇列中的任何車隊中GameLift放置新的遊戲工作階段。

檢視佇列詳情

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，請移至「佇列」頁面並選擇佇列名稱。

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。刪除佇列之後，所有參照該佇列名稱的新遊戲工作階段要求都會失敗。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佇列。

詳細資訊

概要

概觀區段會顯示佇列的 Amazon 資源名稱 (ARN) 和逾時。您可以在 Amazon GameLift 的其他動作或區域中引用隊列時使用 ARN。逾時是遊戲工作階段放置要求在逾時前保留在佇列中的時間上限 (以秒為單位)。

事件通知

事件通知區段列出 Amazon GameLift 發佈事件通知的 SNS 主題，以及新增至此佇列建立之所有事件的事件資料。

Tags (標籤)

「標籤」(Tag) 表格會顯示用來標記資源的索引鍵和值。有關標記的詳細資訊，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。

佇列量度包含一系列說明佇列中放置活動的資訊，包括按「區域」組織的成功放置位置。您可以使用區域資料來瞭解您在哪裡託管遊戲。地區放置指標可協助偵測整體佇列設計的問題。

亞馬遜也提供佇列指標CloudWatch。如需可用量度的說明，請參閱亞馬遜隊列GameLift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。

Amazon 在目的地GameLift搜尋可用資源以託管新遊戲工作階段時，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，亞馬遜就會在那裡GameLift放置新的遊戲會話。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。這項資料會告訴 Amazon GameLift 搜尋玩家平均延遲最低的可用目的地。如需設計佇列的詳細資訊，請參閱設計遊戲工作階段佇列。

工作階段位

播放器延遲政策

[播放程式延遲原則] 區段會顯示佇列使用的所有原則。這些表格會依其強制執行的順序列出策略。

Locations

[位置] 區段會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] 區段會顯示佇列評估遊戲工作階段詳細資料的順序。

位置順序

「位置順序」區段會顯示佇列在進行遊戲工作階段時所使用的預設順序。如果您尚未定義其他類型的優先順序，佇列會使用此順序。

監控 Amazon GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

監控是維護 Amazon GameLift 和其他 AWS 解決方案的可靠性、可用性和效能的重要組成部分。Amazon 的指標主要有三種用途 GameLift：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情況，以及使用手動或自動調整規模來管理容量。

AWS 提供以下監控工具來觀看 Amazon GameLift、在發生錯誤時報告，並在適當時採取自動動作：

- Amazon GameLift 遊戲
- Amazon CloudWatch—— 您可以即時監控 Amazon GameLift 指標，以及您在 AWS 服務上執行的其他 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，以及設定警示，以便在指標達到指定臨界值時通知或採取動作的功能。
- AWS CloudTrail— 擷取由或代表您 AWS 帳戶在 Amazon 和其他 AWS 服務中發出的所有 API 呼叫 GameLift 和相關事件。資料會以日誌檔的形式傳送到您指定的 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 位址，以及呼叫發生的時間。
- 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。

主題

- 監控亞馬遜GameLift與亞馬遜 CloudWatch
- 記錄亞馬遜 GameLift API 呼叫 AWS CloudTrail
- 在亞馬遜記錄服務器消息 GameLift

監控亞馬遜GameLift與亞馬遜 CloudWatch

您可以GameLift使用 Amazon 監控 AmazonCloudWatch，這是一種AWS收集原始資料並將其處理為可讀且近乎即時的指標的服務。這些統計資料會保留 15 個月，以提供您透過 Amazon GameLift 託管遊戲伺服器執行的歷史觀點。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。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

下表列出亞馬遜的指標和維度GameLift。中提供的所有指標也可CloudWatch在 Amazon GameLift 主控台取得，該主控台會以一組可自訂的圖形形式提供資料。若要存取遊戲的 CloudWatch 指標，可使用 AWS Management Console、AWS CLI 或 CloudWatch API。

如果量度沒有位置，則會使用本位置。

亞馬遜指標的維GameLift度

Amazon GameLift 支援依下列維度篩選指標。

維度	描述
Location	篩選叢集部署位置的指標。如果量度沒有位置，則會使用本位置。
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。透過將測量結果群組名稱新增至叢集的屬性，將叢集新增至測量結果群組 UpdateFleetAttributes(請參閱 ())。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。
GameServerGroup	篩選遊戲伺服器群組的 FleetIQ 指標。

適用於艦隊的亞馬遜GameLift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：叢集可與受管 Amazon GameLift 解決方案搭配使用。亞馬遜GameLift服務將指標發送到CloudWatch每分鐘。

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統計數據：平均、最低、最高 尺寸：位置
DesiredInstances	Amazon 正在叢集中維護的作GameLift用中執行個體目標數量。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統計數據：平均、最低、最高 尺寸：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統計數據：平均、最低、最高

指標	描述
	尺寸：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統計數據：平均、最低、最高 尺寸：位置
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統計數據：平均、最低、最高 尺寸：位置
PercentIdleInstances	所有閒置的作用中執行個體的百分比 (計算方式為 IdleInstances / ActiveInstances)。此指標可用於自動調整規模。 單位：百分比 相關CloudWatch統計數據：平均、最低、最高 尺寸：位置

指標	描述
RecycledInstances	已回收和更換的競價型執行個體數目。Amazon GameLift 回收目前未託管遊戲工作階段且中斷可能性很高的競價型執行個體。 單位：計數 相關CloudWatch統計數字：總和、平均、最小值、最高 尺寸：位置
InstanceInterruptions	已中斷的 Spot 執行個體的數量。 單位：計數 相關CloudWatch統計數字：總和、平均、最小值、最高 尺寸：位置
CPUUtilization	EC2 指標。對於 Amazon，GameLift此指標代表叢集位置中所有作用中執行個體的硬體效能。Amazon EC2 用於執行執行個體的實體 CPU 時間百分比，包括執行使用者程式碼和 Amazon EC2 程式碼所花費的時間。CloudWatch由於傳統裝置模擬、非舊式裝置的組態、大量中斷的工作負載、即時移轉和即時更新等因素，作業系統中的工具顯示的百分比可能會有所不同。 單位：百分比
NetworkIn	EC2 指標。對於 Amazon，GameLift此指標代表叢集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

指標	描述
NetworkOut	EC2 指標。對於 Amazon，GameLift此指標代表叢集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
DiskReadBytes	EC2 指標。對於 Amazon，GameLift此指標代表叢集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。您可以使用它來確定應用程序的速度。 單位：位元組
DiskWriteBytes	EC2 指標。對於 Amazon，GameLift此指標代表叢集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。您可以使用它來確定應用程序的速度。 單位：位元組
DiskReadOps	EC2 指標。對於 Amazon，GameLift此指標代表叢集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

指標	描述
DiskWriteOps	EC2 指標。對於 Amazon，GameLift此指標代表叢集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統計數據：平均、最低、最高 尺寸：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統計數據：平均、最低、最高 尺寸：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 HealthyServerProcesses / ActiveServerProcesses)。

指標	描述
	單位：百分比 相關CloudWatch統計數據：平均、最低、最高 尺寸：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。此指標包括 Amazon GameLift 服務啟動的終止。當伺服器處理序停止回應、一致地報告運作狀態檢查失敗，或未乾淨地終止 (藉由呼叫 ProcessEnding()) 時，就會發生這種情況。 單位：計數 相關CloudWatch統計數字：總和、平均、最小值、最高 尺寸：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統計數字：總和、平均、最小值、最高 尺寸：位置

指標	描述
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統計數字：總和、平均、最小值、最高 尺寸：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統計數據：平均、最低、最高 尺寸：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統計數據：平均、最低、最高 尺寸：位置

指標	描述
AvailableGameSessions	目前尚未用於主控遊戲工作階段的作用中狀態良好的伺服器處理序，而且可以立即啟動新的遊戲工作階段，以啟動新的伺服器處理序或執行個體。此指標可搭配使用自動調整規模。 **Note** 對於限制同時遊戲工作階段啟動的艦隊，請使用指標。ConcurrentActivatableGameSessions 該指標更準確地代表可以在沒有任何延遲的情況下啟動的新遊戲工作階段數量。 單位：計數 相關CloudWatch統計數據：平均、最低、最高 尺寸：位置
ConcurrentActivatableGameSessions	目前尚未用於主控遊戲工作階段，且可以立即啟動新的遊戲工作階段的作用中、健康狀態的伺服器處理序。 此量度與下列方式有所不同：由於遊戲工作階段啟動的限制，目前無法啟動新遊戲工作階段的伺服器處理序不會計算。AvailableGameSessions （請參閱車隊RuntimeConfiguration可選設置MaxConcurrentGameSessionActivations ）。對於不限制遊戲工作階段啟動的艦隊，此度量與. AvailableGameSessions 單位：計數 相關CloudWatch統計數據：平均、最低、最高 尺寸：位置

指標	描述
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統計數字：平均 尺寸：位置
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統計數字：總和、平均、最小值、最高 尺寸：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統計數據：平均、最低、最高

指標	描述
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統計數字：總和、平均、最小值、最高

亞馬遜隊列GameLift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：佇列可與受管 Amazon GameLift 解決方案搭配使用。亞馬遜GameLift服務將指標發送到CloudWatch每分鐘。

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統計數字：平均值、最低值、最高值、總和 尺寸：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)。此指標是以成本為基礎，而非延遲。第一選擇艦隊可能是佇列中列出的第一個艦隊，或者當放置請求包含玩家延遲資料時，這是 FleetIQ 優先順序選擇的第一個艦隊。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

指標	描述
	相關CloudWatch統計數字：平均值、最低值、最高值、總和
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。第一選擇艦隊可能是佇列中列出的第一個艦隊，或者當放置請求包含玩家延遲資料時，這是您定義的 FleetIQ 優先順序選擇的第一個艦隊。 單位：計數 相關CloudWatch統計數字：平均值、最低值、最高值、總和
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統計數字：平均值、最低值、最高值、總和
LowestPricePlacement	已成功放置在艦隊中的遊戲會話，該隊列價格在所選區域中的可能價格最低。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統計數字：平均值、最低值、最高值、總和

指標	描述
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統計數字：總和
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統計數字：平均值、最低值、最高值、總和
PlacementsFailed	從上次報告之後，因任何原因失敗的遊戲工作階段放置請求。 單位：計數 相關CloudWatch統計數字：平均值、最低值、最高值、總和
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統計數字：平均值、最低值、最高值、總和
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統計數字：平均值、最低值、最高值、總和

指標	描述
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統計數字：平均值、最低值、最高值、總和
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統計數字：平均值、最低值、最高值、總和 尺寸：位置

用於配對的亞馬遜GameLift指標

命Amazon GameLift名空間包括配對設定和配對規則的FlexMatch活動量度。FlexMatch配對功能可與亞馬遜託管GameLift解決方案一起使用。亞馬遜GameLift服務將指標發送到CloudWatch每分鐘。

如需配對活動順序的詳細資訊，請參閱 Amazon 的GameLiftFlexMatch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統計數字：平均值、最低值、最高值、總和
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。

指標	描述
	單位：計數 相關CloudWatch統計數字：總和
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。 單位：計數 相關CloudWatch統計數字：總和
MatchesCreated	從上次報告之後，已建立的潛在匹配。 單位：計數 相關CloudWatch統計數字：總和
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統計數字：總和
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統計數字：總和
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統計數字：總和

指標	描述
TicketsFailed	從上次報告之後，導致失敗的配對請求。 單位：計數 相關CloudWatch統計數字：總和
TicketsStarted	從上次報告之後，已建立的新配對請求。 單位：計數 相關CloudWatch統計數字：總和
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統計數字：總和
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統計數據：數據樣本，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統計數據：數據樣本，平均值，最小值，最大值

指標	描述
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統計數據：數據樣本，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 相關CloudWatch統計數字：總和
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統計數字：總和

針對 Fle GameLift etIQ 的亞馬遜指標

Amazon GameLift命名空間包括 FleetIQ 遊戲伺服器群組和遊戲伺服器活動的指標，這是 FleetIQ 獨立遊戲主機解決方案的一部分。亞馬遜GameLift服務將指標發送到CloudWatch每分鐘。另請參閱 Amazon EC2 自動擴展使用者指南CloudWatch中的使用 Amazon 監控您的自動擴展群組和執行個體。

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CloudWatch統計數字：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CloudWatch統計數字：總和 維度：GameServerGroup
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CloudWatch統計數字：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CloudWatch統計數字：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

指標	描述
	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CloudWatch統計數據：平均、最低、最高 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CloudWatch統計數字：總和 尺寸:GameServerGroup, InstanceType
InstanceInterruptions	Spot 執行個體由於可用性有限而中斷。 單位：計數 相關CloudWatch統計數字：總和 尺寸:GameServerGroup, InstanceType

記錄亞馬遜 GameLift API 呼叫 AWS CloudTrail

亞馬遜GameLift集成了一種服務AWS CloudTrail，該服務可提供亞馬遜中的用戶，角色或AWS服務採取的操作記錄GameLift。CloudTrailGameLift以事件形式擷取亞馬遜的所有 API 呼叫。擷取的呼叫包括來自 Amazon GameLift 主控台的呼叫，以及對 Amazon GameLift API 操作的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付CloudTrail事件到 Amazon S3 儲存貯體，包括 Amazon 的事件GameLift。如果您不設定追蹤記錄，仍然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新的事件。使用收集的資訊CloudTrail，您可以判斷向 Amazon 發出的請求GameLift、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 使用者指南。

亞馬遜GameLift信息 CloudTrail

當您建立帳戶時，系統會在您的 AWS 帳戶 中啟用 CloudTrail。在 Amazon 中發生活動時GameLift，該活動會與事件歷史記錄中的其他AWS服務CloudTrail事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶 的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

對於您的事件的持續記錄AWS 帳戶，包括亞馬遜的事件GameLift，請創建一個跟踪。追蹤可CloudTrail將日誌檔交付到 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。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 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 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案

所有亞馬遜GameLift動作都會記錄下來，CloudTrail並記錄在 Amazon GameLift API 參考中。例如，呼叫`CreatePlayerSession`和`UpdateGameSession`動作會在CloudTrail記錄檔中產生項目。`CreateGameSession`

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

了解亞馬遜GameLift日誌文件條目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一個或多個日誌項目。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 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

以下範例說明 `CreateFleet` 和 `DescribeFleetAttributes` 動作的 CloudTrail 日誌項目。

```
{
    "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"
        },
    ]
```

```
}
```

在亞馬遜記錄服務器消息 GameLift

您可以在日誌檔中從 Amazon GameLift 伺服器擷取自訂伺服器訊息。配置日誌記錄的方式取決於您使用的是自定義服務器還是實時服務器（請參閱本章中的相應小節）。

主題

- 記錄伺服器訊息 (自訂伺服器)
- 記錄服務器消息（實時服務器）

記錄伺服器訊息 (自訂伺服器)

您可以在日誌檔中從 Amazon 自訂伺服器擷取 GameLift 自訂伺服器訊息。若要瞭解有關即時伺服器的記錄，請參閱記錄服務器消息（實時服務器）。

> **Important**
>
> 每個遊戲工作階段的記錄檔大小都有限制 (請參閱中的 Amazon GameLift 端點和配額 AWS 一般參考)。當遊戲工作階段結束時，Amazon 會將伺服器日誌 GameLift 上傳到亞馬遜簡單儲存服務 (Amazon S3)。Amazon 不 GameLift 會上傳超過限制的日誌。日誌可以非常快速地成長並超過大小限制。您應該監視您的日誌，並將日誌輸出限制為只有必要的消息。

設定自訂伺服器的記錄

使用 Amazon 自 GameLift 訂伺服器，您可以撰寫自己的程式碼來執行記錄，並將其設定為伺服器處理序組態的一部分。Amazon 會 GameLift 使用您的記錄組態來識別在每個遊戲工作階段結束時必須上傳到 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++

若要設定記錄 (C++)

1. 建立作為遊戲伺服器記錄檔目錄路徑的字串向量。

```
std::string serverLog("serverOut.log");        // Example server log file
```

```
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. 提供你的向量作為你LogParameters的ProcessParameters對象。

```
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() 時提供ProcessParameters物件。

```
Aws::GameLift::GenericOutcome outcome =
   Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

如需更完整的範例，請參閱ProcessReady()。

C#

若要設定記錄 (C#)

1. 建立作為遊戲伺服器記錄檔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. 提供您的列表作為您LogParameters的ProcessParameters對象。

```
var processReadyParameter = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(logPaths));
```

3. 當您呼叫 ProcessReady() 時提供ProcessParameters物件。

```
var processReadyOutcome =
   GameLiftServerAPI.ProcessReady(processReadyParameter);
```

如需更完整的範例，請參閱ProcessReady()。

寫入記錄

您的記錄檔會在伺服器處理序啟動之後存在。您可以使用任何寫入檔案的方法寫入記錄。若要擷取伺服器的所有標準輸出和錯誤輸出，請將輸出串流重新對應至記錄檔，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"));
```

存取伺服器記錄

當遊戲工作階段結束時，Amazon 會 GameLift 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記錄的位置，您可以使用 GetGameSessionLogUrlAPI 作業。若要下載記錄檔，請使用作業傳回的 URL。

記錄服務器消息（實時服務器）

您可以從日誌文件中的實時服務器捕獲自定義服務器消息。若要瞭解自訂伺服器的記錄，請參閱記錄伺服器訊息 (自訂伺服器)。

您可以將不同類型的訊息輸出至記錄檔 (請參閱在伺服器指令碼中記錄訊息)。除了自定義消息之外，您的實時服務器還使用相同的消息類型輸出系統消息，並寫入相同的日誌文件。您可以調整叢集的記錄等級，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記錄檔大小都有限制 (請參閱中的 Amazon GameLift 端點和配額 AWS 一般參考)。當遊戲工作階段結束時，Amazon 會將伺服器日誌 GameLift 上傳到亞馬遜簡單儲存服務 (Amazon S3)。Amazon 不 GameLift 會上傳超過限制的日誌。日誌可以非常快速地成長並超過大小限制。您應該監視您的日誌，並將日誌輸出限制為只有必要的消息。

在伺服器指令碼中記錄訊息

您可以在實時服務器的腳本中輸出自定義消息。請使用下列步驟將伺服器訊息傳送至記錄檔：

1. 創建一個可變的來保存對記錄器對象的引用。

```
var logger;
```

2. 在`init()`函數中，從會話對象獲取記錄器並將其分配給記錄器變量。

```
function init(rtSession) {
    session = rtSession;
    logger = session.getLogger();
}
```

3. 呼叫記錄器上的適當函數以輸出訊息。

   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...");
```

客戶經驗嚴重錯誤訊息

```
logger.cxfatal("This is my customer experience fatal error message...");
```

如需指令碼中記錄陳述式的範例，請參閱實時服務器腳本示例。

記錄檔中的輸出會指出訊息的類型
(DEBUG、、、、、CXFATAL)，，，，，INFO，，WARN，，ERROR，，FATAL，，，，，，，，，，，，，

```
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.
```

存取伺服器記錄

當遊戲工作階段結束時，Amazon 會 GameLift 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記錄的位置。使用 API 呼叫傳回的 URL 來下載記錄檔。

調整記錄層級

日誌可以非常快速地成長並超過大小限制。您應該監視您的日誌，並將日誌輸出限制為只有必要的消息。對於「即時伺服器」，您可以在叢集的執行階段設定中提供參數，以調整記錄層級`loggingLevel:`*`LOGGING_LEVEL`*，其中*`LOGGING_LEVEL`*為下列其中一個值：

1. debug
2. info(預設值)

3. warn
4. error
5. fatal
6. cxfatal

此列表的排序從最不嚴重（debug）到最嚴重（cxfatal）。您可以設定單一，loggingLevel而伺服器只會記錄該嚴重性層級或較高嚴重性層級的訊息。例如，設定loggingLevel:error會使叢集中的所有伺服器只會將errorfatal、和cxfatal訊息寫入記錄。

您可以在建立叢集時或執行後設定叢集的記錄等級。在船隊執行後變更其記錄等級，只會影響更新後建立的遊戲工作階段的記錄。任何現有遊戲工作階段的記錄都不會受到影響。如果您在建立叢集時未設定記錄等級，伺服器會預設將記錄等級設info定為。如需設定記錄層級的指示，請參閱下列各節。

創建實時服務器叢集時設置日誌記錄級別（控制台）

請依照中的指示建立亞馬遜 GameLift 受管叢集建立您的叢集，並加入下列項目：

- 在「程序管理」步驟的「伺服器處理作業配置」子步驟中，提供記錄層次索引鍵值配對 (例如loggingLevel:error) 作為 Launch 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分隔開來 (例如，loggingLevel:error +map Winter444)。

創建實時服務器叢集時設置日誌記錄級別（AWS CLI）

請依照中的指示建立亞馬遜 GameLift 受管叢集建立您的叢集，並加入下列項目：

- 在的--runtime-configuration參數的引數中create-fleet，提供記錄層級索引鍵值配對 (例如loggingLevel:error) 做為的值。Parameters使用非英數字元 (逗號除外)，將記錄層級與任何其他參數分隔開來。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}]"
```

為正在運行的實時服務器叢集（控制台）設置日誌記錄級別

請依照中的更新叢集組態指示使用 Amazon GameLift 主控台更新您的叢集，並加入下列項目：

- 在 [編輯叢集] 頁面的 [伺服器處理序配置] 下，提供記錄層級索引鍵值配對 (例如`loggingLevel:error`) 做為 Launch 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分隔開來 (例如，`loggingLevel:error +map Winter444`)。

為正在運行的實時服務器叢集設置日誌記錄級別（ AWS CLI ）

請依照中的更新叢集組態指示使用更新您的叢集AWS CLI，並加入下列項目：

- 在的`--runtime-configuration`參數的引數中`update-runtime-configuration`，提供記錄層級索引鍵值配對 (例如`loggingLevel:error`) 做為的值。`Parameters`使用非英數字元 (逗號除外)，將記錄層級與任何其他參數分隔開來。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 的安全性 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

雲安全 AWS 是最高的優先級。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。

安全是 AWS 與您之間共同的責任。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 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要了解適用於 Amazon 的合規計劃 GameLift，請參閱合規計劃AWS 服務範圍內的合規計劃AWS的服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您還需要對其他因素負責，包括數據的敏感性，公司的要求以及適用的 l AWS 和法規。

本文件可協助您了解如何在使用 Amazon 時應用共同的責任模型 GameLift。下列主題說明如何設定 Amazon GameLift 以符合安全和合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 Amazon GameLift 資源的服務。

主題

- Amazon 的數據保護 GameLift
- Amazon 的身份和訪問管理 GameLift
- 使用亞馬遜記錄和監控 GameLift
- Amazon 的合規驗證 GameLift
- Amazon 的韌性 GameLift
- Amazon 基礎設施安全 GameLift
- Amazon 中的配置和漏洞分析 GameLift
- Amazon 的安全最佳實踐 GameLift

Amazon 的數據保護 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

AWS 共同責任模型適用於 Amazon 中的資料保護 GameLift。如此模型中所述， AWS 負責保護執行所有 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，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。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 資源進行通訊。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。
- 使用 AWS 加密解決方案，以及其中的所有默認安全控制 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。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，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) 140-2 概觀。

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。這包括當您使用控制台，API GameLift 或 AWS SDK 與 Amazon 或其他 AWS 服務 AWS CLI人合作時。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 特定資料的處理方式如下：

- 您上傳到亞馬遜的遊戲伺服器組建和指令碼 GameLift 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。授權使用者可以取得上傳檔案的臨時存取權，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 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 中存放一段有限的時間。獲授權的使用者可以透過 Amazon GameLift 主控台中的連結或呼叫服務 API 來存取日誌資料。
- 指標和事件資料存放在 Amazon 中 GameLift ，可透過 Amazon 主 GameLift 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、遊戲工作階段和玩家工作階段上擷取資料。數據也可以通過 Amazon CloudWatch 和 CloudWatch 事件訪問。
- 客戶提供的資料會儲存在 Amazon GameLift 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。

如需有關資料保護的詳細資訊，請參閱AWS 安全部落格上的AWS 共同責任模型和 GDPR 部落格文章。

靜態加密

Amazon GameLift 特定資料的靜態加密處理方式如下：

- 遊戲伺服器組建和指令碼存放在具有伺服器端加密的 Amazon S3 儲存貯體中。
- 客戶提供的資料會以加密格式儲存 GameLift 在 Amazon 中。

傳輸中加密

與 Amazon GameLift API 的連線是透過安全 (SSL) 連線建立，並使用AWS 簽章版本 4 進行驗證 (透過 AWS CLI 或 AWS SDK 連線時，簽署會自動處理)。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 資源上託管的自定義遊戲服務器，通信不涉及 Amazon GameLift 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。
- 對於啟用了 TLS 證書生成的實時服務器，使用實時客戶端 SDK 的遊戲客戶端和實時服務器之間的流量在飛行中被加密。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

您可以安全地遠端存取 Amazon GameLift 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 Amazon GameLift FleetIQ，使用 AWS 系統管理員工作階段管理員和執行命令對執行個體的遠端存取會使用 TLS 1.2 進行加密，而建立連線的請求則使用 Sigv4 簽署。如需連線到受管 Amazon GameLift 執行個體的說明，請參閱遠端連線至 Amazon GameLift 叢集執行個體。

Amazon 的身份和訪問管理 GameLift

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全控制對 AWS 資源的存取權限。IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon GameLift 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

物件

您的使用方式 AWS Identity and Access Management (IAM) 會有所不同，具體取決於您在 Amazon 所做的工作 GameLift。

服務使用者 — 如果您使用 Amazon GameLift 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon GameLift 功能完成工作時，您可能需要額外的許可。瞭解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在 Amazon 中存取某個功能 GameLift，請參閱疑難排解 Amazon GameLift 身分和存取。

服務管理員 — 如果您負責公司的 Amazon GameLift 資源，則可能擁有對 Amazon 的完全訪問權限 GameLift。判斷服務使用者應存取哪些 Amazon GameLift 功能和資源是您的工作。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如何將 IAM 與 Amazon 搭配使用 GameLift，請參閱Amazon 如何與 IAM 合 GameLift 作。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策來管理 Amazon 存取權限的詳細資訊 GameLift。若要檢視可在 IAM 中使用的 Amazon GameLift 身分型政策範例，請參閱。Amazon 的基於身份的政策示例 GameLift

使用身分驗證

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者身分，或擔任 IAM 角色進行 驗證 (登入至 AWS)。

您可以使用透過身分來源 AWS IAM Identity Center 提供的憑證，以聯合身分登入 AWS。(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證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。您 AWS 藉由使用聯合進行存取時，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。如需登入至 AWS 的相關資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的相關資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

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 建議您使用多重要素驗證 (MFA) 以提高帳戶的安全。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》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。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

最佳實務是要求人類使用者 (包括需要管理員存取權的使用者)搭配身分提供者使用聯合功能，使用暫時憑證來存取 AWS 服務。

聯合身分是來自您企業使用者目錄的使用者、Web 身分供應商、AWS Directory Service、Identity Center 目錄或透過身分來源提供的憑證來存取 AWS 服務 的任何使用者。聯合身分存取 AWS 帳戶時，會擔任角色，並由角色提供暫時憑證。

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，也可以連線並同步到自己身分來源中的一組使用者和群組，以便在您的所有 AWS

帳戶和應用程式中使用。如需 IAM Identity Center 的相關資訊，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 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，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。如需詳細資訊，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。您無法以群組身分登入。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過程變得更為容易。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，但角色僅提供暫時憑證。若要進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法的相關資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 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，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-idp.html中的為第三方身分提供者建立角色。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 (信任的委託人)存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而非使用角色作為代理)。若要了解跨帳戶存取角色和資源型政策間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。

- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。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。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，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給 AWS 服務 服務。
  - 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。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 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。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暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其附加到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體 (使用者、根使用者或角色工作階段)發出請求時評估這些政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文件形式儲存在 AWS 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哪些內容。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。若要授與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。具備該政策的使用者便可以從 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。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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。受管政策則是獨立的政策，您可以將這些政策附加到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需瞭解如何在受管政策及內嵌政策間選擇，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。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。主體可以包括帳戶、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。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (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 Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 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 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 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。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。
- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。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 AWS 帳戶根使用者。如需組織和 SCP 的更多相關資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。如需瞭解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱 IAM 使用者指南中的政策評估邏輯。

Amazon 如何與 IAM 合 GameLift 作

在您使用 IAM 管理 Amazon 的存取權限之前 GameLift，請先了解哪些 IAM 功能可用於 Amazon GameLift。

您可以與 Amazon 搭配使用的 IAM 功能 GameLift

IAM 功能	Amazon GameLift 支持
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

IAM 功能	Amazon GameLift 支持
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

若要深入瞭解 Amazon GameLift 和其他AWS服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS服務。

Amazon 的基於身份的政策 GameLift

支援身分型政策	是

身分型政策是可以連接到身分 (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。若要瞭解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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，因為這會套用至附加的使用者或角色。如要瞭解您在 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 的基於身份的政策示例 GameLift

若要檢視 Amazon GameLift 身分型政策的範例，請參閱。Amazon 的基於身份的政策示例 GameLift

Amazon 內基於資源的政策 GameLift

支援以資源基礎的政策	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。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。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

若要啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作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。當主體和資源在不同的 AWS 帳戶 中時，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 存取資源的許可。其透過將身分型政策附加到實體來授予許可。不過，如果資源型政策會為相同帳戶中的主體授與存取，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 角色與資源型政策有何差異。

Amazon 的政策行動 GameLift

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哪些內容。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。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作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與執行相關聯操作的許可。

如需 Amazon GameLift 動作的清單，請參閱服務授權參考 GameLift中 Amazon 定義的動作。

Amazon 中的政策動作會在動作之前 GameLift 使用下列前置詞：

```
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 身分型政策的範例，請參閱。Amazon 的基於身份的政策示例 GameLift

Amazon 政策資源 GameLift

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哪些內容。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 (例如列出作業)，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 資源類型及其 ARN 的清單，請參閱服務授權參考 GameLift中 Amazon 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon GameLift 定義的動作。

某些 Amazon GameLift 資源具有 ARN 值，可讓資源使用 IAM 政策來管理其存取權限。Amazon GameLift 叢集資源具有以下語法的 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叢集，請使用萬用字元 (*)：

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

若要檢視 Amazon GameLift 身分型政策的範例，請參閱。Amazon 的基於身份的政策示例 GameLift

Amazon 的政策條件密鑰 GameLift

支援服務特定政策條件索引鍵	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在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與該 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 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 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 條件金鑰的清單，請參閱服務授權參考 GameLift中的 Amazon 條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon 定義的動作 GameLift。

若要檢視 Amazon GameLift 身分型政策的範例，請參閱。Amazon 的基於身份的政策示例 GameLift

Amazon 的 ACL GameLift

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

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。您可以將標籤附加到 IAM 實體 (使用者或角色)，以及許多 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。

若要根據標籤控制存取，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索引鍵，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索引鍵，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。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)。

如需以身分識別為基礎的政策範例，該策略會根據該資源上的標籤限制對資源的存取，請參閱。根據標籤查看 Amazon GameLift 車隊

使用 Amazon 臨時登入資料 GameLift

支援臨時憑證	是

您使用臨時憑證進行登入時，某些 AWS 服務 無法運作。如需詳細資訊，包括那些 AWS 服務 搭配臨時憑證運作，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

如果您使用使用者名稱和密碼之外的任何方法登入 AWS Management Console，則您正在使用臨時憑證。例如，當您使用公司的單一登入(SSO)連結存取 AWS 時，該程序會自動建立臨時憑證。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 使用者指南中的切換至角色 (主控台)。

您可使用 AWS CLI 或 AWS API，手動建立臨時憑證。接著，您可以使用這些臨時憑證來存取 AWS。AWS 建議您動態產生臨時憑證，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon 的跨服務主體許可 GameLift

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在 AWS 中執行動作時，您會被視為主體。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 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，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 的服務角色 GameLift

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可可可可能會中斷 Amazon GameLift 功能。只有在 Amazon GameLift 提供指導時才編輯服務角色。

允許您的亞馬遜 GameLift託管遊戲伺服器存取其他AWS資源，例如AWS Lambda功能或 Amazon DynamoDB 資料庫。由於遊戲伺服器託管在 Amazon GameLift 管理的叢集上，因此您需要一個可讓 Amazon GameLift 有限存取其他AWS資源的服務角色。如需詳細資訊，請參閱 與您車隊的其他AWS資源進行溝通。

Amazon 的服務連結角色 GameLift

支援服務連結角色。	否

服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。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AWS務連結角色的詳細資訊，請參閱 IAM 使用者指南中的與 IAM 搭配使用的服務。在表格中尋找「服務連結角色」欄**Yes**中包含的服務。選擇是以檢視該服務的服務連結角色文件。

Amazon 的基於身份的政策示例 GameLift

依預設，使用者和角色沒有建立或修改 Amazon GameLift 資源的權限。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 執行任務。若要授與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。

若要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。

有關 Amazon 定義的動作和資源類型的詳細資訊 GameLift，包括每種資源類型的 ARN 格式，請參閱服務授權參考 GameLift中適用於 Amazon 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 Amazon GameLift 控制台
- 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
- 允許訪問一個 Amazon GameLift 隊列
- 根據標籤查看 Amazon GameLift 車隊
- 在 Amazon S3 中訪問遊戲構建文件

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon GameLift 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 受管政策並朝向最低權限許可的目標邁進：如需開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我

們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。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。

- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務 (例如 AWS CloudFormation)使用條件。如需更多資訊，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，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA)：如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 最佳安全實務。

使用 Amazon GameLift 控制台

若要存取 Amazon GameLift 主控台，您必須擁有最少一組許可。這些許可必須允許您列出和檢視有關AWS 帳戶. GameLift 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。

為確保這些實體仍然可以使用 Amazon GameLift 主控台，請使用以下範例和中的語法向使用者和群組新增許可管理員權限範例。如需詳細資訊，請參閱 管理 Amazon 的用戶許可 GameLift。

GameLift 透過AWS CLI或 AWS API 操作使用 Amazon 的使用者不需要最低主控台許可。相反地，您可以限制只存取使用者需要執行的作業。例如，代表遊戲客戶端行事的玩家用戶需要訪問以請求遊戲會話，將玩家放入遊戲以及其他任務。

如需使用所有 Amazon GameLift 主控台功能所需許可的相關資訊，請參閱中的管理員的許可語法管理員權限範例。

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上，或是使用 AWS CLI 或 AWS API 透過編寫程式的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

若要將玩家置於遊戲工作階段中，遊戲用戶端和後端服務需要權限。如需這些案例的原則範例，請參閱玩家使用者權限範例。

允許訪問一個 Amazon GameLift 隊列

下列範例提供使用者存取特定 Amazon GameLift 佇列的權限。

此原則會透過下列動作授與使用者新增、更新和刪除佇列目的地的權限：gamelift:UpdateGameSessionQueuegamelift:DeleteGameSessionQueue、和gamelift:DescribeGameSessionQueues。如圖所示，此原則會使用Resource元素來限制對單一佇列的存取：gamesessionqueue/examplequeue123。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ViewSpecificQueueInfo",
         "Effect":"Allow",
         "Action":[
            "gamelift:DescribeGameSessionQueues"
         ],
         "Resource":"arn:aws:gamelift:::gamesessionqueue/examplequeue123"
      },
      {
         "Sid":"ManageSpecificQueue",
         "Effect":"Allow",
         "Action":[
            "gamelift:UpdateGameSessionQueue",
            "gamelift:DeleteGameSessionQueue"
         ],
         "Resource":"arn:aws:gamelift:::gamesessionqueue/examplequeue123"
      }
   ]
}
```

根據標籤查看 Amazon GameLift 車隊

您可以使用身分型政策中的條件，根據標籤控制對 Amazon GameLift 資源的存取。此範例顯示如何建立策略，以便在Owner標記符合使用者的使用者名稱時檢視叢集。此原則也會授與在主控台中完成此作業所需的權限。

```
{
    "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": {"gamelift:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

在 Amazon S3 中訪問遊戲構建文件

將遊戲伺服器與亞馬遜整合後 GameLift，請將建置檔案上傳到 Amazon S3。對於 Amazon GameLift 訪問構建文件，請使用以下策略。

```
{
    "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 遊戲檔案的詳細資訊，請參閱將自訂伺服器組建上傳到亞馬遜 GameLift。

疑難排解 Amazon GameLift 身分和存取

使用下列資訊協助您診斷和修正使用 Amazon GameLift 和 AWS Identity and Access Management (IAM) 時可能會遇到的常見問題。

主題

我沒有授權在 Amazon 執行操作 GameLift

如果AWS Management Console告訴您您沒有執行動作的授權，請聯絡您的AWS帳戶管理員以尋求協助。您的管理員是為您提供登入憑證的人員。

當 `mateojackson` IAM 使用者嘗試使用主控台檢視佇列的詳細資料但沒有`gamelift:DescribeGameSessionQueues`權限時，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策略，以允許他使用`gamelift:DescribeGameSessionQueues`動作讀取`examplequeue123`資源的存取權。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行`iam:PassRole`動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon GameLift。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須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。

當名為的 IAM 使用者`marymajor`嘗試使用主控台在 Amazon 中執行動作時，會發生下列範例錯誤 GameLift。但是，該動作要求服務具備服務角色授與的許可。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。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我想允許我以外的人訪AWS 帳戶問我的 Amazon GameLift 資源

您可以建立一個角色，讓其他帳戶中的使用者或您的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 是否 GameLift 支援這些功能，請參閱Amazon 如何與 IAM 合 GameLift 作。
- 如需了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 如需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擁有的 AWS 帳戶。
- 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)。
- 若要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 角色與資源型政策的差異。

使用亞馬遜記錄和監控 GameLift

監控是維持 Amazon GameLift 和AWS解決方案的可靠性、可用性和效能的重要組成部分。您應該收集 AWS 解決方案全面的監控資料，以便在出現多點故障時更輕鬆地進行偵錯。

AWSAmazon 也GameLift提供數種工具來監控您的遊戲託管資源並回應潛在事件。

亞馬遜CloudWatch警報

使用 Amazon CloudWatch 警示，您可以在指定的時間段內觀看單一指標。如果指標超過指定的閾值，會傳送一則通知至 Amazon SNS 主題或 AWS Auto Scaling 政策。CloudWatch警示會在其狀態變更時觸發，並且會在指定的期間數目 (而非處於特定狀態) 維護警示。如需詳細資訊，請參閱監控亞馬遜GameLift與亞馬遜 CloudWatch。

AWSCloudTrail日誌

CloudTrail提供 Amazon 中使用者、角色或AWS服務所採取的動作記錄GameLift。使用收集的資訊 CloudTrail，您可以判斷向 Amazon 發出的請求GameLift、提出請求的 IP 地址、提出請求的人員、

提出請求的時間以及其他詳細資訊。如需詳細資訊，請參閱 記錄亞馬遜 GameLift API 呼叫 AWS CloudTrail。

Amazon 的合規驗證 GameLift

Amazon 不 GameLift 在任何 AWS 合規計劃的範圍內。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、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遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

  > **Note**
  >
  > 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的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 提供了內部安全狀態的全面視圖 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 controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。

- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon 的韌性 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。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。

如需區域和可用區域的詳 AWS 細資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎設施之外，Amazon 還 GameLift 提供下列功能來協助支援您的資料備援需求：

- 多區域佇列 — Amazon GameLift 遊戲工作階段佇列用於透過可用託管資源放置新的遊戲工作階段。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，請參閱設計遊戲工作階段佇列。
- 自動容量擴展 — 使用 Amazon GameLift 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，請參閱擴展亞馬遜GameLift託管容量。
- 跨執行個體分配 — Amazon 會根據叢集大小， GameLift將傳入流量分散到多個執行個體。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 — 上傳到 Amazon 的遊戲伺服器組建和指令碼會使用標準儲存類別存放在 Amazon S3，該類別使用多個資料中心複寫來提高彈性。 GameLift 遊戲工作階段日誌也會使用標準儲存類別存放在 Amazon S3 中。

Amazon 基礎設施安全 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

作為受管服務，Amazon GameLift 受到 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫 GameLift 透過網路存取 Amazon。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 服務會將所有機隊置於 Amazon 虛擬私有雲 (VPC)，以便每個叢集都存在於雲端中邏輯隔離的區域中 AWS 。您可以使用 Amazon GameLift 政策來控制來自特定 VPC 端點或特定 VPC 的存取。實際上，這可以將對指定 Amazon GameLift 資源的網路存取從網路內的特定 VPC 隔離出來 AWS 。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 中的配置和漏洞分析 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

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 共用的責任模型。 AWS 處理基本安全性工作，例如客體作業系統 (OS) 和資料庫修補、防火牆組態和嚴重損壞修復。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)。

下列安全最佳實務也可解決 Amazon 中的組態和漏洞分析問題 GameLift：

- 客戶必須負責管理部署到 Amazon GameLift 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，請將新組建上傳至 Amazon GameLift、為其建立新叢集，然後將流量重新導向至新叢集。
  - 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。為了修補、更新和保護屬於 AMI 一部分的作業系統和其他應用程式，不論遊戲伺服器更新為何，都會定期回收機群。
- 客戶應考慮定期使用最新的 SDK 版本更新遊戲，包括 AWS SDK、Amazon GameLift Server SDK 和適用於即時伺服器的 Amazon GameLift 用戶端開發套件。

Amazon 的安全最佳實踐 GameLift

如果您使用 Amazon GameLift FleetIQ 做為 Amazon EC2 的獨立功能，請參閱亞馬遜 EC2 使用者指南中的 Amazon EC2 中的安全性。

GameLift Amazon 在開發和實作自己的安全政策時，提供許多安全功能供您考慮。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。

不要開啟連接至網際網路的連接埠

我們強烈建議您不要開啟網際網路連接埠，因為這樣做會造成安全性風險。例如，如果您使UpdateFleetPortSettings用這樣開啟遠端桌面連接埠：

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
      {
        "FromPort": 3389,
        "IpRange": "0.0.0.0/0",
        "Protocol": "RDP",
        "ToPort": 3389
      }
  ]
}
```

那麼你允許互聯網上的任何人訪問該實例。

請改為開啟具有特定 IP 位址或位址範圍的連接埠。例如，像這樣：

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
      {
        "FromPort": 3389,
        "IpRange": "54.186.139.221/32",
        "Protocol": "TCP",
        "ToPort": 3389
      }
  ]
}
```

進一步了解

有關如何更安全地使用 Amazon 的 GameLift 更多信息，請參閱AWS Well-Architected Tool 安全性支柱。 。

亞馬遜GameLift參考指南

本節包含使用 Amazon 的參考文件GameLift。

主題

- 亞馬遜GameLift服務 API 參考 (AWSSDK)
- 亞馬遜GameLift實時服務器參
- 亞馬遜GameLift服務器 SDK 參考
- 遊戲工作階段安置事

亞馬遜GameLift服務 API 參考 (AWSSDK)

本主題提供與 Amazon 託管主機解決方案搭配使用的 API 操作清單，包括託GameLift管自訂遊戲伺服器和即時伺服器。這些作業會封裝到命`aws.gamelift`名空間中的 AWS SDK 中。 下載開AWS發套件或檢視亞馬遜 GameLift API 參考文件。

該 API 包含兩組用於託管遊戲託管的操作：

- 設置和管理亞馬遜GameLift託管資源
- 開始遊戲工作階段並加入玩家

亞馬遜GameLift服務 API 還包含與其他亞馬遜GameLift工具和解決方案一起使用的操作。如需 FleetIQ API 的清單，請參閱 F le etIQ API 動作。如需配對的 FlexMatch API 清單，請參閱 FlexMatchAPI 動作。

設置和管理亞馬遜GameLift託管資源

呼叫這些作業以設定遊戲伺服器的託管資源、擴充容量以符合玩家需求、存取效能和使用率指標等。這些 API 操作可用於 Amazon 上託管的遊戲伺服器GameLift，包括即時伺服器。您可以使用 Amazon GameLift 主控台執行大部分的資源管理任務，也可以使用 AWS Command Line Interface (AWS CLI) 工具或 AWS SDK 撥打服務。

準備要部署的遊戲伺服器

上傳並配置遊戲的遊戲服務器代碼，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

- 上傳建置 — 從本機路徑上傳建置檔案，並建立新的 Amazon GameLift 建置資源。此操作僅作為一個AWS CLI命令使用，是上傳遊戲伺服器組建的最常用方法。
- CreateBuild— 使用存放在 Amazon S3 儲存貯體中的檔案建立新組建。
- ListBuilds— 獲取上傳到亞馬遜GameLift區域的所有構建的列表。
- DescribeBuild— 擷取與組建相關聯的資訊。
- UpdateBuild— 變更組建中繼資料，包括組建名稱和版本。
- DeleteBuild— 從亞馬遜刪除構建GameLift。

管理實時服務器配置腳本

- CreateScript— 上傳JavaScript檔案並建立新的 Amazon GameLift 指令碼資源。
- ListScripts— 獲取上傳到亞馬遜GameLift區域的所有實時腳本的列表。
- DescribeScript— 擷取與「即時」指令碼相關聯的資訊。
- UpdateScript— 更改腳本元數據並上傳修改後的腳本內容。
- DeleteScript— 從亞馬遜刪除實時腳本GameLift.

設定託管的運算資源

配置託管資源並使用遊戲服務器構建或實時配置腳本部署它們。

建立和管理車隊

- CreateFleet— 配置和部署新的 Amazon 運算資源GameLift集，以執行遊戲伺服器。部署完成後，遊戲伺服器會自動依設定啟動，並準備好主持遊戲工作階段。
- ListFleets— 獲取亞馬遜GameLift地區所有艦隊的列表。
- DeleteFleet— 終止不再運行遊戲服務器或託管玩家的艦隊。
- 檢視/更新叢集位置。
  - CreateFleetLocations— 將遠端位置新增至支援多個位置的現有叢集
  - DescribeFleetLocationAttributes— 取得叢集所有遠端位置的清單，並檢視每個位置的目前狀態。
  - DeleteFleetLocations— 從支援多個位置的叢集中移除遠端位置。
- 檢視/更新機群組態。
  - DescribeFleetAttributes/UpdateFleetAttributes— 檢視或變更叢集的中繼資料和遊戲工作階段保護和資源建立限制的設定。

- DescribeFleetPortSettings/UpdateFleetPortSettings— 檢視或變更叢集允許的輸入權限 (IP 位址和連接埠設定範圍)。
- DescribeRuntimeConfiguration/UpdateRuntimeConfiguration— 檢視或變更叢集中每個執行個體要執行的伺服器處理序 (以及多少個)。

管理車隊容量

- 描述beEC2 InstanceLimits — 擷取目前AWS帳戶和目前使用層級允許的執行個體數目上限。
- DescribeFleetCapacity— 檢索車隊本地區域的當前容量設置。
- DescribeFleetLocationCapacity— 擷取多位置叢集每個位置的目前容量設定。
- UpdateFleetCapacity— 手動調整叢集的容量設定。
- 設定自動調整規模：
  - PutScalingPolicy— 開啟以目標為基礎的自動調整規模功能，或建立自訂自動調整規模政策，或更新現有政策。
  - DescribeScalingPolicies— 擷取現有的自動調整規模政策。
  - DeleteScalingPolicy— 刪除自動調整規模政策並阻止其影響叢集的容量。
  - StartFleetActions— 重新啟動叢集的自動調整規模政策。
  - StopFleetActions— 暫停叢集的自動調整規模政策。

監控機群活動。

- DescribeFleetUtilization— 檢索有關當前在機隊中活躍的服務器進程，遊戲會話和玩家的數量的統計信息。
- DescribeFleetLocationUtilization— 擷取多位置叢集中每個位置的使用率統計資料。
- DescribeFleetEvents— 檢視指定時間範圍內叢集的記錄事件。
- DescribeGameSessions— 檢索遊戲會話元數據，包括遊戲的運行時間和當前玩家數量。

設定佇列以實現最佳遊戲工作階段放置

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— 建立佇列以供處理遊戲工作階段刊登位置的要求時使用。
- DescribeGameSessionQueues— 擷取 Amazon GameLift 區域中定義的遊戲工作階段佇列。

- UpdateGameSessionQueue— 更改遊戲會話隊列的配置。
- DeleteGameSessionQueue— 從該地區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— 定義新別名，並選擇性地將其指派給叢集。
- ListAliases— 獲取在亞馬遜GameLift區域中定義的所有車隊別名。
- DescribeAlias— 擷取現有別名的資訊。
- UpdateAlias— 變更別名的設定，例如將其從一個叢集重新導向至另一個叢集。
- DeleteAlias— 從區域移除別名。
- ResolveAlias— 取得指定別名指向的叢集 ID。

存取託管執行個體

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— 取得叢集中每個執行個體的相關資訊，包括執行個體 ID、IP 位址、位置和狀態。
- GetInstanceAccess— 請求遠端連線至叢集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 託管資源和其AWS他資源之間的 VPC 對等連接。

- CreateVpcPeeringAuthorization— 授權與其中一個 VPC 的對等連線。
- DescribeVpcPeeringAuthorizations— 擷取有效的對等連線授權。
- DeleteVpcPeeringAuthorization— 刪除對等連線授權。
- CreateVpcPeeringConnection— 在 Amazon GameLift 叢集的 VPC 與其中一個 VPC 之間建立對等連接。
- DescribeVpcPeeringConnections— 擷取 Amazon GameLift 叢集的作用中或擱置中 VPC 對等連線的資訊。
- DeleteVpcPeeringConnection— 刪除與亞馬遜GameLift叢集的 VPC 對等連接。

開始遊戲工作階段並加入玩家

從您的遊戲用戶端服務呼叫這些操作，以開始新的遊戲工作階段、取得現有遊戲工作階段的資訊，並加入玩家參加遊戲工作階段。這些操作適用於 Amazon 上託管的自訂遊戲伺服器GameLift。如果您使用的是即時伺服器，請使用即時伺服器用戶端 API (C#) 參考.

- 為一或多個玩家啟動新的遊戲工作階段。
  - StartGameSessionPlacement— 要求亞馬遜找GameLift到最佳的可用託管資源並開始新的遊戲會話。這是創建新遊戲會話的首選方法。它依賴於遊戲會話隊列來跟踪多個區域的託管可用性，並使用 FleetIQ 算法根據玩家延遲，託管成本，位置等優先放置位置。
  - DescribeGameSessionPlacement— 取得刊登位置要求的詳細資訊和狀態。
  - StopGameSessionPlacement— 取消放置請求。
  - CreateGameSession— 在特定艦隊位置開始一個新的空遊戲會話。此操作使您可以更好地控制開始遊戲會話的位置，而不是使用 FleetIQ 來評估放置選項。您必須在單獨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。查找帶有可用玩家位置的正在運行的遊戲會話，並將其保留給新玩家。
  - CreatePlayerSession-保留一個開放的插槽供玩家加入遊戲會話。
  - CreatePlayerSessions-保留開放的插槽，供多個玩家加入遊戲會話。
- 使用遊戲工作階段和玩家工作階段資料。管理有關遊戲會話和玩家會話的信息。
  - SearchGameSessions— 根據一組搜索條件請求活動遊戲會話列表.
  - DescribeGameSessions— 檢索特定遊戲會話的元數據，包括活動時間長度和當前玩家數量。
  - DescribeGameSessionDetails— 檢索一個或多個遊戲會話的元數據，包括遊戲會話保護設置。
  - DescribePlayerSessions— 獲取有關玩家活動的詳細信息，包括狀態，播放時間和播放器數據。
  - UpdateGameSession— 更改遊戲會話設置，例如最大玩家數量和加入策略。
  - GetGameSessionLogUrl— 獲取遊戲會話保存日誌的位置.

亞馬遜GameLift實時服務器參

本節包含 Amazon GameLift 即時伺服器開發套件的參考文件。它包括實時客戶端 API 以及配置實時服務器腳本的指導。

主題

- 即時伺服器用戶端 API (C#) 參考

- 亞馬遜GameLift即時伺服器指令碼

即時伺服器用戶端 API (C#) 參考

使用即時用戶端 API 準備您的多人遊戲用戶端，以便與 Amazon GameLift 即時伺服器搭配使用。如需整合流程的詳細資訊，請參閱準備您的實時服務器。Client API 包含一組同步 API 呼叫和非同步回呼，可讓遊戲用戶端連線到即時伺服器，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

若要設定即時用戶端 API

1. 下載亞馬遜GameLift實時客戶端開發套件。
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 SDK 程式庫，請還原NuGet套件並建置解決方案。
3. 將實時客戶端庫添加到您的遊戲客戶端項目中。

實時服務器客戶端 API（C＃）參考：操作

此 C# 即時用戶端 API 參考可協助您準備多人遊戲，以便與部署在 Amazon GameLift 叢集上的即時伺服器搭配使用。如需整合流程的詳細資訊，請參閱 準備您的實時服務器。

- 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

endpoint

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呼叫 AWSSDK Amazon GameLift API 動作StartGameSessionPlacementCreateGameSession、或DescribeGameSessions。

> **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。此資訊在物件中指定，該GameSession物件會回應StartGameSessionPlacementCreateGameSession、或DescribeGameSession要求而傳回。

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

返回一個ConnectionStatus枚舉值，指示客戶端的連接狀態。

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。

實時服務器客戶端 API（C# ）參考：異步回調

使用此 C# 即時用戶端 API 參考可協助您準備多人遊戲，以便與部署在 Amazon GameLift 叢集上的即時伺服器搭配使用。如需整合流程的詳細資訊，請參閱 準備您的實時服務器。

- 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。實時服務器調用這些回調以將遊戲相關信息發送到遊戲客戶端。同一事件的回調也可以使用實時服務器腳本中的自定義遊戲邏輯來實現。請參閱 實時服務器的腳本回調。

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()

當遊戲客戶端收到來自實時服務器的消息時調用。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

dataReceivedEvent參數

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

groupMembershipEvent參數

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。

實時服務器客戶端 API（C# ）參考：數據類型

此 C# 即時用戶端 API 參考可協助您準備多人遊戲，以便與部署在 Amazon GameLift 叢集上的即時伺服器搭配使用。如需整合流程的詳細資訊，請參閱 準備您的實時服務器。

- 同步動作
- 非同步回呼
- 資料類型

ClientConfiguration

有關遊戲客戶端如何連接到實時服務器的信息。

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

有關要求與實時服務器連接的遊戲客戶端和/或玩家的信息。

目錄

playerSessionId

創建新玩家會話GameLift時由亞馬遜發出的唯一 ID。玩家工作階段 ID 是在PlayerSession物件中指定的，該物件是為了回應用戶端對 GameLiftAPI 動作StartGameSessionPlacement、CreateGameSessionDescribeGameSessionPlacement、或的呼叫而傳回DescribePlayerSessions。

類型：字串

必要：是

payload

開發人員定義的信息將在連接時傳達給實時服務器。這包括可能用於自訂登入機制的任何資料。例如，有效負載可能會在允許客戶端連接之前提供由實時服務器腳本處理的身份驗證信息。

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 () 建立的訊息已經設定了作業代碼，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枚舉

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

data

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

為實時客戶端 SDK 定義的枚舉定義如下：

ConnectionStatus

- 連接 — 遊戲客戶端僅通過 TCP 連接連接連接到實時服務器。無論交付意圖為何，所有訊息的傳送都是透過 TCP。
- 連接到快速 — 遊戲客戶端通過 TCP 和 UDP 連接連接到實時服務器。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。
- 連接到遊戲客戶端通過 TCP 和 UDP 連接連接到實時服務器。遊戲用戶端可以使用 TCP 或 UDP 傳送和接收訊息。
- CONNECTING 遊戲客戶端已發送連接請求，並且實時服務器正在處理它。
- 中斷連接客戶端-遊戲客戶端與實時服務器的響應來自遊戲客戶端的請求而斷開連接。Disconnect()
- [中斷連線] — 由於用戶端中斷連線呼叫以外的原因，遊戲用戶端與即時伺服器中斷連線。

ConnectionType

- 安全連線類型

  用於在已產生 TLS 憑證的GameLift叢集上執行的即時伺服器。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。
- 不安全 — 非安全連線類型。

- 網路通訊端 — 非安全連線類型。不再偏好使用此值。

DeliveryIntent

- 快速 — 使用 UDP 通道交付。
- 可靠 — 使用 TCP 連線傳送。

亞馬遜GameLift即時伺服器指令碼

使用這些資源在您的實時腳本中構建自定義邏輯。

主題

- 實時服務器的腳本回調
- 實時服務器接口

實時服務器的腳本回調

您可以通過在實時腳本中實現這些回調來提供自定義邏輯來響應事件。

初始化

初始化實時服務器並接收實時服務器接口。

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()
```

onStartGame階段

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

當 Amazon GameLift 服務終止伺服器處理序時叫用。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)
```

onSendTo玩家

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)
```

onSendTo集團

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))
```

onPlayerJoin集團

當玩家傳送加入群組的請求時叫用。

語法

```
onPlayerJoinGroup(groupId, peerId)
```

onPlayerLeave集團

當玩家傳送離開群組的請求時叫用。

語法

```
onPlayerLeaveGroup(groupId, peerId)
```

實時服務器接口

當實時腳本初始化時，將返回到實時服務器的接口。本主題會說明透過界面可使用的屬性和方法。進一步瞭解如何撰寫即時指令碼，並在中檢視詳細的指令碼範例創建實時腳本。

「即時」介面可讓您存取下列物件：

- 工作階段
- player
- gameMessage
- 組態

實時會話對象

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()
```

broadcastGroupMembership更新（ ）

觸發將更新後的群組成員資格清單交付至玩家群組。指定要廣播的成員資格 (groupIdTo廣播) 和要接收更新的群組 (targetGroupId)。群組 ID 必須是正整數或「-1」以表示所有群組。如實時服務器腳本示例需使用者定義群組 ID 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()

觸發即時伺服器以終止遊戲伺服器。必須從 Realtime 腳本調用此函數，才能從遊戲會話中乾淨地退出。

語法

```
rtSession.processEnding()
```

getGameSession身份證 ()

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`newTextGameMessage`或`newBinaryGameMessage`從即時伺服器建立的訊息傳送至使用 UDP 通道的播放器收件者。使用玩家的對等端 ID 識別收件人。

語法

```
rtSession.sendMessage(gameMessage, targetPlayer)
```

sendGroupMessage()

使用newTextGameMessage或newBinaryGameMessage從即時伺服器建立的訊息傳送給使用 UDP 頻道的玩家群組中的所有玩家。群組 ID 必須是正整數或「-1」以表示所有群組。如實時服務器腳本示例需使用者定義群組 ID 的範例，請參閱。

語法

```
rtSession.sendGroupMessage(gameMessage, targetGroup)
```

sendReliableMessage()

使用 TCP 通道將使用newTextGameMessage或newBinaryGameMessage從即時伺服器建立的訊息傳送給播放器收件者。使用玩家的對等端 ID 識別收件人。

語法

```
rtSession.sendReliableMessage(gameMessage, targetPlayer)
```

sendReliableGroup訊息 ()

使用newTextGameMessage或newBinaryGameMessage從即時伺服器建立的訊息傳送給使用 TCP 通道的玩家群組中的所有玩家。群組 ID 必須為正整數或「-1」以表示所有群組。如實時服務器腳本示例需使用者定義群組 ID 的範例，請參閱。

語法

```
rtSession.sendReliableGroupMessage(gameMessage, targetGroup)
```

newTextGame訊息 ()

創建一個包含文本的新消息，從服務器發送到使用該SendMessage功能的播放器接收者。訊息的格式與 Realtime 用戶端開發套件中所使用的格式相似 (請參閱 RTMessage)。其會傳回 gameMessage 物件。

語法

```
rtSession.newTextGameMessage(opcode, sender, payload)
```

newBinaryGame訊息 ()

創建一個包含二進制數據的新消息，使用這些SendMessage函數從服務器發送給播放器接收者。訊息的格式與 Realtime 用戶端開發套件中所使用的格式相似 (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

當遊戲用戶端連線至即時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。

播放器。 playerSessionId

當遊戲用戶端連線至即時伺服器並加入遊戲工作階段時，所參考的玩家工作階段 ID。

遊戲訊息物件

使用這些方法來訪問實時服務器接收的消息。從遊戲用戶端接收到的訊息具有 RTMessage 結構。

getPayloadAs文字 ()

獲取遊戲消息有效負載作為文本。

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。

配置對象

配置對象可用於覆蓋默認配置。

配置. 最大播放器

可接受的用戶端/伺服器連線數目上限RealTimeServers。

預設值為 32。

配置。 pingIntervalTime

伺服器嘗試傳送 ping 到所有連線的用戶端，以確認連線狀況良好的時間間隔 (以毫秒為單位)。

預設值為 3000 毫秒。

亞馬遜GameLift服務器 SDK 參考

本節包含 Amazon GameLift 伺服器開發套件的參考文件。使用伺服器 SDK 整合您的自訂遊戲伺服器，以便與 Amazon GameLift 服務進行通訊。

主題

- C++ 的亞馬遜GameLift伺服器 SDK 參考
- C# 的亞馬遜GameLift伺服器 SDK 參考
- 適用於圍棋的亞馬遜GameLift服務器 SDK
- 虛幻引擎的亞馬遜GameLift服務器 SDK 參考

C++ 的亞馬遜GameLift伺服器 SDK 參考

您可以使用此 Amazon GameLift C++ 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

主題

- 亞馬遜GameLift服務器 SDK 5.x 參考 C ++
- 亞馬遜 GameLift C ++ 服務器 SDK 3.x 參考

亞馬遜GameLift服務器 SDK 5.x 參考 C ++

這個 Amazon GameLift C++ 伺服器 SDK 5.x 參考可以幫助您準備好與亞馬遜GameLift搭配使用的多人遊戲。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

> **Note**
>
> 本主題說明使用 GameLift C++ 標準程式庫 (std) 建置時可以使用的 Amazon C++ API。具體而言，本文件適用於您使用-DGAMELIFT_USE_STD=1選項編譯的程式碼。

主題

- Amazon GameLift 服務器開發套件（C ++）5.x 參考：操作
- Amazon GameLift 服務器開發套件（C ++）參考：數據類型

Amazon GameLift 服務器開發套件（C ++）5.x 參考：操作

您可以使用此 Amazon GameLift C++ 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

> **Note**
>
> 本主題說明使用 GameLift C++ 標準程式庫 (std) 建置時可以使用的 Amazon C++ API。具體而言，本文件適用於您使用-DGAMELIFT_USE_STD=1選項編譯的程式碼。

動作

- GetSdkVersion()

- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessReadyAsync()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- 摧毀 ( )

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號碼 (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()

為受管 EC2 叢集初始化 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法會從主機環境讀取伺服器參數，以設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

傳回值

返回一個the section called “首頁成果”對象，指示服務器進程是否準備好調用ProcessReady()。

範例

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
  Aws::GameLift::Server::InitSDK();
```

InitSDK()

初始化Anywhere叢集的 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法需要明確的伺服器參數來設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

參數

ServerParameters

若要初始化 Amazon GameLift Anywhere 叢集上的遊戲伺服器，請使用下列資訊建構`ServerParameters`物件：

- WebSocket 用來連線到遊戲伺服器的 URL。
- 用來託管遊戲伺服器的程序 ID。

- 主控遊戲伺服器處理程序的運算 ID。
- 包含您的 Amazon GameLift Anywhere 運算的 Amazon GameLift 車隊的 ID。
- Amazon GameLift 操作生成的授權令牌。

傳回值

返回一個the section called “首頁成果”對象，指示服務器進程是否準備好調用ProcessReady()。

> **Note**
>
> 如果部署到 InitSDK() Anywhere 叢集的遊戲組建時呼叫失敗，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使用中的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新組建並將其部署到新的叢集。

範例

Amazon GameLift Anywhere 示例

```
//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 GameLift agent to enable
 further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
 Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

通 GameLift 知 Amazon 伺服器程序已準備好主持遊戲工作階段。調用後調用InitSDK()此方法。每個進程應該只調用一次此方法。

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：

- Amazon GameLift 服務呼叫以與伺服器處理序通訊之遊戲伺服器程式碼中實作的回呼方法名稱。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案 GameLift 的路徑。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務伺服器處理序已準備好主持遊戲工作階段。在伺服器進程準備好主持遊戲工作階段之後，應該呼叫此方法。這些參數指定 Amazon GameLift 在某些情況下調用的回調函數名稱。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處理序。

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：

- Amazon GameLift 服務呼叫以與伺服器處理序通訊之遊戲伺服器程式碼中實作的回呼方法名稱。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案 GameLift 的路徑。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通 GameLift 知 Amazon 服務器進程正在終止。在所有其他清理任務（包括關閉活動的遊戲會話）之後以及終止進程之前調用此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束，並產生叢集事件。如果程序因錯誤而終止，則產生的叢集事件為SERVER_PROCESS_TERMINATED_UNHEALTHY。

語法

```
Aws::GameLift::GenericOutcome processEndingOutcome =
 Aws::GameLift::Server::ProcessEnding();
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會Destroy()在使用成功或錯誤結束代碼來終止伺服器處理序之前呼叫ProcessEnding()和。

```
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()

通 GameLift 知 Amazon 伺服器程序已啟動遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲會話初始化之後，應將此操作作為onStartGameSession()回調函數的一部分調用。

語法

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =
 Aws::GameLift::Server::ActivateGameSession();
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

這個例子顯示了作ActivateGameSession()為onStartGameSession()委託函數的一部分調用。

```
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

playerCreationSession政策

類型：PlayerSessionCreationPolicy枚舉值。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

擷取使用中伺服器處理序主控的遊戲工作階段 ID。

對於未使用遊戲工作階段啟動的閒置進程，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。

對於未使用遊戲會話激活的空閒進程，調用返回 `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。服務器進程在收到來自 Amazon 的`onProcessTerminate()`回調後採取行動 GameLift。Amazon GameLift 呼叫`onProcessTerminate()`的原因如下：

- 當服務器進程報告健康狀況不佳或沒有響應 Amazon 時 GameLift。
- 在縮小事件期間終止執行個體時。
- 執行個體因為現場執行個體中斷而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，則返回終止時間作為AwsDateTimeOutcome對象。該值是終止時間，以之後經過的刻度表示。0001 00:00:00例如，日期時間值等2020-09-13 12:26:40 -000Z於637355968000000000刻度。如果沒有可用的終止時間，則返回錯誤消息。

如果程序尚未收到 ProcessParameters. OnProcessTerminate() 回調，返回錯誤消息。如需關閉伺服器處理序的詳細資訊，請參閱回應伺服器處理序關閉通知。

範例

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
 Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

通知 Amazon 具 GameLift 有指定玩家工作階段 ID 的玩家已連線到伺服器程序並需要驗證。Amazon 會 GameLift 驗證玩家工作階段 ID 是否有效。玩家會話驗證後，Amazon 將播放器插槽的狀態從保留 GameLift 更改為活動狀態。

語法

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會處理包含驗證和拒絕非有效玩家工作階段 ID 的連線要求。

```
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()

通 GameLift 知 Amazon 播放器已與服務器進程斷開連接。作為回應，Amazon 將播放器插槽 GameLift 更改為可用。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

```
Aws::GameLift::GenericOutcome disconnectOutcome =
 Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

檢索播放器會話數據，其中包括設置，會話元數據和播放器數據。使用此方法取得下列項目的相關資訊：

- 單一玩家工作階段
- 遊戲階段中的所有玩家工作階段
- 與單一玩家 ID 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

此範例要求所有主動連線至指定遊戲工作階段的玩家工作階段。透過省略限制值NextToken並將限制值設定為 10，Amazon 會 GameLift 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。如果配對新玩家，Amazon 會使用回調函`OnUpdateGameSession()`數 GameLift 提供更新的分房系統資料。

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

傳達下列資訊的 StartMatchBackfillRequest 物件：

- 指派給回填請求的票證 ID。此信息是可選的; 如果沒有提供 ID，Amazon GameLift 將生成一個。
- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值會顯示在遊戲工作階段的分房系統資料中。
- 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

傳回具有相符回填工單 ID 的the section called “StartMatchBackfillOutcome”物件，或失敗並顯示錯誤訊息。

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()

取消使用中的比對回填要求。如需詳細資訊，請參閱FlexMatch回填功能。

語法

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

參數

StopMatchBackfillRequest

識別要取消配對票證的 StopMatchBackfillRequest 物件：

- 指派給回填請求的工單 ID。

- 回填要求傳送至的分房系統。
- 與回填要求相關聯的遊戲工作階段。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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 TLS 憑證的路徑，用於加密 Amazon GameLift Anywhere 運算資源和 Amazon 之間的網路連線 GameLift。將運算裝置註冊到 Amazon GameLift Anywhere 叢集時，可以使用憑證路徑。若要取得更多資訊，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 IAM 角色登入資料， GameLift 以授權 Amazon 與其他人互動AWS 服務。如需詳細資訊，請參閱與您車隊的其他AWS資源進行溝通。

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);
```

此範例顯示如何使用選擇性RoleSessionName值將名稱指派給認證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]-[## 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);
```

摧毀（ ）

從記憶體中釋放 Amazon GameLift 遊戲伺服器 SDK。最佳做法是在終止程序之後`ProcessEnding()`和之前呼叫此方法。如果您使用的是 Anywhere 叢集，且未在每個遊戲工作階段後終止伺服器程序，請先呼叫`Destroy()`然`InitSDK()`後重新初始化，再通知 Amazon GameLift 程序已準備好主持遊戲工作階段。`ProcessReady()`

語法

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

參數

沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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);
}
```

Amazon GameLift 服務器開發套件（C ++）參考：數據類型

您可以使用此 Amazon GameLift C++ 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

> **Note**
>
> 本主題說明使用 GameLift C++ 標準程式庫 (`std`) 建置時可以使用的 Amazon C++ API。具體而言，本文件適用於您使用`-DGAMELIFT_USE_STD=1`選項編譯的程式碼。

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
- 首頁成果

- GameLiftError
- 列舉

LogParameters

識別在遊戲工作階段期間產生的物件，您希望 Amazon GameLift 在遊戲工作階段結束後上傳和存放的檔案。遊戲服務器提供`LogParameters`給 Amazon GameLift 作為ProcessReady()調用`ProcessParameters`對象的一部分。

屬性	Description
LogPaths	您希望 Amazon GameLift 存放以供 future 來存取之遊戲伺服器日誌檔的目錄路徑清單。服務器進程在每個遊戲會話期間生成這些文件。您可以在遊戲伺服器中定義檔案路徑和名稱，並將它們儲存在根遊戲建置目錄中。 記錄檔路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段記錄儲存在類似路徑中`MyGame\sessionLogs\`，則該路徑將位於 Windows 執行個體`c:\game\MyGame\sessionLogs` 上。 Type (類型)：`std:vector<std::string>` 必要：否

ProcessParameters

此數據類型包含一組發送到 Amazon GameLift 的參數ProcessReady()。

屬性	Description
LogParameters	具有遊戲階段作業期間所產生之檔案目錄路徑的物件。Amazon GameLift 複製和存儲文件以供 future 訪問。

	Type (類型)：`Aws::GameLift::Server:: LogParameters` 必要：否
OnHealthCheck	Amazon 呼 GameLift 叫以從伺服器處理序要求健康狀態報告的回呼函數。Amazon 每 60 秒 GameLift 調用一次此函數，並等待 60 秒進行響應。如果狀況不良，TRUE則伺服器處理序FALSE會傳回狀況良好。如果未傳回任何回應，Amazon 會將伺服器處理序 GameLift 記錄為狀況不良。 Type (類型)：`std::function<bool()> onHealthCheck` 必要：否
OnProcessTerminate	Amazon GameLift 調用以強制服務器進程關閉的回調函數。呼叫此函數之後，Amazon 會 GameLift 等待 5 分鐘讓伺服器處理序關閉並回應呼ProcessEnding()叫，然後再關閉伺服器處理序。 Type (類型)：`std::function<void()> onProcessTerminate` 必要：是
OnRefreshConnection	Amazon 調 GameLift 用以刷新與遊戲服務器的連接的回調函數的名稱。 Type (類型)：`void OnRefreshConnectionDelegate()` 必要：是

OnStartGameSession	Amazon GameLift 調用以激活新遊戲會話的回調函數。Amazon GameLift 調用此函數以響應客戶請求CreateGameSession。回呼函數會傳遞GameSession物件，如 Amazon GameLift API 參考中所定義。 Type (類型)：`const std::function<void (Aws::GameLift::Model::GameSession)> onStartGameSession` 必要：是
OnUpdateGameSession	Amazon GameLift 調用將更新的遊戲會話對象傳遞給服務器進程的回調函數。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時 GameLift 呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason` ）和匹配回填票證 ID。 Type (類型)：`std::function<void (Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession` 必要：否
連線埠	伺服器處理序偵聽新播放程式連線的連接埠號碼。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

此資料類型會更新為遊戲工作階段物件，其中包括更新遊戲工作階段的原因，以及如果使用回填填補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	由 Amazon GameLift API 定義的GameSession對象。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
UpdateReason	遊戲工作階段正在更新的原因。 Type (類型)：`Aws::GameLift::Server::UpdateReason` 必要：是
BackfillTicketId	嘗試更新遊戲工作階段的回填票證 ID。 Type (類型)：`std::string` 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碼。遊戲工作階段 ARN 的格式如下：`arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 。

屬性	Description
	Type (類型)：`std::string` 必要：否
名稱	遊戲工作階段的描述性標籤。 Type (類型)：`std::string` 必要：否
FleetId	執行遊戲工作階段之叢集的唯一識別碼。 Type (類型)：`std::string` 必要：否
MaximumPlayerSessionCount	與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 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 Type (類型)：`in` 必要：否
IpAddress	遊戲工作階段的 IP 位址。若要連線到 Amazon GameLift 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 Type (類型)：`std::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)：`std::string` 必要：否
MatchmakerData	有關用於建立遊戲工作階段的配對程序的資訊，以 JSON 語法格式化為字串。除了使用的配對配置外，它還包含了所有指定給比賽的玩家的數據，包括球員屬性和團隊分配。 Type (類型)：`std::string` 必要：否
GameProperties	遊戲工作階段的一組自訂屬性，格式化為 key:value 配對。這些屬性會與啟動新遊戲工作階段的要求一起傳遞。 Type (類型)：`std :: vector < 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 識別碼。值的格式如下： • 支援 TLS 的叢集：`.<unique identifier>.<region identifier>.amazongamelift.com` • 未啟用 TLS 的叢集：`。ec2-<unique identifier>.compute.amazonaws.com` 連線至已啟用 TLS 的叢集上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 位址。 Type (類型)：`std::string` 必要：否

ServerParameters

用於維護 Amazon GameLift Anywhere 叢集上遊戲伺服器與 Amazon GameLift 服務之間連線的資訊。當使用啟動新的伺服器處理序時，會使用此資訊InitSDK()。對於託管在 Amazon GameLift 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
webSocketUrl	`GameLiftServerSdkEndpoint` Amazon GameLift 返回當你一`RegisterCompute` 個 Amazon GameLift Anywhere 計算資源。 Type (類型)：`std::string` 必要：是
流程	註冊到託管您遊戲的伺服器處理序的唯一識別碼。

屬性	Description
	Type (類型)：`std::string` 必要：是
hostId	`HostID`這是您註冊運算時`ComputeName` 使用的。如需詳細資訊，請參閱，RegisterCompute。 Type (類型)：`std::string` 必要：是
飛行	計算所註冊之叢集的唯一識別碼。如需詳細資訊，請參閱，RegisterCompute。 Type (類型)：`std::string` 必要：是
驗證令牌	Amazon 生成的身份驗證令牌 GameLift ，用於向 Amazon 驗證您的服務器。 GameLift如需詳細資訊，請參閱，GetComputeAuthToken。 Type (類型)：`std::string` 必要：是

StartMatchBackfillRequest

用於建立配對回填請求的資訊。遊戲服務器通過電話將此信息傳達StartMatchBackfill()給 Amazon GameLift 。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別碼。該 API 操作`GetGameSessionId` 返回 ARN 格式的標識符。

屬性	Description
	Type (類型)：`std::string` 必要：是
MatchmakingConfigurationArn	一個唯一的識別碼，以 ARN 的形式，供分房系統用於此要求。原始遊戲工作階段的分房系統 ARN 位於分房系統資料屬性中的遊戲工作階段物件中。請參閱使用分房系統資料，進一步了解分房系統資料。 Type (類型)：`std::string` 必要：是
Players	一組數據，代表在遊戲會話中的所有玩家。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碼。如果你不提供一個值，Amazon GameLift 生成一個。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。開始配對要求時，玩家會擁有玩家 ID、屬性以及可能的延遲資料。Amazon 在比賽進行後 GameLift 添加球隊信息。

屬性	Description
LatencyIn女士	以毫秒為單位表示的一組值，表示玩家連線至某個位置時所經歷的延遲量。 如果使用此屬性，則播放器僅匹配列出的位置。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	包含可用於配對的玩家資訊的 key: 值組合集合。玩家屬性鍵必須與配對規則集中 PlayerAttributes 使用的鍵相符。 如需播放程式屬性的詳細資訊，請參閱Attribute Value。 Type (類型)：`std::map<std::string,AttributeValue>` 必要：否
PlayerId	玩家的唯一識別碼。 Type (類型)：`std::string` 必要：否
團隊	球員在比賽中被分配到的球隊名稱。您可以在配對規則集中定義小組名稱。 Type (類型)：`std::string` 必要：否

DescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。服務器進程通過DescribePlayerSessions()致電 Amazon 提供此信息 GameLift。

屬性	Description
GameSessionId	唯一的遊戲工作階段識別碼。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`arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>` 。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碼。使用此參數可要求單一特定玩家工作階段。 Type (類型)：`std::string` 必要：否
PlayerId	玩家的唯一識別碼。使用此參數可要求特定玩家的所有玩家工作階段。請參閱 產生玩家 ID。 Type (類型)：`std::string` 必要：否
PlayerSessionStatusFilter	用於篩選結果的玩家工作階段狀態。可能的玩家會話狀態包括： • 保留 — 已收到播放程式工作階段要求，但播放程式尚未連線至伺服器處理序或已驗證。 • 作用中 — 播放程式已由伺服器處理序驗證並已連線。

屬性	Description
	• 已完成 — 玩家連線中斷。 • TIMEDOUT — 收到玩家工作階段要求，但玩家沒有連線或未在逾時限制 (60 秒) 內驗證。 Type (類型)：`std::string` 必要：否
NextToken	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。如果您提供玩家工作階段 ID，則會忽略此參數。 Type (類型)：`std::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

用於取消配對回填要求的資訊。遊戲服務器在通話中將此信息傳達StopMatchBackfill()給 Amazon GameLift 服務。

屬性	Description
GameSessionArn	要求取消的唯一遊戲工作階段識別碼。 Type (類型)：`char[]` 必要：否
MatchmakingConfigurationArn	傳送此要求的分房系統的唯一識別碼。

屬性	Description
	Type (類型)：`char[]` 必要：否
TicketId	要取消之回填要求票證的唯一識別碼。 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、字串陣列或資料對應。每個`AttributeValue`物件都必須正好使用其中一個可用屬性：`SNSL`、、或`SDM`。

屬性	Description
AttrType	指定屬性值的類型。可能的屬性值類型包括： • NONE • 字符串 • 雙 • 字符串列表 • 弦雙地圖 必要：否
S	表示字串屬性值。 Type (類型)：`std::string` 必要：否
N	表示數值屬性值。 Type (類型)：`double`

屬性	Description
	必要：否
SL	表示字符串屬性值的數組。 Type (類型)：`std::vector<std::string>` 必要：否
代決人	代表字串索引鍵和 double 值的字典。 Type (類型)：`std::map<std::string, double>` 必要：否

GetFleetRoleCredentialsRequest

這種數據類型使遊戲服務器對您的其他AWS資源的訪問有限。如需詳細資訊，請參閱，為 Amazon 設置 IAM 服務角色 GameLift。

屬性	Description
RoleArn	服務角色的 Amazon 資源名稱 (ARN)，可延伸對AWS資源的有限存取權限。 Type (類型)：`std::string` 必要：否
RoleSessionName	可用來唯一識別工作階段的角色AWS Security Token ServiceAssumeRole工作階段名稱。此名稱會公開在稽核記錄中，例如中的名稱 CloudTrail。 Type (類型)：`std::string` 必要：否

AwsLongOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：`long` 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`long&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

AwsStringOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：`std::string`

屬性	Description
	必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`long&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

DescribePlayerSessionsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “DescribePlayerSessionsResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。

屬性	Description
	Type (類型)：`Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

DescribePlayerSessionsResult

物件集合，其中包含符合要求之每個播放程式工作階段的屬性。

屬性	Description
NextToken	表示下一個連續結果頁面開始的權杖。使用先前呼叫此作業時傳回的 Token。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)：`std::string` 必要：是
PlayerSessions	Type (類型)：`IList<`the section called “PlayerSession” `>` 必要：

屬性	Description
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`std::string&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GenericOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GenericOutcomeCallable

這種數據類型是異步泛型結果。它具有下列屬性：

屬性	Description
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

PlayerSession

此資料類型代表 Amazon GameLift 傳遞給遊戲伺服器的玩家工作階段。如需詳細資訊，請參閱PlayerSession。

屬性	Description
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

屬性	Description
PlayerData	Type (類型)：`std::string` 必要：否
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。如果提 PlayerId 供了 PlayerSessionId 或，則對響應沒 PlayerSessionStatusFilter 有影響。 類型：`PlayerSessionStatus` 枚舉。可能的值包括以下： • ACTIVE • COMPLETED (已完成) • 沒有設定 • 已保留 • 定時 必要：否
TerminationTime	Type (類型)：`long` 必要：否

屬性	Description
DnsName	Type (類型)：`std::string` 必要：否

StartMatchBackfillOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “StartMatchBackfillResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`StartMatchBackfillResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

StartMatchBackfillResult

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
TicketId	配對票證的唯一識別碼。如果此處未指定任何票證 ID，Amazon GameLift 將以 UUID 的形式生成一個。使用此識別碼追蹤比賽回填工單狀態並擷取比賽結果。 Type (類型)：`std::string` 必要：否

GetComputeCertificateOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “GetComputeCertificateResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`Aws::GameLift::Server::Model::GetComputeCertificateResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool`

屬性	Description
	必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GetComputeCertificateResult

計算上 TLS 憑證的路徑以及計算機的主機名稱。

屬性	Description
CertificatePath	計算資源上 TLS 憑證的路徑。使用 Amazon 受 GameLift 管叢集時，此路徑包含： • `certificate.pem` ：一般使用者憑證。完整憑證鏈結是`certificateChain.pem` 附加至此憑證的組合。 • `certificateChain.pem` ：包含根憑證和中繼憑證的憑證鏈結。 • `rootCertificate.pem` ：根憑證。 • `privateKey.pem` ：一般使用者憑證的私密金鑰。 Type (類型)：`std::string` 必要：否
ComputeName	計算資源的名稱。 Type (類型)：`std::string` 必要：否

GetFleetRoleCredentialsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “GetFleetRoleCredentialsResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GetFleetRoleCredentialsResult

屬性	Description
AccessKeyId	用於驗證並提供對AWS資源的存取權限的存取金鑰 ID。

屬性	Description
	Type (類型)：`string` 必要：否
AssumedRoleId	服務角色所屬的使用者識別碼。 Type (類型)：`string` 必要：否
AssumedRoleUserArn	服務角色所屬之使用者的 Amazon 資源名稱 (ARN)。 Type (類型)：`string` 必要：否
過期	您的工作階段認證到期之前的時間長度。 Type (類型)：`DateTime` 必要：否
SecretAccessKey	用於驗證的秘密存取金鑰 ID。 Type (類型)：`string` 必要：否
SessionToken	標識當前活動會話與您的AWS資源進行交互的令牌。 Type (類型)：`string` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是

屬性	Description
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

首頁成果

> **Note**
>
> `InitSDKOutcome`只有當您使用`std`標誌構建 SDK 時才會返回。如果您使用標`nostd`誌構建，the section called “GenericOutcome”則返回。

屬性	Description
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GameLiftError

屬性	Description
ErrorType	錯誤類型。 類型：**`GameLiftErrorType`** 枚舉。

屬性	Description
	必要：否
ErrorName	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

針對 Amazon GameLift 伺服器開發套件 (C++) 定義的枚舉定義如下：

GameLiftErrorType

指示錯誤類型的字串值。有效值包含：

- 不要求 _ 例外
- 遊戲設定 — 尚未設定遊戲工作階段識別碼。
- 內部 _ 服務 _ 異常
- 本地連接失敗 — 與 Amazon 的本地連接失敗。 GameLift
- 網路不初始化 — 網路尚未初始化。
- 服務呼叫失敗 — 對服務的呼叫失敗。AWS
- WEB 插槽 _ 連接 _ 失敗
- WEB 插座 _ 連接 _ 失敗 _ 禁止
- 網站插槽 _ 連接 _ 失敗 _ 無效 _URL
- 網路插槽 _ 連線 _ 失敗 _ 逾時
- 已初始化 — Amazon GameLift 伺服器或用戶端已經使用初始化 () 進行初始化。
- FLEET_MISMATCH — 目標叢集與遊戲經營或玩家工作階段的機隊不相符。
- 已初始化 — Amazon 用戶端尚未初始化。 GameLift

- 已初始化 — Amazon 伺服器尚未初始化。 GameLift
- 遊戲工作階段失敗 — Amazon GameLift 伺服器開發套件無法聯絡服務以報告遊戲工作階段已結束。
- 未啟動 Amazon 伺服器遊戲工作階段。 GameLift
- 遊戲工作階段已準備就緒 — Amazon GameLift 伺服器開發套件無法聯絡服務以報告遊戲工作階段已準備就緒。
- 初始化不匹配-在服務器:: 初始化 ( ) 之後調用客戶端方法，反之亦然。
- 初始化 — Amazon GameLift 伺服器或用戶端尚未使用初始化 () 初始化。
- 無目標別名-尚未設定目標別名。
- 目標叢集 — 尚未設定目標叢集。
- 處理結束-Amazon GameLift 伺服器開發套件無法聯絡服務以報告程序已結束。
- PROCES_NOT_ACT_ACTIVE — 伺服器處理作業尚未啟動、未繫結至 GameSession，且無法接受或處理。 PlayerSessions
- 處理程序 NOT_READY — 伺服器處理作業尚未準備好啟動。
- 處理程序 READY_FAIN — Amazon GameLift 伺服器開發套件無法聯絡服務以報告程序已準備就緒。
- SDK_ 版本偵測失敗 — SDK 版本偵測失敗。
- STX_CALL_ 失敗 — 對 XSTX 伺服器後端元件的呼叫失敗。
- STX_ 初始化失敗 — XSTX 伺服器後端元件無法初始化。
- 意外的玩家工作階段 — 伺服器遇到未註冊的播放程式工作階段。
- WEB 插槽 _ 連接 _ 失敗
- WEB 插座 _ 連接 _ 失敗 _ 禁止
- 網站插槽 _ 連接 _ 失敗 _ 無效 _URL
- 網路插槽 _ 連線 _ 失敗 _ 逾時
- 網站重新擷取 _ 傳送訊息 _ 失敗 — 將訊息傳送至服務的可擷取失敗。 GameLift WebSocket
- 網站傳送訊息失敗 — 無法將訊息傳送至服務。 GameLift WebSocket
- 匹配 _ 後填 _ 請求驗證 — 驗證請求失敗。
- 播放器會話請求驗證 — 請求的驗證失敗。

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玩家工作階段。

亞馬遜 GameLift C ++ 服務器 SDK 3.x 參考

您可以使用這個亞馬遜 GameLift C ++ 服務器 SDK 3.x 參考來幫助您準備與亞馬遜GameLift一起使用的多人遊戲。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

主題

- 亞馬遜GameLift服務器 SDK（C ++）參考：操作
- 亞馬遜GameLift服務器開發套件（C ++）參考：數據類型

亞馬遜GameLift服務器 SDK（C ++）參考：操作

您可以使用此 Amazon GameLift C++ 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

動作

- AcceptPlayerSession()
- ActivateGameSession()
- DescribePlayerSessions()
- GetGameSessionId()
- GetInstanceCertificate()
- GetSdkVersion()
- GetTerminationTime()
- InitSDK()
- ProcessEnding()
- ProcessReady()
- ProcessReadyAsync()
- RemovePlayerSession()
- StartMatchBackfill()
- StopMatchBackfill()

- TerminateGameSession()
- UpdatePlayerSessionCreationPolicy()
- 摧毀（ ）

AcceptPlayerSession()

通知 Amazon GameLift 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並需要驗證。Amazon 會GameLift驗證玩家工作階段 ID 是否有效 — 也就是說，玩家 ID 已在遊戲工作階段中保留一個玩家位置。一旦驗證，亞馬遜GameLift將播放器插槽的狀態從保留更改為活動狀態。

語法

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

參數

playerSessionId

由亞馬遜GameLift服務發出的唯一 ID，以回應對 AWS SDK 亞馬遜 GameLift API 動作的呼叫CreatePlayerSession。遊戲客戶端在連接到服務器進程時引用此 ID。

類型:的標準:: 字符串

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務伺服器處理序已啟動遊戲工作階段，現在已準備好接收玩家連線。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

傳回包含錯誤訊息的成功或失敗的一般結果。

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

describePlayerSessions請求

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。播放器工作階段物件的結構與 AWS SDK Amazon GameLift 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。藉由省略NextToken並將Limit值設定為 10，Amazon 會GameLift傳回符合請求的前 10 個播放器工作階段記錄。

```
// 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>。

對於尚未通過遊戲會話激活的空閒進程，調用返回 Success = True 和 GameSessionId =""（空字符串）。

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()

擷取與叢集及其執行個體相關聯的 PEM 編碼 TLS 憑證的檔案位置。 AWS Certificate Manager當您建立新的叢集並將憑證組態設定為「已產生」時，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，會傳回包含叢集 TLS 憑證檔案和憑證鏈結位置的GetInstanceCertificateOutcome物件，這些檔案儲存在執行個體上。從憑證鏈結擷取的根憑證檔案也會儲存在執行個體上。如果不成功，則會傳回錯誤訊息。

如需有關憑證和憑證鏈結資料的詳細資訊，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。返回的字符串僅包含版本號（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。伺服器處理序會在收到來自 Amazon GameLift 服務的onProcessTerminate()回呼後採取此動作。Amazon GameLift 可能會因onProcessTerminate()為下列原因而致電：(1) 伺服器處理序報告運作狀態不佳或未回應 Amazon 時GameLift、(2) 在縮減事件期間終止執行個體時，或 (3) 執行個體因 Spot 中斷而終止時。

如果進程已收到onProcessTerminate()回調，則返回的值是估計的終止時間。如果處理序未收到onProcessTerminate()回呼，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，則返回終止時間作為AwsLongOutcome對象。該值是終止時間，以自 0001 00:00:00 以來經過的刻度表示。例如，日期時間值 2020-9 月 13 日 12:26：40 -000Z 等於刻度。如果沒有可用的終止時間，則返回錯誤消息。

範例

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

初始化亞馬遜開GameLift發套件。在發生任何其他 Amazon GameLift 相關初始化之前，應在啟動時呼叫此方法。

語法

```
InitSDKOutcome InitSDK();
```

參數

此動作沒有參數。

傳回值

如果成功，則返回一個InitSdkOutcome對象，指示服務器進程已準備好調用ProcessReady()。

範例

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

通知 Amazon GameLift 服務伺服器處理序正在關閉。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。

一旦該方法退出代碼為 0，您可以使用成功的退出代碼終止該進程。您也可以使用錯誤碼結束程序。如果您以錯誤碼結束，叢集事件將指示處理序異常終止 (SERVER_PROCESS_TERMINATED_UNHEALTHY)。

語法

```
GenericOutcome ProcessEnding();
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務伺服器處理序已準備好主持遊戲工作階段。在成功叫用InitSDK()並完成伺服器處理序主控遊戲工作階段之前所需的設定工作之後，呼叫此方法。每個進程應該只調用一次此方法。

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處理序。

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，Amazon GameLift 服務呼叫以與伺服器處理序進行通訊。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案GameLift的路徑。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務伺服器處理序已準備好主持遊戲工作階段。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。這些參數指定亞馬遜GameLift在某些情況下調用的回調函數的名稱。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處理序。

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 服務呼叫以與伺服器處理序進行通訊。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案GameLift的路徑。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務具有指定玩家工作階段 ID 的玩家已中斷與伺服器處理序的連線。作為回應，亞馬遜將播放器插槽GameLift更改為可用，這允許將其分配給新玩家。

語法

```
GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);
```

參數

playerSessionId

由亞馬遜GameLift服務發出的唯一 ID，以回應對 AWS SDK 亞馬遜 GameLift API 動作的呼叫CreatePlayerSession。遊戲客戶端在連接到服務器進程時引用此 ID。

類型:的標準:: 字符串

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。進一步瞭解FlexMatch回填功能。

此為非同步動作。如果成功配對新玩家，Amazon GameLift 服務會透過叫用回呼函數來提供更新的分房系統資料。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。此資訊是選擇性的；如果未提供 ID，Amazon GameLift 將自動產生一個 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

返回匹配回填票證或失敗的StartMatchBackfillOutcome對象，並顯示錯誤消息。您可以使用 AWS SDK 動作 DescribeMatchmaking() 來追蹤工單狀態。

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()。進一步瞭解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

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

此方法已被版本 4.0.1 棄用。相反，在遊戲會話結束ProcessEnding()後，服務器進程應該調用。

通知 Amazon GameLift 服務伺服器處理序已結束目前的遊戲工作階段。當伺服器處理程序保持作用中並準備好主持新遊戲工作階段時，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 Amazon 發出訊號GameLift，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關閉伺服器程序，則不會呼叫此動作。相反，調用ProcessEnding()以表示遊戲會話和服務器進程都結束。

語法

```
GenericOutcome TerminateGameSession();
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。另請參閱 AWS SDK 動作 UpdateGameSession()。

語法

```
GenericOutcome UpdatePlayerSessionCreationPolicy(
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

參數

newPlayerSession政策

字串值代表遊戲工作階段是否可接受新玩家。

類型：Aws::GameLift: 模型:: PlayerSessionCreationPolicy 枚舉。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

傳回包含錯誤訊息的成功或失敗的一般結果。

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
```

摧毀 ( )

在遊戲伺服器初始化期間清除 InitSDK () 配置的記憶體。結束遊戲伺服器程序後，請使用此方法，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

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會在遊戲伺服器處理序結束後清除 InitSDK 配置的記憶體。

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {
  Aws::GameLift::Server::Destroy();
  exit(0);
}
```

亞馬遜GameLift服務器開發套件（C ++）參考：數據類型

您可以使用此 Amazon GameLift C++ 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

此 API 定義於 `GameLiftServerAPI.h`、`LogParameters.h`、`ProcessParameters.h`。

- 動作
- 資料類型

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：

- 提供一個PlayerSessionId要求特定玩家工作階段。
- 提供一個GameSessionId要求指定遊戲工作階段中的所有玩家工作階段。
- 提供一個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。搭配使用此參數可NextToken取得一組連續頁面的結果。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 ID。

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在遊戲工作階段期間產生的檔案，您希望 Amazon GameLift 在遊戲工作階段結束後上傳和存放這些檔案。此信息在通ProcessReady()話中傳達給亞馬遜GameLift服務。

目錄

logPaths

您希望 Amazon GameLift 存放以供將來存取的遊戲伺服器記錄檔的目錄路徑。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。記錄檔路徑必須是絕對的。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

此資料類型包含在ProcessReady()呼叫中傳送至 Amazon GameLift 服務的一組參數。

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: 服務器:: LogParameters

必要：否

onStartGame階段

Amazon GameLift 服務呼叫以啟動新遊戲工作階段的回呼函數名稱。亞馬遜GameLift調用此函數以響應客戶端請求CreateGameSession。回調函數傳遞一個GameSession對象（在亞馬遜GameLift服務 API 參考中定義）。

類型：`const std::function<void(Aws::GameLift::Model::GameSession)> onStartGameSession`

必要：是

onProcessTerminate

Amazon GameLift 服務呼叫以強制伺服器處理序關閉的回呼函數名稱。調用此函數後，亞馬遜GameLift等待五分鐘，以關閉服務器進程並ProcessEnding()通過調用進行響應。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 服務呼叫以從伺服器處理序要求健康狀態報告的回呼函數名稱。亞馬遜每 60 秒GameLift調用一次此函數。調用此函數後，亞馬遜GameLift等待 60 秒的響應，如果沒有收到任何響應。將服務器進程記錄為不健康。

類型：`std::function<bool()> onHealthCheck`

必要：否

onUpdateGame階段

Amazon GameLift 服務呼叫以將更新的遊戲工作階段物件傳遞至伺服器處理序的回呼函數名稱。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時GameLift呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason`）和匹配回填票證 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。該信息在通StartMatchBackfill()話中傳達給亞馬遜GameLift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。透過 Word 中的分房系統資料進一步了解分房系統資料。

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。有關播放器對象格式的說明，請參閱亞馬遜 GameLift 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:標準:向量https://docs.aws.amazon.com/gamelift/latest/apireference/API_Player.html <player>

必要：是

TicketId

配對或配對回填要求票證的唯一識別項。如果這裡沒有提供任何值，亞馬遜GameLift將以 UUID 的形式生成一個值。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。該信息在通StopMatchBackfill()話中傳達給亞馬遜GameLift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

C# 的亞馬遜GameLift伺服器 SDK 參考

您可以使用這個 Amazon GameLift C# 伺服器 SDK 參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

主題

- 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考
- 亞馬遜GameLift服務器 SDK 4.x 參考 C＃

用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考

您可以使用此 Amazon GameLift C# 伺服器 SDK 5.x 參考來協助您準備好與 Amazon GameLift 搭配使用的多人遊戲。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器和如需使用適用於 Unity 的 C# 伺服器 SDK 外掛程式的資訊，請參閱 GameLift 將亞馬遜整合到統一項目中。適用於 C＃ 的亞馬遜GameLift服務器 SDK 5.x 支持 .NET 4.6 和 .NET 6。

主題

- C＃ 和統一的 Amazon GameLift 服務器 SDK 參考：操作
- 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 參考：數據類型

C＃ 和統一的 Amazon GameLift 服務器 SDK 參考：操作

這個 Amazon GameLift C# 伺服器開發套件參考可協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器和如需使用適用於 Unity 的 C# 伺服器 SDK 外掛程式的資訊，請參閱 GameLift 將亞馬遜整合到統一項目中。

動作

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()

- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- 摧毀 ( )

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()

為受管 EC2 叢集初始化 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法會從主機環境讀取伺服器參數，以設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
GenericOutcome InitSDK();
```

傳回值

如果成功，則返回一個 InitSdkOutcome 對象，以指示服務器進程已準備好調用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叢集的 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法需要明確的伺服器參數來設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

參數

ServerParameters

若要初始化 Amazon GameLift Anywhere 叢集上的遊戲伺服器，請使用下列資訊建構`ServerParameters`物件：

- WebSocket 用來連線到遊戲伺服器的 URL。
- 用來託管遊戲伺服器的程序 ID。
- 主控遊戲伺服器處理程序的運算 ID。
- 包含您的 Amazon GameLift Anywhere 運算的 Amazon GameLift 車隊的 ID。
- Amazon GameLift 操作生成的授權令牌。

傳回值

如果成功，則返回一個 InitSdkOutcome 對象，以指示服務器進程已準備好調用ProcessReady()。

> **Note**
>
> 如果部署到 InitSDK() Anywhere 叢集的遊戲組建時呼叫失敗，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使用中的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新組建並將其部署到新的叢集。

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()

通 GameLift 知 Amazon 伺服器程序已準備好主持遊戲工作階段。調用後調用InitSDK()此方法。每個進程應該只調用一次此方法。

語法

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

參數

ProcessParameters

一個ProcessParameters對象保存有關服務器進程的信息。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

這個例子說明了方法和委託函數的實現。

```
// 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()

通 GameLift 知 Amazon 服務器進程正在終止。在所有其他清理任務（包括關閉活動的遊戲會話）之後以及終止進程之前調用此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束，並產生叢集事件。如果程序因錯誤而終止，則產生的叢集事件為SERVER_PROCESS_TERMINATED_UNHEALTHY。

語法

```
GenericOutcome ProcessEnding()
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會Destroy()在使用成功或錯誤結束代碼來終止伺服器處理序之前呼叫ProcessEnding()和。

```
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()

通 GameLift 知 Amazon 伺服器程序已啟動遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲會話初始化之後，應將此操作作為onStartGameSession()回調函數的一部分調用。

語法

```
GenericOutcome ActivateGameSession()
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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

指出遊戲階段作業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

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

擷取使用中伺服器處理序主控的遊戲工作階段 ID。

對於未使用遊戲工作階段啟動的閒置進程，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，則返回錯誤消息。」

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。服務器進程在收到來自 Amazon 的`onProcessTerminate()`回調後採取此操作 GameLift。Amazon GameLift 呼叫`onProcessTerminate()`的原因如下：

- 當服務器進程報告健康狀況不佳或沒有響應 Amazon 時 GameLift。
- 在縮小事件期間終止執行個體時。
- 執行個體因為現場執行個體中斷而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，則返回終止時間作為the section called “AwsDateTimeOutcome”對象。該值是終止時間，以之後經過的刻度表示。`0001 00:00:00`例如，日期時間值等`2020-09-13 12:26:40 -000Z`於`637355968000000000`刻度。如果沒有可用的終止時間，則返回錯誤消息。

範例

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

通知 Amazon 具 GameLift 有指定玩家工作階段 ID 的玩家已連線到伺服器程序並需要驗證。Amazon 會 GameLift 驗證玩家工作階段 ID 是否有效。玩家會話驗證後，Amazon 將播放器插槽的狀態從保留 GameLift 更改為活動狀態。

語法

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

參數

playerSessionId

建立新玩家工作階段 GameLift 時所發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通 GameLift 知 Amazon 播放器已與服務器進程斷開連接。作為回應，Amazon 將播放器插槽 GameLift 更改為可用。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

```
GenericOutcome removePlayerSessionOutcome =
 GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

檢索包括設置，會話元數據和播放器數據的播放器會話數據。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，會傳回包the section called “DescribePlayerSessionsOutcome”含一組符合要求參數之播放程式工作階段物件的物件。

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。透過省略限制值NextToken並將限制值設定為 10，Amazon GameLift 將傳回符合請求的前 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。如果配對新玩家，Amazon 會使用回調函OnUpdateGameSession()數 GameLift 提供更新的分房系統資料。

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

一個StartMatchBackfillRequest對象保存有關回填請求的信息。

傳回值

傳回具有相符回填工單 ID 的the section called “StartMatchBackfillOutcome”物件，或失敗並顯示錯誤訊息。

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()

取消使用中的比對回填要求。如需詳細資訊，請參閱FlexMatch 回填功能。

語法

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

參數

StopMatchBackfillRequest

StopMatchBackfillRequest物件，提供有關您要停止的配對票證的詳細資訊。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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 TLS 憑證的路徑，用來加密遊戲伺服器與遊戲用戶端之間的網路連線。將運算裝置註冊到 Amazon GameLift Anywhere 叢集時，可以使用憑證路徑。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：計算資源上 TLS 憑證的路徑。使用 Amazon 受 GameLift 管叢集時，此路徑包含：
  - `certificate.pem`：一般使用者憑證。完整憑證鏈結是`certificateChain.pem`附加至此憑證的組合。
  - `certificateChain.pem`：包含根憑證和中繼憑證的憑證鏈結。
  - `rootCertificate.pem`：根憑證。
  - `privateKey.pem`：一般使用者憑證的私密金鑰。
- ComputeName：計算資源的名稱。

範例

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
 GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

擷取 IAM 角色登入資料， GameLift 以授權 Amazon 與其他人互動AWS 服務。如需詳細資訊，請參閱與您車隊的其他AWS資源進行溝通。

語法

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
 request);
```

參數

GetFleetRoleCredentialsRequest

將有限AWS資源存取權限延伸至遊戲伺服器的角色認證。

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);
```

摧毀 ( )

從記憶體中釋放 Amazon GameLift 遊戲伺服器 SDK。最佳做法是在終止程序之後ProcessEnding()和之前呼叫此方法。如果您使用的是 Anywhere 叢集，且未在每個遊戲工作階段後終止伺服器程序，請先呼叫Destroy()然InitSDK()後重新初始化，再通知 Amazon GameLift 程序已準備好主持遊戲工作階段。ProcessReady()

語法

```
GenericOutcome Destroy()
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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);
  }
```

用於 C # 和統一的亞馬遜GameLift服務器 SDK 參考：數據類型

這個 Amazon GameLift C# 伺服器開發套件參考可協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器和如需使用適用於 Unity 的 C# 伺服器 SDK 外掛程式的資訊，請參閱 GameLift 將亞馬遜整合到統一項目中。

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

使用此資料類型可識別在遊戲工作階段期間產生的哪些檔案，您希望遊戲伺服器在遊戲工作階段結束GameLift後上傳到 Amazon。遊戲服務器在通ProcessReady()話GameLift中通信LogParameters to亞馬遜。

屬性	Description (描述)
LogPaths	您希望 Amazon GameLift 存放以供將來存取之遊戲伺服器日誌檔的目錄路徑清單。服務器進程在每個遊戲會話期間生成這些文件。您可以在遊

	戲伺服器中定義檔案路徑和名稱，並將它們儲存在根遊戲建置目錄中。 記錄檔路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段記錄儲存在類似路徑中`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

此數據類型包含一組在ProcessReady()調用中發送到亞馬遜GameLift的參數。

屬性	Description (描述)
LogParameters	包含遊戲工作階段記錄檔目錄路徑清單的物件。 Type (類型)：`Aws::GameLift::Server:: LogParameters` 必要：是
OnHealthCheck	Amazon 呼GameLift叫以從伺服器處理序要求健康狀態報告的回呼函數名稱。亞馬遜每 60 秒GameLift調用一次此函數。調用此函數後，亞馬遜GameLift等待 60 秒的響應，如果沒有收到，亞馬遜將服務器進程GameLift記錄為不健康。 Type (類型)：`void OnHealthCheckDelegate()` 必要：是
OnProcessTerminate	亞馬遜調GameLift用以強制服務器進程關閉的回調函數的名稱。呼叫此函數之後，Amazon 會

	GameLift等待五分鐘讓伺服器處理序關閉並回應呼ProcessEnding()叫，然後再關閉伺服器處理序。 Type (類型)：`void OnProcessTerminate Delegate()` 必要：是
OnStartGameSession	亞馬遜調GameLift用以激活新遊戲會話的回調函數的名稱。亞馬遜GameLift調用此函數以響應客戶端請求CreateGameSession。回調函數採用亞馬遜 GameLift API 參考中定義的GameSession對象。 Type (類型)：`void OnStartGameSession Delegate( GameSession )` 必要：是
OnUpdateGameSession	亞馬遜GameLift調用將更新的遊戲會話對象傳遞給服務器進程的回調函數的名稱。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時GameLift呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason`）和匹配回填票證 ID。 類型：無效OnUpdateGameSessionDelegate（UpdateGameSession） 必要：否

連接埠	伺服器處理序偵聽新播放程式連線的連接埠號碼。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

更新了遊戲工作階段物件的資訊，包括遊戲工作階段更新的原因。如果更新與比對回填動作相關，則此資料類型會包含回填工單 ID。

屬性	Description (描述)
GameSession	由亞馬遜 GameLift API 定義的GameSession對象。`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	遊戲工作階段正在更新的原因。 Type (類型)：`UpdateReason UpdateReason()` 必要：是
BackfillTicketId	嘗試更新遊戲工作階段的回填票證 ID。 Type (類型)：`String` 必要：是

GameSession

遊戲工作階段的詳細資料。

屬性	Description (描述)
GameSessionId	遊戲工作階段的唯一識別碼。遊戲工作階段 ARN 的格式如下：`arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 。 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	執行遊戲工作階段之叢集的唯一識別碼。 Type (類型)：`String` 必要：否
MaximumPlayerSessionCount	與遊戲工作階段的玩家連線數目上限。 Type (類型)：`Integer` 必要：否
連接埠	遊戲工作階段的連接埠號碼。若要連線到 Amazon GameLift 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 Type (類型)：`Integer` 必要：否

屬性	Description (描述)
IpAddress	遊戲工作階段的 IP 位址。若要連線到 Amazon GameLift 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 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	關於用來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。除了使用的配對配置外，它還包含指定給該比賽的所有玩家的數據，包括球員屬性和團隊分配。 Type (類型)：`String` 必要：否
GameProperties	遊戲工作階段的一組自訂屬性，格式化為 key:value 配對。這些屬性會與啟動新遊戲工作階段的要求一起傳遞。 Type (類型)：`Dictionary<string, string>` 必要：否

屬性	Description (描述)
DnsName	指派給執行遊戲工作階段之執行個體的 DNS 識別碼。值的格式如下： • 具備 TLS 功能的叢集：。`<unique identifier>.<region identifier>.amazongamelift.com` • 未啟用 TLS 的叢集：`ec2-<unique identifier>.compute.amazonaws.com` 連線至已啟用 TLS 的叢集上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 位址。 Type (類型)：`String` 必要：否

ServerParameters

用於維護亞馬遜服務器和亞馬遜GameLiftAnywhere服GameLift務之間連接的信息。當使用啟動新的伺服器處理序時，會使用此資訊InitSDK()。對於託管在 Amazon GameLift 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description (描述)
WebSocketUrl	當你`RegisterCompute` 作為亞馬遜的一部分`GameLiftServerSdkEndpoint` 返回GameLiftAnywhere。 Type (類型)：`String` 必要：是
ProcessId	註冊到託管您遊戲的伺服器處理序的唯一識別碼。

屬性	Description (描述)
	Type (類型)：`String` 必要：是
HostId	主機與伺服器進程託管您的遊戲的唯一識別碼。HostID 是您註冊計算時ComputeName使用的。有關更多信息，請參閱，RegisterCompute Type (類型)：`String` 必要：是
FleetId	計算所註冊之叢集的叢集識別碼。若要取得更多資訊，請參閱RegisterCompute。 Type (類型)：`String` 必要：是
AuthToken	亞馬遜生成的身份驗證令牌GameLift，用於向亞馬遜驗證您的服務器。GameLift若要取得更多資訊，請參閱GetComputeAuthToken。 Type (類型)：`String` 必要：是

StartMatchBackfillRequest

用於建立配對回填請求的資訊。遊戲服務器通過電話將此信息傳達StartMatchBackfill()給GameLift亞馬遜。

屬性	Description (描述)
GameSessionArn	唯一的遊戲工作階段識別碼。該 API 操作`GetGameSessionId` 返回 ARN 格式的標識符。

屬性	Description (描述)
	Type (類型)：`String` 必要：是
MatchmakingConfigurationArn	分房系統用於此要求的唯一識別碼，以 ARN 的形式呈現。原始遊戲工作階段的分房系統 ARN 位於分房系統資料屬性中的遊戲工作階段物件中。請參閱使用分房系統資料，進一步了解分房系統資料。 Type (類型)：`String` 必要：是
Players	一組數據，代表當前在遊戲會話中的所有玩家。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碼。如果你不提供一個值，亞馬遜GameLift生成一個。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。當配對要求開始時，玩家會擁有玩家 ID、屬性以及可能的延遲資料。亞馬遜在比賽進行後GameLift添加球隊信息。

屬性	Description (描述)
LatencyIn女士	以毫秒為單位表示的一組值，表示玩家連線至某個位置時所經歷的延遲量。 如果使用此屬性，則播放器僅匹配列出的位置。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	包含可用於配對的玩家資訊的 key: 值組合集合。玩家屬性鍵必須與配對規則集中PlayerAttributes使用的鍵相符。 如需播放程式屬性的詳細資訊，請參閱Attribute Value。 Type (類型)：`Dictionary<string, AttributeValue` 必要：否
PlayerId	玩家的唯一識別碼。 Type (類型)：`String` 必要：否
團隊	球員在比賽中被分配到的球隊名稱。您可以在配對規則集中定義小組名稱。 Type (類型)：`String` 必要：否

DescribePlayerSessionsRequest

此資料類型用於指定要擷取的玩家工作階段，它可以通過以下幾種方式使用：（1）提供一PlayerSessionId個請求特定玩家會話;（2）提供一GameSessionId個請求指定遊戲會話中的所有玩家會話; 或（3）提供一PlayerId個請求指定玩家的所有玩家會話。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 (描述)
GameSessionId	唯一的遊戲工作階段識別碼。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。 Type (類型)：`String` 必要：否
PlayerSessionId	玩家工作階段的唯一識別碼。 Type (類型)：`String` 必要：否
PlayerId	玩家的唯一識別碼。請參閱 產生玩家 ID。 Type (類型)：`String` 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。

屬性	Description (描述)
	• 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	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。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

用於取消配對回填要求的資訊。遊戲服務器在通StopMatchBackfill()話中將此信息傳達給亞馬遜GameLift服務。

屬性	Description (描述)
GameSessionArn	要求取消的唯一遊戲工作階段識別碼。 Type (類型)：`string` 必要：是

屬性	Description (描述)
MatchmakingConfigurationArn	傳送此要求的分房系統的唯一識別碼。 Type (類型)：`string` 必要：是
TicketId	要取消之回填要求票證的唯一識別碼。 Type (類型)：`string` 必要：是

GetFleetRoleCredentialsRequest

這種數據類型使遊戲服務器對您的其他AWS資源的訪問有限。如需詳細資訊，請參閱 為 Amazon 設置 IAM 服務角色 GameLift。

屬性	Description (描述)
RoleArn	服務角色的 Amazon 資源名稱 (ARN)，可延伸對AWS資源的有限存取權限。 Type (類型)：`string` 必要：是
RoleSessionName	說明使用角色證明資料之工作階段的名稱。 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、字串陣列或資料對應。每個`AttributeValue`物件只能使用其中一個可用性質。

屬性	Description (描述)
屬性類型	指定屬性值的類型。 類型：**AttrType**枚舉值。 必要：否
S	表示一個字符串屬性值。 Type (類型)：`string` 必要：是
否	表示數值屬性值。 Type (類型)：`double` 必要：是
SL	表示一個字符串屬性值的數組。 Type (類型)：`string[]` 必要：是
代決人	代表字串索引鍵和 double 值的字典。 Type (類型)：`Dictionary<string, double>` 必要：是

AwsStringOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。

屬性	Description (描述)
	Type (類型)：`string` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

GenericOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

DescribePlayerSessionsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。 Type (類型)：the section called “DescribePlayerSessionsResult” 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

DescribePlayerSessionsResult

屬性	Description (描述)
NextToken	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。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	物件集合，其中包含符合要求之每個播放程式工作階段的屬性。 Type (類型)：`IList<`the section called “PlayerSession” `>` 必要：

屬性	Description (描述)
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

PlayerSession

屬性	Description (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` 必要：是

屬性	Description (描述)
PlayerSessionId	Type (類型)：`string` 必要：是
連接埠	Type (類型)：`int` 必要：是
狀態	類型：**`PlayerSessionStatus`** 枚舉。 必要：是
TerminationTime	Type (類型)：`long` 必要：是
DnsName	Type (類型)：`string` 必要：是

StartMatchBackfillOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。 Type (類型)：the section called “StartMatchBackfillResult” 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是

屬性	Description (描述)
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

StartMatchBackfillResult

屬性	Description (描述)
TicketId	Type (類型)：`string` 必要：是

GetComputeCertificateOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。 Type (類型)：the section called “GetComputeCertificateResult” 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError”

屬性	Description (描述)
	必要：否

GetComputeCertificateResult

計算上 TLS 憑證的路徑以及計算機的主機名稱。

屬性	Description (描述)
CertificatePath	Type (類型)：`string` 必要：是
ComputeName	Type (類型)：`string` 必要：是

GetFleetRoleCredentialsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。 Type (類型)：the section called “GetFleetRoleCredentialsResult” 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError”

屬性	Description (描述)
	必要：否

GetFleetRoleCredentialsResult

屬性	Description (描述)
AccessKeyId	用於驗證並提供對AWS資源的存取權限的存取金鑰 ID。 Type (類型)：`string` 必要：否
AssumedRoleId	服務角色所屬的使用者識別碼。 Type (類型)：`string` 必要：否
AssumedRoleUserArn	服務角色所屬之使用者的 Amazon 資源名稱 (ARN)。 Type (類型)：`string` 必要：否
過期	您的工作階段認證到期之前的時間長度。 Type (類型)：`DateTime` 必要：否
SecretAccessKey	用於驗證的秘密存取金鑰 ID。 Type (類型)：`string` 必要：否

屬性	Description (描述)
SessionToken	標識當前活動會話與您的AWS資源進行交互的令牌。 Type (類型)：`string` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

AwsDateTimeOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description (描述)
結果	動作的結果。 Type (類型)：`DateTime` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。

屬性	Description (描述)
	Type (類型)：the section called “GameLiftError” 必要：否

GameLiftError

屬性	Description (描述)
ErrorType	錯誤類型。 類型：**GameLiftErrorType** 枚舉。 必要：否
ErrorName	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

針對亞馬遜GameLift伺服器開發套件 (C#) 定義的枚舉定義如下：

AttrType

- NONE
- 字符串
- 雙
- 字符串列表
- 弦雙地圖

GameLiftErrorType

指示錯誤類型的字符串值。有效值包含：

- 服務呼叫失敗 — 對服務的呼叫失敗。AWS
- 本地連接失敗 — 與亞馬遜的本地連接失敗。GameLift
- 網路不初始化 — 網路尚未初始化。
- 遊戲設定 — 尚未設定遊戲工作階段識別碼。
- 不要求 _ 例外
- 內部 _ 服務 _ 異常
- 已初始化 — 亞馬遜GameLift伺服器或用戶端已經使用初始化 () 進行初始化。
- FLEET_MISMATCH — 目標叢集與遊戲經營或玩家工作階段的機隊不相符。
- 已初始化 — 亞馬遜用戶端尚未初始化。 GameLift
- 已初始化 — 亞馬遜伺服器尚未初始化。GameLift
- 遊戲工作階段失敗 — 亞馬遜GameLift伺服器開發套件無法聯絡服務以報告遊戲工作階段已結束。
- 未啟動亞馬遜伺服器遊戲工作階段。GameLift
- 遊戲工作階段已準備就緒 — 亞馬遜GameLift伺服器開發套件無法聯絡服務以報告遊戲工作階段已準備就緒。
- 初始化不匹配-在服務器:: 初始化（ ）之後調用客戶端方法，反之亦然。
- 初始化 — 亞馬遜GameLift伺服器或用戶端尚未使用初始化 () 初始化。
- 無目標別名-尚未設定目標別名。
- 目標叢集 — 尚未設定目標叢集。
- 處理結束-亞馬遜GameLift伺服器開發套件無法聯絡服務以報告程序已結束。
- PROCES_NOT_ACT_ACTIVE — 伺服器處理作業尚未啟動、未繫結至GameSession，且無法接受或處理。PlayerSessions
- 處理程序 NOT_READY — 伺服器處理作業尚未準備好啟動。
- 處理程序 READY_FAIN — 亞馬遜GameLift伺服器開發套件無法聯絡服務以報告程序已準備就緒。
- SDK_ 版本偵測失敗 — SDK 版本偵測失敗。
- STX_CALL_ 失敗 — 對 XSTX 伺服器後端元件的呼叫失敗。
- STX_ 初始化失敗 — XSTX 伺服器後端元件無法初始化。
- 意外的玩家工作階段 — 伺服器遇到未註冊的玩家工作階段。

- WEB 插槽 _ 連接 _ 失敗
- WEB 插座 _ 連接 _ 失敗 _ 禁止
- 網站插槽 _ 連接 _ 失敗 _ 無效 _URL
- 網路插槽 _ 連線 _ 失敗 _ 逾時
- 網站重新擷取 _ 傳送訊息 _ 失敗 — 將訊息傳送至服務的可擷取失敗。GameLift WebSocket
- 網站傳送訊息失敗 — 無法將訊息傳送至服務。 GameLift WebSocket
- 匹配 _ 後填 _ 請求驗證 — 驗證請求失敗。
- 播放器會話請求驗證 — 請求的驗證失敗。

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玩家工作階段。

PlayerSessionStatus

- 活躍
- COMPLETED (已完成)
- 沒有設定
- 已保留
- 定時

亞馬遜GameLift服務器 SDK 4.x 參考 C＃

這個 Amazon GameLift C# 伺服器 SDK 4.x 參考可協助您準備好與亞馬遜GameLift搭配使用的多人遊戲。如需整合流程的詳細資訊，請參閱 添加亞馬遜GameLift到您的遊戲服務器。

主題

- 亞馬遜GameLift服務器 SDK（C＃）參考：操作
- 亞馬遜GameLift服務器 SDK（C＃）參考：數據類型

亞馬遜GameLift服務器 SDK（C＃）參考：操作

您可以使用這個 Amazon GameLift C# 伺服器 SDK 參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

- 動作
- 資料類型

AcceptPlayerSession()

通知 Amazon GameLift 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並需要驗證。Amazon 會GameLift驗證玩家工作階段 ID 是否有效 — 也就是說，玩家 ID 已在遊戲工作階段中保留一個玩家位置。一旦驗證，亞馬遜GameLift將播放器插槽的狀態從保留更改為活動狀態。

語法

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

參數

playerSessionId

創建新玩家會話GameLift時由亞馬遜發出的唯一 ID。玩家工作階段 ID 是在PlayerSession物件中指定的，該物件是為了回應用戶端對 GameLiftAPI 動作StartGameSessionPlacement、CreateGameSessionDescribeGameSessionPlacement、或的呼叫而傳回DescribePlayerSessions。

類型：字串

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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){
    var acceptPlayerSessionOutcome =
 GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
     if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
```

```
        connection.Accept();
    }
     else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```

ActivateGameSession()

通知 Amazon GameLift 服務伺服器處理序已啟動遊戲工作階段，現在已準備好接收玩家連線。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

傳回包含錯誤訊息的成功或失敗的一般結果。

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

describePlayerSessions請求

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。播放器工作階段物件的結構與 AWS SDK Amazon GameLift 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。透過省略限制值NextToken並將限制值設定為 10，Amazon GameLift 將傳回符合請求的前 10 個玩家工作階段記錄。

```
// 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。

對於尚未通過遊戲會話激活的空閒進程，調用返回 Success = True 和 GameSessionId =""（空字符串）。

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()

擷取與叢集及其執行個體相關聯的 PEM 編碼 TLS 憑證的檔案位置。 AWS Certificate Manager當您建立新的叢集並將憑證組態設定為「已產生」時，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，會傳回包含叢集 TLS 憑證檔案和憑證鏈結位置的GetInstanceCertificateOutcome物件，這些檔案儲存在執行個體上。從憑證鏈結擷取的根憑證檔案也會儲存在執行個體上。如果不成功，則會傳回錯誤訊息。

如需有關憑證和憑證鏈結資料的詳細資訊，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。返回的字符串僅包含版本號（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。伺服器處理序會在收到來自 Amazon GameLift 服務的onProcessTerminate()回呼後採取此動作。Amazon GameLift 可能會因onProcessTerminate()為下列原因而致電：(1) 運作狀態不佳 (伺服器處理序已報告連接埠健康狀態或未回應 AmazonGameLift；(2) 在縮減事件期間終止執行個體時，或 (3) 執行個體因為現場執行個體中斷而終止。

如果進程已收到onProcessTerminate()回調，則返回的值是估計的終止時間。如果處理序未收到onProcessTerminate()回呼，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，則返回終止時間作為AwsDateTimeOutcome對象。該值是終止時間，以自 0001 00:00:00 以來經過的刻度表示。例如，日期時間值 2020-9 月 13 日 12:26：40 -000Z 等於刻度。如果沒有可用的終止時間，則返回錯誤消息。

範例

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

初始化亞馬遜開GameLift發套件。在發生任何其他 Amazon GameLift 相關初始化之前，應在啟動時呼叫此方法。

語法

```
InitSDKOutcome InitSDK()
```

參數

此動作沒有參數。

傳回值

如果成功，則返回一個InitSdkOutcome對象，指示服務器進程已準備好調用ProcessReady()。

範例

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

ProcessEnding()

通知 Amazon GameLift 服務伺服器處理序正在關閉。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。

一旦該方法退出代碼為 0，您可以使用成功的退出代碼終止該進程。您也可以使用錯誤碼結束程序。如果您以錯誤碼結束，叢集事件將指示處理序異常終止 (SERVER_PROCESS_TERMINATED_UNHEALTHY)。

語法

```
GenericOutcome ProcessEnding()
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
   Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

通知 Amazon GameLift 服務伺服器處理序已準備好主持遊戲工作階段。在成功叫用InitSDK()並完成伺服器處理序主控遊戲工作階段之前所需的設定工作之後，呼叫此方法。每個進程應該只調用一次此方法。

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，Amazon GameLift 服務呼叫以與伺服器處理序進行通訊。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案GameLift的路徑。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

通知 Amazon GameLift 服務具有指定玩家工作階段 ID 的玩家已中斷與伺服器處理序的連線。作為回應，亞馬遜將播放器插槽GameLift更改為可用，這允許將其分配給新玩家。

語法

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

參數

playerSessionId

創建新玩家會話GameLift時由亞馬遜發出的唯一 ID。玩家工作階段 ID 是在PlayerSession物件中指定的，該物件是為了回應用戶端對 GameLiftAPI 動作StartGameSessionPlacement、CreateGameSessionDescribeGameSessionPlacement、或的呼叫而傳回DescribePlayerSessions。

類型：字串

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。進一步瞭解FlexMatch回填功能。

此為非同步動作。如果成功配對新玩家，Amazon GameLift 服務會使用回調函OnUpdateGameSession()數提供更新的分房系統資料。

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。此資訊是選擇性的；如果未提供 ID，Amazon GameLift 將自動產生一個 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

傳回具有相符回填工單 ID 或失敗且顯示錯誤訊息的StartMatchBackfillOutcome物件。

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()。進一步瞭解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

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

此方法已被版本 4.0.1 棄用。相反，在遊戲會話結束ProcessEnding()後，服務器進程應該調用。

通知 Amazon GameLift 服務伺服器處理序已結束目前的遊戲工作階段。當伺服器處理程序保持作用中並準備好主持新遊戲工作階段時，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 Amazon 發出訊號GameLift，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關閉伺服器程序，則不會呼叫此動作。相反，調用ProcessEnding()以表示遊戲會話和服務器進程都結束。

語法

```
GenericOutcome TerminateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。（另請參閱亞馬遜GameLift服務 API 參考中的 UpdateGameSession（ ）操作）。

語法

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
 playerSessionPolicy)
```

參數

newPlayerSession政策

字串值代表遊戲工作階段是否可接受新玩家。

類型：PlayerSessionCreationPolicy 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

傳回包含錯誤訊息的成功或失敗的一般結果。

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);
```

亞馬遜GameLift服務器 SDK（C＃）參考：數據類型

您可以使用這個 Amazon GameLift C# 伺服器 SDK 參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

- 動作
- 資料類型

LogParameters

此資料類型用於識別在遊戲工作階段期間產生的檔案，您希望 Amazon GameLift 在遊戲工作階段結束後上傳和存放這些檔案。此信息在通ProcessReady()話中傳達給亞馬遜GameLift服務。

目錄

logPaths

您希望 Amazon GameLift 存放以供將來存取之遊戲伺服器日誌檔的目錄路徑清單。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。記錄檔路徑必須是絕對的。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，它可以通過以下幾種方式使用：（1）提供一PlayerSessionId個請求特定玩家會話;（2）提供一GameSessionId個請求指定遊戲會話中的所有玩家會話; 或（3）提供一PlayerId個請求指定玩家的所有玩家會話。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。搭配使用此參數可NextToken取得一組連續頁面的結果。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 ID。

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

此資料類型包含在ProcessReady()呼叫中傳送至 Amazon GameLift 服務的一組參數。

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

onStartGame階段

Amazon GameLift 服務呼叫以啟動新遊戲工作階段的回呼函數名稱。亞馬遜GameLift調用此函數以響應客戶端請求CreateGameSession。回調函數接受一個GameSession對象（在亞馬遜GameLift服務 API 參考中定義）。

類型：`void OnStartGameSessionDelegate(GameSession gameSession)`

必要：是

onProcessTerminate

Amazon GameLift 服務呼叫以強制伺服器處理序關閉的回呼函數名稱。呼叫此函數之後，Amazon會GameLift等待五分鐘讓伺服器處理序關閉並回應呼ProcessEnding()叫，然後再關閉伺服器處理序。

類型：`void OnProcessTerminateDelegate()`

必要：是

onHealthCheck

Amazon GameLift 服務呼叫以從伺服器處理序要求健康狀態報告的回呼函數名稱。亞馬遜每 60 秒GameLift調用一次此函數。調用此函數後，亞馬遜GameLift等待 60 秒的響應，如果沒有收到任何響應。將服務器進程記錄為不健康。

類型：`bool OnHealthCheckDelegate()`

必要：是

onUpdateGame階段

Amazon GameLift 服務呼叫以將更新的遊戲工作階段物件傳遞至伺服器處理序的回呼函數名稱。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時GameLift呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason`）和匹配回填票證 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。該信息在通StartMatchBackfill()話中傳達給亞馬遜GameLift服務。

目錄

GameSessionArn

獨一無二的遊戲工作階段識別項。該 SDK 方法GetGameSessionId()返回 ARN 格式的標識符。

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。請參閱使用分房系統資料，進一步了解分房系統資料。

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。有關播放器對象格式的說明，請參閱亞馬遜 GameLift 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。如果這裡沒有提供任何值，亞馬遜GameLift將以 UUID 的形式生成一個值。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。該信息在通StopMatchBackfill()話中傳達給亞馬遜GameLift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

適用於圍棋的亞馬遜GameLift服務器 SDK

您可以使用此 Amazon GameLift Go 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

主題

- Amazon GameLift 服務器開發套件（Go）參考：操作
- Amazon GameLift 服務器開發套件（Go）參考：數據類型

Amazon GameLift 服務器開發套件（Go）參考：操作

您可以使用此 Amazon GameLift Go 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

`GameLiftServerAPI.go`定義移至伺服器 SDK 動作。

動作

- GetSdkVersion()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()

- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- 摧毀 ( )

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，則返回當前 SDK 版本作為字符串。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，例如`common.SdkVersionDetectionFailed`。

範例

```
version, err := server.GetSdkVersion()
```

InitSDK()

初始化 Amazon 開 GameLift 發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法設置服務器和 Amazon GameLift 服務之間的通信。

語法

```
func InitSDK(params ServerParameters) error
```

參數

ServerParameters

若要初始化 Amazon GameLift Anywhere 叢集上的遊戲伺服器，請使用下列資訊建構`ServerParameters`物件：

- WebSocket 用來連線到遊戲伺服器的 URL。
- 用來託管遊戲伺服器的程序 ID。
- 主控遊戲伺服器處理程序的運算 ID。
- 包含您的 Amazon GameLift Anywhere 運算的 Amazon GameLift 車隊的 ID。
- Amazon GameLift 操作生成的授權令牌。

若要在 Amazon GameLift 受管 EC2 叢集上初始化遊戲伺服器，請建構不含參數的`ServerParameters`物件。透過此呼叫，Amazon GameLift 代理程式會設定運算環境，並自動為您連線到 Amazon GameLift 服務。

傳回值

如果成功，則返回 `nil` error 以指示服務器進程已準備好調用ProcessReady()。

> **Note**
>
> 如果部署到 `InitSDK()` Anywhere 叢集的遊戲組建時呼叫失敗，請檢查建立組建資源時使用的`ServerSdkVersion`參數。您必須明確地將此值設定為使用中的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新組建並將其部署到新的叢集。

範例

Amazon GameLift Anywhere 示例

```
//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 GameLift agent to enable
 further communication.
err := server.InitSDK(serverParameters)
```

Amazon GameLift 託管 EC2 示例

```
//Define the server parameters
serverParameters := ServerParameters {}

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

通 GameLift 知 Amazon 伺服器程序已準備好主持遊戲工作階段。調用後調用InitSDK()此方法。這種方法應該被調用每個進程只有一次。

語法

```
func ProcessReady(param ProcessParameters) error
```

參數

ProcessParameters

ProcessParameters物件會傳達有關伺服器處理序的下列資訊：

- Amazon GameLift 服務呼叫以與伺服器處理序通訊之遊戲伺服器程式碼中實作的回呼方法名稱。
- 伺服器處理序偵聽的連接埠號碼。
- 包含您希望 Amazon GameLift 擷取和LogParameters存放之任何遊戲工作階段特定檔案路徑的資料類型。

傳回值

如果方法失敗，則傳回錯誤訊息的錯誤。nil如果該方法成功返回。

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()

通 GameLift 知 Amazon 服務器進程正在終止。在所有其他清理任務（包括關閉活動的遊戲會話）之後以及終止進程之前調用此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束，並產生叢集事件。如果程序因錯誤而終止，則產生的叢集事件為SERVER_PROCESS_TERMINATED_UNHEALTHY。

語法

```
func ProcessEnding() error
```

傳回值

傳回 0 錯誤碼或定義的錯誤碼。

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()

通 GameLift 知 Amazon 伺服器程序已啟動遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲會話初始化之後，此操作被稱為onStartGameSession()回調函數的一部分。

語法

```
func ActivateGameSession() error
```

傳回值

如果方法失敗，則傳回錯誤訊息的錯誤。

範例

這個例子顯示了作ActivateGameSession()為onStartGameSession()委託函數的一部分調用。

```
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

playerSessionCreation政策

指出遊戲階段作業是否接受新玩家的字串值。

有效值包含：

- **model.AcceptAll**— 接受所有新玩家會話.
- **model.DenyAll**— 拒絕所有新玩家會話.

傳回值

如果發生失敗，則傳回錯誤訊息的錯誤訊息。

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。

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

擷取使用中伺服器處理序主控的遊戲工作階段 ID。

語法

```
func GetGameSessionID() (string, error)
```

參數

此動作沒有參數。

傳回值

如果成功，返回遊戲會話 ID 和零錯誤。對於尚未通過遊戲會話激活的空閒進程，調用返回一個空字符串和nil錯誤。

範例

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

返回一個服務器進程被計劃關閉，如果終止時間是可用的時間。服務器進程在收到來自 Amazon 的onProcessTerminate()回調後採取此操作 GameLift。Amazon GameLift 呼叫onProcessTerminate()的原因如下：

- 當服務器進程報告健康狀況不佳或沒有響應 Amazon 時 GameLift。

- 在縮小事件期間終止執行個體時。
- 執行個體因為現場執行個體中斷而終止時。

語法

```
func GetTerminationTime() (int64, error)
```

傳回值

如果成功，會傳回伺服器處理序排定關閉的時間戳記 (以紀元秒為單位)，並終止nil錯誤。該值是終止時間，以來0001 00:00:00自的經過刻度表示。例如，日期時間值等2020-09-13 12:26:40 -000Z於637355968000000000刻度。如果沒有可用的終止時間，則返回錯誤消息。

範例

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

通知 Amazon 具 GameLift 有指定玩家工作階段 ID 的玩家已連線到伺服器程序並需要驗證。Amazon 會 GameLift 驗證玩家工作階段 ID 是否有效。玩家會話驗證後，Amazon 將玩家插槽的狀態從 GameLift 更改RESERVED為ACTIVE。

語法

```
func AcceptPlayerSession(playerSessionID string) error
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會處理包含驗證和拒絕非有效玩家工作階段 ID 的連線要求。

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {
    err := server.AcceptPlayerSession(playerSessionID)
    if err != nil {
        connection.Accept()
    } else {
        connection.Reject(err.Error())
    }
}
```

RemovePlayerSession()

通 GameLift 知 Amazon 播放器已與服務器進程斷開連接。作為回應，Amazon 將播放器插槽 GameLift 更改為可用。

語法

```
func RemovePlayerSession(playerSessionID string) error
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

檢索播放器會話數據，其中包括設置，會話元數據和播放器數據。使用此方法可取得下列項目的相關資訊：

- 單一玩家工作階段
- 遊戲工作階段中的所有玩家工作階

- 與單一玩家 ID 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物件描述要擷取哪些玩家工作階段。

傳回值

如果成功，會傳回包DescribePlayerSessionsResult含一組符合要求參數之播放程式工作階段物件的物件。

範例

此範例要求所有主動連線至指定遊戲工作階段的玩家工作階段。透過省略限制值NextToken並將限制值設定為 10，Amazon 會 GameLift 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。如果配對新玩家，Amazon 會使用回調函`OnUpdateGameSession()`數 GameLift 提供更新的分房系統資料。

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

```
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。此資訊是選擇性的；如果未提供 ID，Amazon GameLift 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。此值會顯示在遊戲工作階段的分房系統資料中。
- 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

傳回具有相符回填工單 ID 的`StartMatchBackfillResult`物件，或失敗並顯示錯誤訊息。

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()

取消使用中的比對回填要求。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

識別要取消的配對票證的 StopMatchBackfillRequest 物件：

- 指派給回填請求的工單 ID。
- 回填要求傳送至的分房系統。
- 與回填要求相關聯的遊戲工作階段。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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 TLS 憑證的路徑，用來加密遊戲伺服器與遊戲用戶端之間的網路連線。將運算裝置註冊到 Amazon GameLift Anywhere 叢集時，可以使用憑證路徑。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：計算資源上 TLS 憑證的路徑。使用 Amazon 受 GameLift 管叢集時，此路徑包含：
  - certificate.pem：一般使用者憑證。完整憑證鏈結是certificateChain.pem附加至此憑證的組合。
  - certificateChain.pem：包含根憑證和中繼憑證的憑證鏈結。
  - rootCertificate.pem：根憑證。
  - privateKey.pem：一般使用者憑證的私密金鑰。
- ComputeName：計算資源的名稱。

範例

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

擷取您建立的服務角色登入資料，以將許可擴展AWS 服務到另一個 Amazon GameLift。這些認證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 為 Amazon 設置 IAM 服務角色 GameLift。

語法

```
func GetFleetRoleCredentials(
```

```
  req request.GetFleetRoleCredentialsRequest,
) (result.GetFleetRoleCredentialsResult, error) {
  return srv.getFleetRoleCredentials(&req)
}
```

參數

GetFleetRoleCredentialsRequest

將有限AWS資源存取權限延伸至遊戲伺服器的角色認證。

傳回值

傳回包含下列項目的GetFleetRoleCredentialsResult物件：

- AssumedRoleUserArn -服務角色所屬之使用者的 Amazon 資源名稱 (ARN)。
- AssumedRoleId -服務角色所屬的使用者識別碼。
- AccessKeyId -用於驗證並提供對AWS資源的訪問權限的訪問密鑰 ID。
- SecretAccessKey -用於驗證的密鑰訪問密鑰 ID。
- SessionToken -用於識別與AWS資源交互的當前活動會話的令牌。
- 到期-您的工作階段認證到期之前的時間量。

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)
```

摧毀 ( )

從記憶體中釋放 Amazon GameLift 遊戲伺服器 SDK。最佳做法是在終止程序之後ProcessEnding()和之前呼叫此方法。如果您使用的是 Anywhere 叢集，且未在每個遊戲工作階段後終止伺服器程序，請先呼叫Destroy()然InitSDK()後重新初始化，然後通知 Amazon GameLift 程序已準備好主持遊戲工作階段。ProcessReady()

語法

```
func Destroy() error {
 return srv.destroy()
}
```

傳回值

如果方法失敗，則傳回錯誤訊息的錯誤。

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)
  }
}
```

Amazon GameLift 服務器開發套件（Go）參考：數據類型

您可以使用此 Amazon GameLift Go 伺服器開發套件參考來協助您準備好與 Amazon 搭配使用的多人遊戲 GameLift。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

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物件，您希望 Amazon GameLift 在遊戲工作階段結束後上傳和存放的檔案。遊戲服務器提供`LogParameters`給 Amazon GameLift 作為ProcessReady()調用`ProcessParameters`對象的一部分。

屬性	Description
`LogPaths`	您希望 Amazon GameLift 存放以供 future 來存取的遊戲伺服器記錄檔的目錄路徑清單。服務器進程在每個遊戲會話期間生成這些文件。您可以在遊戲伺服器中定義檔案路徑和名稱，並將它們儲存在根遊戲建置目錄中。 記錄檔路徑必須是絕對的。例如，如果您的遊戲組建將遊戲工作階段記錄儲存在路徑中`MyGame\sessionLogs\`，則路徑將位於 Windows 執行個體`c:\game\MyGame\sessionLogs`上。 Type (類型)：`[]string` 必要：否

ProcessParameters

描述服務器進程和 Amazon 之間通信的對象 GameLift。伺服器處理序會將此資訊提供給 Amazon GameLift ，並呼叫ProcessReady()。

屬性	Description
`LogParameters`	具有遊戲階段作業期間所產生之檔案目錄路徑的物件。Amazon GameLift 複製和存儲文件以供 future 訪問。 Type (類型)：`LogParameters` 必要：否
`OnHealthCheck`	Amazon 呼 GameLift 叫以從伺服器處理序要求健康狀態報告的回呼函數。Amazon 每 60 秒 GameLift 調用一次此函數，並等待 60 秒進行響應。如

	果狀況不良，TRUE則伺服器處理序FALSE會傳回狀況良好。如果沒有回應傳回，Amazon 會將伺服器處理序 GameLift 記錄為狀況不良。 Type (類型)：`OnHealthCheck func() bool` 必要：否
`OnProcessTerminate`	Amazon GameLift 調用以強制服務器進程關閉的回調函數。呼叫此函數之後，Amazon 會 GameLift 等待 5 分鐘讓伺服器處理序關閉並回應呼ProcessEnding()叫，然後再關閉伺服器處理序。 Type (類型)：`OnProcessTerminate func()` 必要：是
`OnStartGameSession`	Amazon GameLift 調用將更新的遊戲會話對象傳遞給服務器進程的回調函數。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時 GameLift 呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason` ）和匹配回填票證 ID。 Type (類型)：`OnStartGameSession func (model.GameSession )` 必要：是
`OnUpdateGameSession`	Amazon GameLift 調用將更新的遊戲會話信息傳遞給服務器進程的回調函數。Amazon 在處理比賽回填請求後 GameLift 呼叫此函式，以提供更新的分房系統資料。 Type (類型)：`OnUpdateGameSession func (model.UpdateGameSession)` 必要：否
`Port`	伺服器處理序偵聽新播放程式連線的連接埠號碼。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，其中包括更新遊戲工作階段的原因，以及如果使用回填填補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	由 Amazon GameLift API 定義的GameSession對象。`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	遊戲工作階段正在更新的原因。 Type (類型)：`UpdateReason UpdateReason()` 必要：是
BackfillTicketId	嘗試更新遊戲工作階段的回填票證 ID。 Type (類型)：`String` 必要：否

GameSession

一個遊戲會話的細節。

屬性	Description
GameSessionId	遊戲工作階段的唯一識別碼。遊戲會話 Amazon 資源名稱（ARN）具有以下格式：`arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 。 Type (類型)：`String` 必要：否
名稱	遊戲工作階段的描述性標籤。

屬性	Description
	Type (類型)：`String` 必要：否
FleetId	執行遊戲工作階段之叢集的唯一識別碼。 Type (類型)：`String` 必要：否
MaximumPlayerSessionCount	與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 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 Type (類型)：`Integer` 必要：否
IpAddress	遊戲工作階段的 IP 位址。若要連線到 Amazon GameLift 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 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

屬性	Description
MatchmakerData	關於用來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。除了使用的配對配置外，它還包含了所有指定給比賽的玩家的數據，包括球員屬性和團隊分配。 Type (類型)：`String` 必要：否
GameProperties	遊戲工作階段的一組自訂屬性，格式化為 key: value 配對。這些屬性會與啟動新遊戲工作階段的要求一起傳遞。 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 識別碼。值的格式如下： • 具備 TLS 功能的叢集：。`<unique identifier>.<region identifier>.amazongamelift.com` • 未啟用 TLS 的叢集：`ec2-<unique identifier>.compute.amazonaws.com` 連線至已啟用 TLS 的叢集上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 位址。 Type (類型)：`String` 必要：否

ServerParameters

用於維護 Amazon 服務 GameLift Anywhere器和 Amazon 服 GameLift 務之間連接的信息。使用啟動新伺服器處理序時會使用此資訊InitSDK()。對於託管在 Amazon GameLift 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
`WebSocket URL`	`GameLiftServerSdkEndpoint` Amazon GameLift 返回時，你`RegisterCompute` 的 Amazon GameLift Anywhere 計算資源。 Type (類型)：`string` 必要：是
`ProcessID`	註冊到託管您遊戲的伺服器處理序的唯一識別碼。 Type (類型)：`string` 必要：是
`HostID`	託管新伺服器處理序之計算資源的唯一識別碼。 `HostID`這是您註冊運算時`ComputeName` 使用的。如需詳細資訊，請參閱RegisterCompute。 Type (類型)：`string` 必要：是
`FleetID`	計算所註冊之叢集的唯一識別碼。如需詳細資訊，請參閱RegisterCompute。 Type (類型)：`string` 必要：是
`AuthToken`	Amazon 生成的身份驗證令牌 GameLift ，用於向 Amazon 驗證您的服務器。GameLift如需詳細資訊，請參閱GetComputeAuthToken。 Type (類型)：`string` 必要：是

StartMatchBackfillRequest

用於建立配對回填請求的資訊。遊戲服務器通過電話將此信息傳達StartMatchBackfill()給 Amazon GameLift 。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別碼。該 API 操作GetGameSessionId 返回 ARN 格式的標識符。 Type (類型)：String 必要：是
MatchmakingConfigurationArn	分房系統用於此要求的唯一識別碼 (以 ARN 的形式)。原始遊戲工作階段的分房系統 ARN 位於分房系統資料屬性中的遊戲工作階段物件中。如需分房系統資料的詳細資訊，請參閱使用分房系統資料。 Type (類型)：String 必要：是
Players	一組數據，代表當前在遊戲會話中的所有玩家。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碼。如果你不提供一個值，Amazon GameLift 生成一個。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。當配對要求開始時，玩家會擁有玩家 ID、屬性以及可能的延遲資料。Amazon 在比賽進行後 GameLift 添加球隊信息。

屬性	Description
LatencyIn女士	以毫秒為單位表示的一組值，表示玩家連線至某個位置時所經歷的延遲量。

屬性	Description
	如果使用此屬性，則播放器僅匹配列出的位置。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	包含可用於配對的玩家資訊的 key: 值組合集合。玩家屬性鍵必須與配對規則集中 PlayerAttributes 使用的鍵相符。 如需播放程式屬性的詳細資訊，請參閱AttributeValue。 Type (類型)：`map[string] AttributeValue` 必要：否
PlayerId	玩家的唯一識別碼。 Type (類型)：`String` 必要：否
團隊	球員在比賽中被分配到的球隊名稱。您可以在配對規則集中定義小組名稱。 Type (類型)：`String` 必要：否

DescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。服務器進程通過DescribePlayerSessions()致電 Amazon 提供此信息 GameLift。

屬性	Description
`GameSessionID`	唯一的遊戲工作階段識別碼。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。

屬性	Description
	遊戲工作階段 ID 格式為arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string> . 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碼。使用此參數可要求單一特定玩家工作階段。 Type (類型)：String 必要：否
PlayerID	玩家的唯一識別碼。使用此參數可要求特定玩家的所有玩家工作階段。請參閱產生玩家 ID。 Type (類型)：String 必要：否
PlayerSessionStatusFilter	用於篩選結果的玩家工作階段狀態。玩家工作階段狀態可能包括： • 保留 — 已收到播放器工作階段要求，但播放程式尚未連線至伺服器處理序或已驗證。 • 作用中 — 播放程式已由伺服器處理序驗證並已連線。 • 已完成 — 玩家連線中斷。 • TIMEDOUT — 收到玩家工作階段要求，但玩家沒有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	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。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

用於取消配對回填要求的資訊。遊戲服務器在通話中將此信息傳達StopMatchBackfill()給 Amazon GameLift 服務。

屬性	Description
GameSessionArn	要求取消的唯一遊戲工作階段識別碼。 Type (類型)：string 必要：否
MatchmakingConfigurationArn	傳送此要求的分房系統的唯一識別碼。 Type (類型)：string 必要：否
TicketId	要取消之回填要求票證的唯一識別碼。 Type (類型)：string 必要：否

GetFleetRoleCredentialsRequest

將有限AWS資源存取權限延伸到遊戲伺服器的角色認證。若要取得更多資訊，請參閱為 Amazon 設置 IAM 服務角色 GameLift。

屬性	Description
RoleArn	服務角色的 ARN，可延伸對AWS資源的有限存取權限。 Type (類型)：`string` 必要：是
RoleSessionName	說明使用角色證明資料之工作階段的名稱。 Type (類型)：`string` 必要：是

虛幻引擎的亞馬遜GameLift服務器 SDK 參考

這個亞馬遜GameLift伺服器開發套件參考可以幫助您準備虛幻引擎遊戲專案，以便與亞馬遜GameLift搭配使用。如需整合流程的詳細資訊，請參閱 添加亞馬遜GameLift到您的遊戲服務器。

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 GameLift 將 Amazon 集成到虛幻引擎項目中。

主題

- 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考
- 亞馬遜GameLift虛幻引擎服務器 SDK 3.x 參考

亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考

您可以使用此亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考來幫助您準備與亞馬遜一起使用的多人遊戲。GameLift如需有關整合程序的詳細資訊，請參閱和添加亞馬遜GameLift到您的遊戲服務器，以取得有關使用 Unreal SDK 伺服器外掛程式的資訊，請參閱 GameLift 將 Amazon 集成到虛幻引擎項目中。

主題

- Amazon GameLift 服務器 SDK（虛幻）5.x 參考：操作
- Amazon GameLift 服務器開發套件（虛幻）參考：數據類型

Amazon GameLift 服務器 SDK（虛幻）5.x 參考：操作

您可以使用這個 Amazon GameLift 虛幻伺服器開發套件參考來協助您準備好與 Amazon GameLift 搭配使用的多人遊戲。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器和如需使用 Unreal SDK 伺服器外掛程式的資訊，請參閱 GameLift 將 Amazon 集成到虛幻引擎項目中。

動作

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()

> **Note**
>
> 本主題說明您在為虛幻引擎建置時可以使用的 Amazon GameLift 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。

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 “F GameLiftStringOutcome” 物件傳回。傳回的物件包含版本號碼 (範例`5.0.0`)。如果不成功，則會傳回錯誤訊息。

範例

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
 Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

為受管 EC2 叢集初始化 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法會從主機環境讀取伺服器參數，以設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
FGameLiftGenericOutcome InitSDK()
```

傳回值

如果成功，則返回一個`InitSdkOutcome`對象，指示服務器進程已準備好調用ProcessReady()。

範例

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
FGameLiftGenericOutcome initSdkOutcome = gameLiftSdkModule->InitSDK();
```

InitSDK()

初始化Anywhere叢集的 Amazon GameLift 開發套件。在與 Amazon 相關的任何其他初始化發生之前，請在啟動 GameLift 時呼叫此方法。此方法需要明確的伺服器參數來設定伺服器與 Amazon GameLift 服務之間的通訊。

語法

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

參數

F ServerParameters

若要初始化 Amazon GameLift Anywhere 叢集上的遊戲伺服器，請使用下列資訊建構`ServerParameters`物件：

- WebSocket 用來連線到遊戲伺服器的 URL。
- 用來託管遊戲伺服器的程序 ID。
- 主控遊戲伺服器處理程序的運算 ID。
- 包含您的 Amazon GameLift Anywhere 運算的 Amazon GameLift 車隊的 ID。
- Amazon GameLift 操作生成的授權令牌。

傳回值

如果成功，則返回一個`InitSdkOutcome`對象，指示服務器進程已準備好調用ProcessReady()。

> **Note**
>
> 如果部署到 InitSDK() Anywhere 叢集的遊戲組建時呼叫失敗，請檢查建立組建資源時使用的`ServerSdkVersion`參數。您必須明確地將此值設定為使用中的伺服器 SDK 版本。此參數的預設值為 4.x，不相容。若要解決此問題，請建立新組建並將其部署到新的叢集。

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 GameLift agent to enable
 further communication.
FGameLiftGenericOutcome initSdkOutcome = gameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

通 GameLift 知 Amazon 伺服器程序已準備好主持遊戲工作階段。調用後調用InitSDK()此方法。每個進程應該只調用一次此方法。

語法

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

參數

processParameters

通訊有關伺服器處理序的下列資訊的F ProcessParameters物件：

- Amazon GameLift 服務呼叫以與伺服器處理序通訊之遊戲伺服器程式碼中實作的回呼方法名稱。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案 GameLift 的路徑。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會說明 ProcessReady() 呼叫和委派函數的實作。

```
//Calling ProcessReady tells GameLift this game server is ready to receive incoming
 game sessions!
```

```
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = gameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

通 GameLift 知 Amazon 服務器進程正在終止。在所有其他清理任務（包括關閉活動的遊戲會話）之後以及終止進程之前調用此方法。根據的結果ProcessEnding()，程序會以成功 (0) 或錯誤 (-1) 結束，並產生叢集事件。如果處理序因錯誤而終止，則產生的叢集事件為SERVER_PROCESS_TERMINATED_UNHEALTHY)。

語法

```
FGameLiftGenericOutcome ProcessEnding()
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

```
//OnProcessTerminate callback. GameLift 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 GameLift we are indeed going to shutdown.
params->OnTerminate.BindLambda([=]() {
  UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
  gameLiftSdkModule->ProcessEnding();
});
```

ActivateGameSession()

通 GameLift 知 Amazon 伺服器程序已啟動遊戲工作階段，現在已準備好接收玩家連線。在所有遊戲會話初始化之後，應將此操作作為onStartGameSession()回調函數的一部分調用。

語法

```
FGameLiftGenericOutcome ActivateGameSession()
```

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

這個例子顯示了作ActivateGameSession()為onStartGameSession()委託函數的一部分調用。

```
//When a game session is created, GameLift 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();
};
```

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

playerCreationSession政策

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

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

擷取使用中伺服器處理序主控的遊戲工作階段 ID。

對於未使用遊戲工作階段啟動的閒置進程，呼叫會傳回the section called “F GameLiftError”.

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 “F GameLiftStringOutcome” 物件傳回。如果不成功，則返回錯誤消息。」

對於未使用遊戲會話激活的空閒進程，調用返回 Success = True 和 GameSessionId = ""。

範例

```
//When a game session is created, GameLift 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。服務器進程在收到來自 Amazon 的`onProcessTerminate()`回調後採取行動 GameLift。Amazon GameLift 呼叫`onProcessTerminate()`的原因如下：

- 當服務器進程報告健康狀況不佳或沒有響應 Amazon 時 GameLift。
- 在縮小事件期間終止執行個體時。
- 執行個體因為現場執行個體中斷而終止時。

語法

```
AwsDateTimeOutcome GetTerminationTime()
```

傳回值

如果成功，則返回終止時間作為`AwsDateTimeOutcome`對象。該值是終止時間，以之後經過的刻度表示。`0001 00:00:00`例如，日期時間值等`2020-09-13 12:26:40 -000Z`於`637355968000000000`刻度。如果沒有可用的終止時間，則返回錯誤消息。

如果進程沒有收到回 `ProcessParameters.OnProcessTerminate()`調，則返回錯誤消息。如需關閉伺服器處理序的詳細資訊，請參閱回應伺服器處理序關閉通知。

範例

```
AwsDateTimeOutcome TermTimeOutcome = gameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

通知 Amazon 具 GameLift 有指定玩家工作階段 ID 的玩家已連線到伺服器程序並需要驗證。Amazon 會 GameLift 驗證玩家工作階段 ID 是否有效。玩家會話驗證後，Amazon 將播放器插槽的狀態從保留 GameLift 更改為活動狀態。

語法

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

此範例會處理包含驗證和拒絕非有效玩家工作階段 ID 的連線要求。

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerSessionId, const
 FString& playerId)
{
  #if WITH_GAMELIFT
  UE_LOG(GameServerLog, Log, TEXT("Accepting GameLift PlayerSession: %s . PlayerId:
 %s"), *playerSessionId, *playerId);
  FString gsId = GetCurrentGameSessionId();
  if (gsId.IsEmpty()) {
    UE_LOG(GameServerLog, Log, TEXT("No GameLift GameSessionId. Returning early!"));
    return false;
  }

  if (!gameLiftSdkModule->AcceptPlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession not Accepted."));
    return false;
  }

  // Add PlayerSession from internal data structures keeping track of connected players
  connectedPlayerSessionIds.Add(playerSessionId);
  idToPlayerSessionMap.Add(playerSessionId, PlayerSession{ playerId,
 playerSessionId });
  return true;
  #else
  return false;
  #endif
```

```
}
```

RemovePlayerSession()

通 GameLift 知 Amazon 播放器已與服務器進程斷開連接。作為回應，Amazon 將播放器插槽 GameLift 更改為可用。

語法

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

參數

playerSessionId

創建新玩家會話 GameLift 時由 Amazon 發出的唯一 ID。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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()

檢索播放器會話數據，其中包括設置，會話元數據和播放器數據。使用此方法可取得下列項目的相關資訊：

- 單一玩家工作階段
- 遊戲工作階段中的所有玩家工作階
- 與單一玩家 ID 關聯的所有玩家工作階段

語法

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
 FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

參數

F GameLiftDescribePlayerSessionsRequest

描述要擷取哪些玩家工作階段的the section called “F GameLiftDescribePlayerSessionsRequest”物件。

傳回值

如果成功，會傳回 the section called “F GameLiftDescribePlayerSessionsOutcome” 物件，內含一組與請求參數相符的玩家工作階段物件。

範例

此範例要求所有主動連線至指定遊戲工作階段的玩家工作階段。透過省略限制值NextToken並將限制值設定為 10，Amazon 會 GameLift 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。如果配對新玩家，Amazon 會使用回調函`OnUpdateGameSession()`數 GameLift 提供更新的分房系統資料。

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

F StartMatchBackfillRequest

傳達下列資訊的 StartMatchBackfillRequest 物件：

- 指派給回填請求的票證 ID。此信息是可選的; 如果沒有提供 ID，Amazon GameLift 將生成一個。

- 傳送請求對象的配對建構器。必須填入完整的組態 ARN。此值會顯示在遊戲工作階段的分房系統資料中。
- 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

傳回具有相符回填工單 ID 的`StartMatchBackfillOutcome`物件，或失敗並顯示錯誤訊息。

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()

取消使用中的比對回填要求。如需詳細資訊，請參閱FlexMatch回填功能。

語法

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

參數

F StopMatchBackfillRequest

識別要取消配對票證的 StopMatchBackfillRequest 物件：

- 指派給回填請求的工單 ID。
- 回填要求傳送至的分房系統。
- 與回填要求相關聯的遊戲工作階段。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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 TLS 憑證的路徑，用於加密 Amazon GameLift Anywhere 運算資源和 Amazon 之間的網路連線 GameLift。將運算裝置註冊到 Amazon GameLift Anywhere 叢集時，可以使用憑證路徑。若要取得更多資訊，請參閱RegisterCompute。

語法

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

傳回值

返回包含以下內容的GetComputeCertificateResponse對象：

- CertificatePath：計算資源上 TLS 憑證的路徑。
- HostName：計算資源的主機名稱。

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_TO_TCHAR(outres.GetCertificatePath());
    result.m_computeName = UTF8_TO_TCHAR(outres.GetComputeName());
    return FGameLiftGetComputeCertificateOutcome(result);
  }
  else {
    return FGameLiftGetComputeCertificateOutcome(FGameLiftError(outcome.GetError()));
  }
  #else
  return FGameLiftGetComputeCertificateOutcome(FGameLiftGetComputeCertificateResult());
  #endif
}
```

GetFleetRoleCredentials()

擷取 IAM 角色登入資料， GameLift 以授權 Amazon 與其他人互動AWS 服務。如需詳細資訊，請參閱[與您車隊的其他AWS資源進行溝通](#)。

語法

```
FGameLiftGetFleetRoleCredentialsOutcome
 FGameLiftServerSDKModule::GetFleetRoleCredentials(const
 FGameLiftGetFleetRoleCredentialsRequest &request)
```

參數

F GameLiftGetFleetRoleCredentialsRequest

傳回值

其會傳回 the section called “F GameLiftGetFleetRoleCredentialsOutcome” 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
}
```

Amazon GameLift 服務器開發套件（虛幻）參考：數據類型

您可以使用這個 Amazon GameLift 虛幻伺服器開發套件參考來協助您準備好與 Amazon GameLift 搭配使用的多人遊戲。如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器和如需使用 Unreal SDK 伺服器外掛程式的資訊，請參閱 GameLift 將 Amazon 集成到虛幻引擎項目中。

資料類型

- F ProcessParameters
- UpdateGameSession
- GameSession
- F ServerParameters
- F StartMatchBackfillRequest
- FPLAYER
- F GameLiftDescribePlayerSessionsRequest
- F StopMatchBackfillRequest
- F AttributeValue
- F GameLiftGetFleetRoleCredentialsRequest
- F GameLiftLongOutcome
- F GameLiftStringOutcome
- F GameLiftDescribePlayerSessionsOutcome
- F GameLiftDescribePlayerSessionsResult
- F GenericOutcome
- F GameLiftPlayerSession
- F GameLiftGetComputeCertificateOutcome
- F GameLiftGetComputeCertificateResult
- F GameLiftGetFleetRoleCredentialsOutcome
- F GetFleetRoleCredentialsResult
- F GameLiftError
- 列舉

> **Note**
>
> 本主題說明您在為虛幻引擎建置時可以使用的 Amazon GameLift 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。

F ProcessParameters

此數據類型包含一組發送到 Amazon GameLift 的參數ProcessReady()。

屬性	Description
LogParameters	具有遊戲階段作業期間所產生之檔案目錄路徑的物件。Amazon GameLift 複製和存儲文件以供 future 訪問。 Type (類型)：`TArray<FString>` 必要：否
OnHealthCheck	Amazon 呼 GameLift 叫以從伺服器處理序要求健康狀態報告的回呼函數。Amazon 每 60 秒 GameLift 調用一次此函數，並等待 60 秒進行響應。如果狀況不良，`TRUE`則伺服器處理序`FALSE`會傳回狀況良好。如果未傳回任何回應，Amazon 會將伺服器處理序 GameLift 記錄為狀況不良。 這個屬性是一個委託函數，定義為`DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)`； Type (類型)：`FOnHealthCheck` 必要：否
OnProcessTerminate	Amazon GameLift 調用以強制服務器進程關閉的回調函數。呼叫此函數之後，Amazon 會 GameLift 等待 5 分鐘讓伺服器處理序關閉並回

	應呼ProcessEnding()叫，然後再關閉伺服器處理序。 Type (類型)：`FSimpleDelegate` 必要：是
OnStartGameSession	Amazon GameLift 調用以激活新遊戲會話的回調函數。Amazon GameLift 調用此函數以響應客戶請求CreateGameSession。回呼函數會傳遞GameSession物件，如 Amazon GameLift API 參考中所定義。 這個屬性是一個委託函數，定義為 `DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);` Type (類型)：`FOnStartGameSession` 必要：是
OnUpdateGameSession	Amazon GameLift 調用將更新的遊戲會話對象傳遞給服務器進程的回調函數。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時 GameLift 呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason` ）和匹配回填票證 ID。 這個屬性是一個委託函數，定義為 `DECLARE_DELEGATE_OneParam(FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession);` Type (類型)：`FOnUpdateGameSession` 必要：否

連線埠	伺服器處理序偵聽新播放程式連線的連接埠號碼。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

此資料類型會更新為遊戲工作階段物件，其中包括更新遊戲工作階段的原因，以及如果使用回填填補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	由 Amazon GameLift API 定義的GameSession對象。`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	遊戲工作階段正在更新的原因。 Type (類型)：`enum class UpdateReason` • 比對資料更新 • 回填失敗 (_C) • 回填 _ 定時 _ 輸出 • 回填 (_ 已取消) 必要：否
BackfillTicketId	嘗試更新遊戲工作階段的回填票證 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碼。遊戲工作階段 ARN 的格式如下：arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token> 。 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	執行遊戲工作階段之叢集的唯一識別碼。 Type (類型)：char[] 必要：否
MaximumPlayerSessionCount	與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 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 Type (類型)：`int` 必要：否
IpAddress	遊戲工作階段的 IP 位址。若要連線到 Amazon GameLift 遊戲伺服器，應用程式需要 IP 位址和連接埠號碼。 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	有關用於建立遊戲工作階段的配對程序的資訊，以 JSON 語法格式化為字串。除了使用的配對配置外，它還包含了所有指定給比賽的玩家的數據，包括球員屬性和團隊分配。 Type (類型)：`char[]` 必要：否
GameProperties	遊戲工作階段的一組自訂屬性，格式化為 key:value 配對。這些屬性會與啟動新遊戲工作階段的要求一起傳遞。 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 識別碼。值的格式如下： • 具備 TLS 功能的叢集：。`<unique identifier>.<region identifier>.amazongamelift.com` • 未啟用 TLS 的叢集：`ec2-<unique identifier>.compute.amazonaws.com` 連線至已啟用 TLS 的叢集上執行的遊戲工作階段時，您必須使用 DNS 名稱，而非 IP 位址。 Type (類型)：`char[]` 必要：否

F ServerParameters

用於維護 Amazon 服務 GameLift Anywhere器和 Amazon 服 GameLift 務之間連接的信息。當使用啟動新的伺服器處理序時，會使用此資訊InitSDK()。對於託管在 Amazon GameLift 受管 EC2 執行個體上的伺服器，請使用空物件。

屬性	Description
webSocketUrl	`GameLiftServerSdkEndpoint` Amazon GameLift 返回當你一`RegisterCompute` 個 Amazon GameLift Anywhere 計算資源。 Type (類型)：`char[]` 必要：是
流程	註冊到託管您遊戲的伺服器處理序的唯一識別碼。

屬性	Description
	Type (類型)：`char[]` 必要：是
hostId	`HostID`這是您註冊運算時`ComputeName` 使用的。若要取得更多資訊，請參閱RegisterCompute。 Type (類型)：`char[]` 必要：是
飛行	計算所註冊之叢集的唯一識別碼。若要取得更多資訊，請參閱RegisterCompute。 Type (類型)：`char[]` 必要：是
驗證令牌	Amazon 生成的身份驗證令牌 GameLift ，用於向 Amazon 驗證您的服務器。 GameLift若要取得更多資訊，請參閱GetComputeAuthToken。 Type (類型)：`char[]` 必要：是

F StartMatchBackfillRequest

用於建立配對回填請求的資訊。遊戲服務器通過電話將此信息傳達StartMatchBackfill()給 Amazon GameLift 。

屬性	Description
GameSessionArn	唯一的遊戲工作階段識別碼。該 API 操作`GetGameSessionId` 返回 ARN 格式的標識符。

屬性	Description
	Type (類型)：`char[]` 必要：是
MatchmakingConfigurationArn	一個唯一的識別碼，以 ARN 的形式，供分房系統用於此要求。原始遊戲工作階段的分房系統 ARN 位於分房系統資料屬性中的遊戲工作階段物件中。請參閱使用分房系統資料，進一步了解分房系統資料。 Type (類型)：`char[]` 必要：是
Players	一組數據，代表在遊戲會話中的所有玩家。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碼。如果你不提供一個值，Amazon GameLift 生成一個。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。開始配對要求時，玩家會擁有玩家 ID、屬性以及可能的延遲資料。Amazon 在比賽進行後 GameLift 添加球隊信息。

屬性	Description
LatencyIn女士	以毫秒為單位表示的一組值，表示玩家連線至某個位置時所經歷的延遲量。 如果使用此屬性，則播放器僅匹配列出的位置。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	包含可用於配對的玩家資訊的 key: 值組合集合。玩家屬性鍵必須與配對規則集中 PlayerAttributes 使用的鍵相符。 如需播放程式屬性的詳細資訊，請參閱Attribute Value。 Type (類型)：`TMap>FString, FAttributeValue<` 必要：否
PlayerId	玩家的唯一識別碼。 Type (類型)：`std::string` 必要：否
團隊	球員在比賽中被分配到的球隊名稱。您可以在配對規則集中定義小組名稱。 Type (類型)：`FString` 必要：否

F GameLiftDescribePlayerSessionsRequest

指定要擷取哪些玩家工作階段的物件。服務器進程通過DescribePlayerSessions()致電 Amazon 提供此信息 GameLift。

屬性	Description
GameSessionId	唯一的遊戲工作階段識別碼。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碼。使用此參數可要求單一特定玩家工作階段。 Type (類型)：`FString` 必要：否
PlayerId	玩家的唯一識別碼。使用此參數可要求特定玩家的所有玩家工作階段。請參閱 產生玩家 ID。 Type (類型)：`FString` 必要：否
PlayerSessionStatusFilter	用於篩選結果的玩家工作階段狀態。玩家工作階段狀態可能包括： • 保留 — 已收到播放器工作階段要求，但播放程式尚未連線至伺服器處理序或已驗證。 • 作用中 — 播放程式已由伺服器處理序驗證並已連線。 • 已完成 — 玩家連線中斷。

屬性	Description
	• TIMEDOUT — 收到玩家工作階段要求，但玩家沒有連線或未在逾時限制 (60 秒) 內驗證。 Type (類型)：`FString` 必要：否
NextToken	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。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

F StopMatchBackfillRequest

用於取消配對回填要求的資訊。遊戲服務器在通話中將此信息傳達StopMatchBackfill()給 Amazon GameLift 服務。

屬性	Description
GameSessionArn	要求取消的唯一遊戲工作階段識別碼。 Type (類型)：`FString` 必要：是
MatchmakingConfigurationArn	傳送此要求的分房系統的唯一識別碼。 Type (類型)：`FString`

屬性	Description
	必要：是
TicketId	要取消之回填要求票證的唯一識別碼。 Type (類型)：FString 必要：是

F AttributeValue

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、字串陣列或資料對應。每個AttributeValue物件只能使用其中一個可用性質。

屬性	Description
屬性類型	指定屬性值的類型。 類型：**FAttributeType** 枚舉值。 必要：否
S	表示一個字符串屬性值。 Type (類型)：FString 必要：否
N	表示數值屬性值。 Type (類型)：double 必要：否
SL	表示一個字符串屬性值的數組。 Type (類型)：TArray<FString> 必要：否
代決人	代表字串索引鍵和 double 值的字典。

屬性	Description
	Type (類型)：`TMap<FString, double>` 必要：否

F GameLiftGetFleetRoleCredentialsRequest

此資料類型提供角色認證，可將有限的AWS資源存取權限延伸至遊戲伺服器。若要取得更多資訊，請參閱為 Amazon 設置 IAM 服務角色 GameLift。

屬性	Description
RoleArn	服務角色的 Amazon 資源名稱 (ARN)，可延伸對AWS資源的有限存取權限。 Type (類型)：`FString` 必要：否
RoleSessionName	說明使用角色認證之工作階段的名稱。 Type (類型)：`FString` 必要：否

F GameLiftLongOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：`long` 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。

屬性	Description
	Type (類型)：`long&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GameLiftStringOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：`FString` 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`FString&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool`

屬性	Description
	必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GameLiftDescribePlayerSessionsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “F GameLiftDescribePlayerSessionsResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`FGameLiftDescribePlayerSessionsResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。

屬性	Description
	Type (類型)：the section called “F GameLiftError” 必要：否

F GameLiftDescribePlayerSessionsResult

屬性	Description
PlayerSessions	Type (類型)：`TArray<FGameLiftPlayerSession>` 必要：是
NextToken	表示下一頁結果開始的令牌。若要指定結果集的開頭，請勿提供值。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
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GenericOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GameLiftPlayerSession

屬性	Description
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`

屬性	Description
	必要：是
PlayerId	Type (類型)：`FString` 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`** 枚舉。 必要：是
TerminationTime	Type (類型)：`long` 必要：是
DnsName	Type (類型)：`FString` 必要：是

F GameLiftGetComputeCertificateOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “F GameLiftGetComputeCertificateResult” 必要：否

屬性	Description
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`FGameLiftGetComputeCertificateResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GameLiftGetComputeCertificateResult

計算上 TLS 憑證的路徑以及計算機的主機名稱。

屬性	Description
CertificatePath	Type (類型)：`FString` 必要：是
ComputeName	Type (類型)：`FString` 必要：是

F GameLiftGetFleetRoleCredentialsOutcome

此資料類型是由動作產生的，並產生具有下列屬性的物件：

屬性	Description
結果	動作的結果。 Type (類型)：the section called “F GetFleetRoleCredentialsResult” 必要：否
ResultWithOwnership	動作的結果，轉換為右值，以便調用代碼可以獲得對象的所有權。 Type (類型)：`FGameLiftGetFleetRoleCredentialsResult&&` 必要：否
Success (成功)	動作是否成功。 Type (類型)：`bool` 必要：是
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “F GameLiftError” 必要：否

F GetFleetRoleCredentialsResult

屬性	Description
AccessKeyId	用於驗證並提供AWS資源存取權的存取金鑰ID。 Type (類型)：`FString` 必要：否

屬性	Description
AssumedRoleId	服務角色所屬的使用者識別碼。 Type (類型)：FString 必要：否
AssumedRoleUserArn	服務角色所屬之使用者的 Amazon 資源名稱 (ARN)。 Type (類型)：FString 必要：否
過期	您的工作階段認證到期之前的時間長度。 Type (類型)：FDateTime 必要：否
SecretAccessKey	用於驗證的秘密存取金鑰 ID。 Type (類型)：FString 必要：否
SessionToken	標識當前活動會話與您的AWS資源進行交互的令牌。 Type (類型)：FString 必要：否
Success (成功)	動作是否成功。 Type (類型)：bool 必要：是

屬性	Description
錯誤	處理行動失敗時發生的錯誤。 Type (類型)：the section called “GameLiftError” 必要：否

F GameLiftError

屬性	Description
ErrorType	錯誤類型。 類型：**GameLiftErrorType** 枚舉。 必要：否
ErrorName	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

針對 Amazon GameLift 伺服器開發套件 (虛幻) 定義的枚舉定義如下：

F AttributeType

- NONE
- 字符串
- 雙

- 字符串列表
- 弦雙地圖

GameLiftErrorType

指示錯誤類型的字符串值。有效值包含：

- 服務呼叫失敗 — 對服務的呼叫失敗。AWS
- 本地連接失敗 — 與 Amazon 的本地連接失敗。 GameLift
- 網路不初始化 — 網路尚未初始化。
- 遊戲設定 — 尚未設定遊戲工作階段識別碼。
- 不要求 _ 例外
- 內部 _ 服務 _ 異常
- 已初始化 — Amazon GameLift 伺服器或用戶端已經使用初始化 () 進行初始化。
- FLEET_MISMATCH — 目標叢集與遊戲經營或玩家工作階段的機隊不相符。
- 已初始化 — Amazon 用戶端尚未初始化。 GameLift
- 已初始化 — Amazon 伺服器尚未初始化。 GameLift
- 遊戲工作階段失敗 — Amazon GameLift 伺服器開發套件無法聯絡服務以報告遊戲工作階段已結束。
- 未啟動 Amazon 伺服器遊戲工作階段。 GameLift
- 遊戲工作階段已準備就緒 — Amazon GameLift 伺服器開發套件無法聯絡服務以報告遊戲工作階段已準備就緒。
- 初始化不匹配-在服務器:: 初始化（ ）之後調用客戶端方法，反之亦然。
- 初始化 — Amazon GameLift 伺服器或用戶端尚未使用初始化 () 初始化。
- 無目標別名-尚未設定目標別名。
- 目標叢集 — 尚未設定目標叢集。
- 處理結束-Amazon GameLift 伺服器開發套件無法聯絡服務以報告程序已結束。
- PROCES_NOT_ACT_ACTIVE — 伺服器處理作業尚未啟動、未繫結至 GameSession，且無法接受或處理。 PlayerSessions
- 處理程序 NOT_READY — 伺服器處理作業尚未準備好啟動。
- 處理程序 READY_FAIN — Amazon GameLift 伺服器開發套件無法聯絡服務以報告程序已準備就緒。

- SDK_ 版本偵測失敗 — SDK 版本偵測失敗。
- STX_CALL_ 失敗 — 對 XSTX 伺服器後端元件的呼叫失敗。
- STX_ 初始化失敗 — XSTX 伺服器後端元件無法初始化。
- 意外的玩家工作階段 — 伺服器遇到未註冊的玩家工作階段。
- WEB 插槽 _ 連接 _ 失敗
- WEB 插座 _ 連接 _ 失敗 _ 禁止
- 網站插槽 _ 連接 _ 失敗 _ 無效 _URL
- 網路插槽 _ 連線 _ 失敗 _ 逾時
- 網站重新擷取 _ 傳送訊息 _ 失敗 — 將訊息傳送至服務的可擷取失敗。 GameLift WebSocket
- 網站傳送訊息失敗 — 無法將訊息傳送至服務。 GameLift WebSocket
- 匹配 _ 後填 _ 請求驗證 — 驗證請求失敗。
- 播放器會話請求驗證 — 請求的驗證失敗。

E 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玩家工作階段。

E PlayerSessionStatus

- ACTIVE
- COMPLETED (已完成)
- 沒有設定
- 已保留
- 定時

亞馬遜GameLift虛幻引擎服務器 SDK 3.x 參考

您可以使用這個亞馬遜GameLift虛幻引擎服務器 SDK 3.x 參考來幫助您準備與亞馬遜一起使用的多人遊戲。GameLift如需有關整合程序的詳細資訊，請參閱添加亞馬遜GameLift到您的遊戲服務器。

主題

- 虛幻引擎的亞馬遜GameLift服務器 SDK 參考：操作
- 虛幻引擎的亞馬遜GameLift服務器開發套件參考：數據類型

虛幻引擎的亞馬遜GameLift服務器 SDK 參考：操作

這個亞馬遜GameLift伺服器開發套件參考可以幫助您準備虛幻引擎遊戲專案，以便與亞馬遜GameLift搭配使用。如需整合流程的詳細資訊，請參閱 添加亞馬遜GameLift到您的遊戲服務器。

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 GameLift 將 Amazon 集成到虛幻引擎項目中。

- 動作
- 資料類型

AcceptPlayerSession()

通知 Amazon GameLift 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並需要驗證。Amazon 會GameLift驗證玩家工作階段 ID 是否有效 — 也就是說，玩家 ID 已在遊戲工作階段中保留一個玩家位置。一旦驗證，亞馬遜GameLift將播放器插槽的狀態從保留更改為活動狀態。

語法

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

參數

playerSessionId

由亞馬遜GameLift服務發出的唯一 ID，以回應對 AWS SDK 亞馬遜 GameLift API 動作的呼叫CreatePlayerSession。遊戲客戶端在連接到服務器進程時引用此 ID。

類型：FString

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

ActivateGameSession()

通知 Amazon GameLift 服務伺服器處理序已啟動遊戲工作階段，現在已準備好接收玩家連線。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

傳回包含錯誤訊息的成功或失敗的一般結果。

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

describePlayerSessions請求

F 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 F DescribePlayerSessionsRequest 物件，內含一組與請求參數相符的玩家工作階段物件。播放器工作階段物件的結構與 AWS SDK Amazon GameLift 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()

擷取與叢集及其執行個體相關聯的 PEM 編碼 TLS 憑證的檔案位置。 AWS Certificate Manager當您建立新的叢集並將憑證組態設定為「已產生」時，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，會傳回包含叢集 TLS 憑證檔案和憑證鏈結位置的GetInstanceCertificateOutcome物件，這些檔案儲存在執行個體上。從憑證鏈結擷取的根憑證檔案也會儲存在執行個體上。如果不成功，則會傳回錯誤訊息。

如需有關憑證和憑證鏈結資料的詳細資訊，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。返回的字符串僅包含版本號（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()

初始化亞馬遜開GameLift發套件。在發生任何其他 Amazon GameLift 相關初始化之前，應在啟動時呼叫此方法。

語法

```
FGameLiftGenericOutcome InitSDK()
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

ProcessEnding()

通知 Amazon GameLift 服務伺服器處理序正在關閉。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

傳回包含錯誤訊息的成功或失敗的一般結果。

ProcessReady()

通知 Amazon GameLift 服務伺服器處理序已準備好主持遊戲工作階段。在成功叫用InitSDK()並完成伺服器處理序主控遊戲工作階段之前所需的設定工作之後，呼叫此方法。每個進程應該只調用一次此方法。

語法

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

參數

F ProcessParameters

F 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 服務呼叫以與伺服器處理序進行通訊。
- 伺服器程序正在接聽的埠號。
- 您希望 Amazon 擷取和存放的任何遊戲工作階段特定檔案GameLift的路徑。

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

範例

請在使用 Unreal Engine 外掛程式檢視範本程式碼。

RemovePlayerSession()

通知 Amazon GameLift 服務具有指定玩家工作階段 ID 的玩家已中斷與伺服器處理序的連線。作為回應，亞馬遜將播放器插槽GameLift更改為可用，這允許將其分配給新玩家。

語法

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

參數

playerSessionId

由亞馬遜GameLift服務發出的唯一 ID，以回應對 AWS SDK 亞馬遜 GameLift API 動作的呼叫CreatePlayerSession。遊戲客戶端在連接到服務器進程時引用此 ID。

類型：FString

必要：是

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。進一步瞭解FlexMatch回填功能。

此為非同步動作。如果成功配對新玩家，Amazon GameLift 服務會使用回調函`OnUpdateGameSession()`數提供更新的分房系統資料。

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

F StartMatchBackfillRequest

F StartMatchBackfillRequest 物件會傳達以下資訊：

- 指派給回填請求的票證 ID。此資訊是選擇性的；如果未提供 ID，Amazon GameLift 將自動產生一個 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() 來追蹤工單狀態。

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()。進一步瞭解FlexMatch回填功能。

語法

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

參數

StopMatchBackfillRequest

識別配對票證的 F 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

傳回包含錯誤訊息的成功或失敗的一般結果。

TerminateGameSession()

此方法已被版本 4.0.1 棄用。相反地，伺服器程序應該在遊戲工作階段結束ProcessEnding()後呼叫。

通知 Amazon GameLift 服務伺服器處理序已結束目前的遊戲工作階段。當伺服器處理程序保持作用中並準備好主持新遊戲工作階段時，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 Amazon 發出訊號GameLift，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關閉伺服器程序，則不會呼叫此動作。相反，ProcessEnding()請調用以表示遊戲會話和服務器進程都結束。

語法

```
FGameLiftGenericOutcome TerminateGameSession()
```

參數

此動作沒有參數。

傳回值

傳回包含錯誤訊息的成功或失敗的一般結果。

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。（另請參閱亞馬遜GameLift服務 API 參考中的UpdateGameSession()操作）。

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

傳回包含錯誤訊息的成功或失敗的一般結果。

虛幻引擎的亞馬遜GameLift服務器開發套件參考：數據類型

這個亞馬遜GameLift伺服器開發套件參考可以幫助您準備虛幻引擎遊戲專案，以便與亞馬遜GameLift搭配使用。如需整合流程的詳細資訊，請參閱 添加亞馬遜GameLift到您的遊戲服務器。

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 GameLift 將 Amazon 集成到虛幻引擎項目中。

- 動作
- 資料類型

F 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：

- 提供一個PlayerSessionId要求特定玩家工作階段。
- 提供一個GameSessionId要求指定遊戲工作階段中的所有玩家工作階段。
- 提供一個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。搭配使用此參數可NextToken取得一組連續頁面的結果。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 ID。

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

F ProcessParameters

此資料類型包含在ProcessReady()呼叫中傳送至 Amazon GameLift 服務的一組參數。

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

onStartGame階段

Amazon GameLift 服務呼叫以啟動新遊戲工作階段的回呼函數名稱。亞馬遜GameLift調用此函數以響應客戶端請求CreateGameSession。回調函數接受一個GameSession對象（在亞馬遜GameLift服務 API 參考中定義）。

類型:F OnStartGameSession

必要：是

onProcessTerminate

Amazon GameLift 服務呼叫以強制伺服器處理序關閉的回呼函數名稱。呼叫此函數之後，Amazon 會GameLift等待五分鐘讓伺服器處理序關閉並回應呼ProcessEnding()叫，然後再關閉伺服器處理序。

類型:F SimpleDelegate

必要：否

onHealthCheck

Amazon GameLift 服務呼叫以從伺服器處理序要求健康狀態報告的回呼函數名稱。亞馬遜每 60 秒GameLift調用一次此函數。調用此函數後，亞馬遜GameLift等待 60 秒的響應，如果沒有收到任何響應。將服務器進程記錄為不健康。

類型:F OnHealthCheck

必要：否

onUpdateGame階段

Amazon GameLift 服務呼叫以將更新的遊戲工作階段物件傳遞至伺服器處理序的回呼函數名稱。Amazon 會在處理比賽回填請求以提供更新的分房系統資料時GameLift呼叫此函數。它傳遞一個GameSession對象，一個狀態更新（`updateReason`）和匹配回填票證 ID。

類型:F OnUpdateGameSession

必要：否

F StartMatchBackfillRequest

此項資料類型用於傳送配對回填要求。該信息在通StartMatchBackfill()話中傳達給亞馬遜GameLift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。請參閱使用分房系統資料，進一步了解分房系統資料。

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。有關播放器對象格式的說明，請參閱 Amazon GameLift 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。如果這裡沒有提供任何值，亞馬遜GameLift將以 UUID 的形式生成一個值。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

F StopMatchBackfillRequest

此項資料類型用於取消配對回填要求。該信息在通StopMatchBackfill()話中傳達給亞馬遜GameLift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

遊戲工作階段安置事

Amazon 會在處理每個遊戲工作階段放置請求時 GameLift 發出事件。您可以將這些事件發佈到 Amazon SNS 主題，如中所述設定遊戲工作階段位置的事件通知。這些事件也會以近乎即時的方式發出至 Amazon CloudWatch 活動，並以最大的努力為基礎。

本主題說明遊戲工作階段放置事件的結構，並提供每個事件類型的範例。如需有關遊戲工作階段放置請求狀態的詳細資訊，請參閱 Amazon GameLift API 參考GameSessionPlacement中的。

放置事件語法

事件會以 JSON 物件的形式表示。事件結構符合 CloudWatch 事件模式，具有類似的頂層欄位和服務特定詳細資料。

頂級字段包括以下內容（有關更多詳細信息，請參閱事件模式）：

version

此欄位永遠設定為 0 (零)。

id

事件的唯一追蹤識別碼。

詳細資訊類型

價值始終是`GameLift Queue Placement Event`。

source

價值始終是aws.gamelift。

帳戶

用於管理 Amazon 的 AWS 帳戶 GameLift。

time

事件時間戳記。

region

正在處理放置請求的 AWS 區域。這是使用中遊戲工作階段佇列所在的區域。

resources

正在處理放置要求的遊戲工作階段佇列的 ARN 值。

PlacementFulfilled

放置要求已成功完成。新的遊戲工作階段已經開始，並且已經為遊戲工作階段要求中列出的每位玩家建立了新的玩家工作階段。玩家連接信息可用。

詳細語法：

安慰劑 ID

指派給遊戲工作階段放置要求的唯一識別碼。

port

新遊戲工作階段的連接埠號碼。

gameSessionArn

新遊戲工作階段的 ARN 識別碼。

ipAddress

遊戲工作階段的 IP 位址。

DNS 名稱

指派給執行新遊戲工作階段之執行個體的 DNS 識別碼。值格式會因執行遊戲工作階段的執行個體是否啟用 TLS 而有所不同。連線至啟用 TLS 的叢集上的遊戲工作階段時，玩家必須使用 DNS 名稱，而非 IP 位址。

具備 TLS 功能的叢集：。<unique identifier>.<region identifier>.amazongamelift.com

未啟用 TLS 的叢集：。ec2-<unique identifier>.compute.amazonaws.com

startTime

指出此要求放入佇列的時間戳記。

endTime

時間戳記指出此要求的完成時間。

gameSessionRegion

AWS 主控遊戲工作階段的艦隊區域。這對應於中的區域權杖GameSessionArn。

placedPlayerSessions

在遊戲工作階段位置請求中為每位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-123213123232412412312 3"
      }
    ]
  }
}
```

PlacementCancelled

通過呼叫 GameLift 服務而取消放置請求StopGameSessionPlacement。

詳細資料:

安慰劑 ID

指派給遊戲工作階段放置要求的唯一識別碼。

startTime

指出此要求放入佇列的時間戳記。

endTime

指出取消此請求的時間戳記。

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

在佇列的時間限制到期前，遊戲工作階段放置未順利完成。您可以視需要重新提交放置請求。

詳細資料:

安慰劑 ID

指派給遊戲工作階段放置要求的唯一識別碼。

startTime

指出此要求放入佇列的時間戳記。

endTime

指出取消此請求的時間戳記。

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 無法滿足遊戲會話請求。這通常是由意外的內部錯誤引起的。您可以視需要重新提交放置請求。

詳細資料:

安慰劑 ID

指派給遊戲工作階段放置要求的唯一識別碼。

startTime

指出此要求放入佇列的時間戳記。

endTime

指出此要求失敗的時間戳記。

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"
  }
}
```

生成亞馬遜GameLift定價估算

使用AWS Pricing Calculator，您可以為亞馬遜創建定價估算GameLift。您不需要AWS 帳戶或深入的知識即可AWS使用計算器。

AWS Pricing Calculator計算器會引導您完成影響服務成本的決策，讓您了解 Amazon GameLift 可能會為您的遊戲項目花費多少。如果您還不確定如何使用 AmazonGameLift，請使用預設值來產生估算值。在規劃生產使用時，計算器可以幫助您測試潛在情況並產生更準確的估計值。

您可以使用AWS Pricing Calculator來產生下列 Amazon GameLift 託管選項的估算值：

- 估計亞馬遜GameLift託管
- 估計亞馬遜GameLift獨立 FlexMatch

估計亞馬遜GameLift託管

此選項提供在 Amazon GameLift 受管伺服器上託管遊戲的成本估算，包括伺服器執行個體使用和資料傳輸的成本。FlexMatch配對功能已包含在 Amazon 託GameLift管託管的費用中。

如果您要託管或計劃在多個AWS區域或多個執行個體類型中託管遊戲伺服器，請為每個區域和執行個體類型建立估算值。

亞馬遜GameLift實例

本節可協助您估計為玩家主持遊戲工作階段所需的運算資源類型和數量。亞馬遜GameLift使用亞馬遜彈性運算雲（亞馬遜 EC2）實例來管理遊戲服務器。在 Amazon 中GameLift，您可以部署具有特定執行個體類型和作業系統的執行個體叢集。如果您已經或計劃擁有多個艦隊，請為每個叢集建立估算值。

若要開始使用，請開啟的設定 Amazon GameLift 頁面AWS Pricing Calculator。新增說明，選擇一個區域，然後選擇估計 Amazon GameLift 託管 (執行個體 + 資料傳出)。在 Amazon GameLift 執行個體下，完成下列欄位：

- 峰值並發球員（CCU 峰值）

  這是可以同時連接到遊戲伺服器的最大玩家數量。此字段指示 Amazon GameLift 需要多少託管容量才能滿足高峰玩家的需求。輸入您希望在所選AWS區域中使用執行個體託管的玩家每日尖峰數量。

  例如，如果您想讓 1,000 名玩家同時連線到您的遊戲，請保留預設值**1000**。

- 每小時的平均 CCU 佔每日高峰 CCU 的百分比

  這是 24 小時期間內每小時同時玩家的平均數量。我們使用這個值來估算 Amazon GameLift 需要為您的玩家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**百分比值。

  例如，如果您的遊戲的平均每小時 CCU 為 6,000，尖峰 CCU 為 10,000，則輸入百分比值。**60**

- 每個實例的遊戲會話

  這是每個遊戲伺服器執行個體可同時託管的遊戲工作階段數。可能會影響這個數字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接待的玩家人數，以及玩家的表現期望。如果您知道遊戲的同時遊戲工作階段數目，請輸入該值。或者，保留的預設值**20**。

- 每個遊戲階段的玩家

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

- 實例閒置緩衝區%

  這是未使用的託管容量的百分比，用於保留以處理玩家需求突然峰值。緩衝區大小是叢集中執行個體總數的百分比。預設值為**10**百分比。

  例如，如果有 20% 的閒置緩衝區，支援擁有 100 個作用中執行個體的玩家的叢集會維護 20 個閒置執行個體。

- 定點執行個體%

  Amazon GameLift 叢集可以使用隨需執行個體和 Spot 執行個體的組合。隨需執行個體提供更可靠的可用性，但 Spot 執行個體提供了高度經濟效益的替代 我們建議您使用組合來最佳化成本節約和可用性。如需 Amazon 如何GameLift使用 Spot 執行個體的相關資訊，請參閱隨需執行個體與 Spot 執行。

  在此欄位中，輸入要在叢集中維護的 Spot 執行個體百分比。我們建議使用 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 個執行個體的叢集並指定**40**百分比，Amazon 會GameLift努力維護 60 個隨需執行個體和 40 個 Spot 執行個體。

- 執行個體類型

  Amazon GameLift 叢集可以使用各種 Amazon EC2 執行個體類型，這些執行個體類型因運算能力、記憶體、儲存和聯網功能而異。設定 Amazon GameLift 叢集時，請選擇最符合您遊戲需求的執行個

體類型。如需使用 Amazon 選取執行個體類型的相關資訊GameLift，請參閱選擇 Amazon GameLift 運算資源。

如果您知道正在使用或計劃在 Amazon GameLift 叢集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

  此欄位指定您的遊戲伺服器執行的作業系統 (不論是 Linux 或 Windows)。預設值為 Linux。

資料傳出 (DTO)

本節可協助您估算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輸出流量。輸入資料傳輸不需要任何費用。

在的設定 Amazon GameLift 頁面上AWS Pricing Calculator，展開資料傳出 (DTO)，然後完成以下欄位：

- DTO 估計類型

  您可以選擇通過以下兩種方式之一估算 DTO，具體取決於跟踪遊戲數據傳輸的方式。

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

    在下面的字段中，您可以根據您在上一節中計算的玩家小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 (GB) DTO 估計類型，請輸入每個執行個體、每個區域的預估每月 DTO 用量 (以 GB 為單位)。

- 每位玩家的 DTO

  如果您選擇了「每位玩家 DTO」估計類型，請輸入遊戲中每位玩家的 DTO 估計使用量（KB/ 秒）。預設值為 **4**。

設定完 Amazon 估GameLift價後，請選擇 [新增至我的估價]。有關在中建立和管理預估的詳細資訊 AWS Pricing Calculator，請參閱《AWS Pricing Calculator使用者指南》中的建立估計值、設定服務以及新增更多服務。

估計亞馬遜GameLift獨立 FlexMatch

此選項提供將FlexMatch配對作為獨立服務使用的成本估算，同時以其他遊戲伺服器解決方案代管您的遊戲。這包括使用 FleetIQ 和現場部署託GameLift管的 Amazon 自我管理託管peer-to-peer，或雲端運算原始資料類型。獨立式FlexMatch成本取決於所使用的運算能力。

如果您已經或計劃在不同區AWS域中有多個分房系統，請為每個區域建立一個估算值。

> **Note**
>
> Amazon GameLift FlexMatch 在以下區域提供：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (首爾)、亞太區域 (雪梨)、亞太區域 (東京)、歐洲 (法蘭克福)、歐洲 (愛爾蘭)。

若要開始使用，請開啟的設定 Amazon GameLift 頁面AWS Pricing Calculator。新增說明，選擇一個區域，然後選擇估算 Amazon GameLift 獨立版FlexMatch。在 Amazon 下 GameLiftFlexMatch，完成下列欄位：

- 峰值並發球員（CCU 峰值）

  這是可以同時連接到您的遊戲伺服器並要求配對的玩家人數上限。輸入您希望在所選區域中與遊戲階段中匹配的每日玩家峰值數量。

  例如，如果您想要一次比對多達 1,000 名玩家，請保留預設值**1000**。
- 每小時的平均 CCU 佔每日高峰 CCU 的百分比

  這是 24 小時期間內每小時同時玩家的平均數量。此值有助於估計配對要求的數量。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**百分比值。

  例如，如果您的遊戲的平均每小時 CCU 為 6,000，尖峰 CCU 為 10,000，則輸入百分比值。**60**
- 每場比賽的玩家人數

  根據您的遊戲設計所定義，這是與某個遊戲工作階段相符的平均玩家人數。如果您有不同玩家人數的遊戲模式，請估計整個遊戲中每個遊戲工作階段的平均玩家人數。預設值為 **8**。
- 遊戲時長（分鐘）

  這是玩家從頭到尾停留在遊戲階段中的平均時間長度。此值有助於決定玩家需要新配對的頻率。輸入玩家的平均遊戲持續時間 (以分鐘為單位)。預設值為 **1**。
- 配對規則複雜性

配對規則複雜度是指您用來配對玩家的規則數量和類型。規則集的複雜程度有助於判斷每個相符項目所需的運算能力。

- 較低的複雜性 — 如果您的配對規則集包含少量規則、使用較簡單的規則類型 (例如比較規則)，且具有成功比對且嘗試次數較少的規則，請選擇此選項。
- 較高的複雜性 — 如果配對規則集包含多個規則、使用較複雜的規則類型 (例如距離或延遲規則)，並具有可導致更多失敗且需要更多比對嘗試次數的限制性規則，請選擇此選項。

有關規則複雜性和定價的詳細資訊，請參閱 Amazon GameLift 定價頁面GameLiftFlexMatch上的Amazon。

設定完 Amazon 估GameLiftFlexMatch價後，請選擇 [新增至我的估價]。有關在中建立和管理預估的詳細資訊AWS Pricing Calculator，請參閱《AWS Pricing Calculator使用者指南》中的建立估計值、設定服務以及新增更多服務。

配額和支援的區域

有關AWS亞馬遜GameLift服務配額，請參閱亞馬遜GameLift配額。

如需請求增加資源配額的相關AWS資訊，請參閱AWS服務 l quota。

有關AWS 區域支持亞馬遜的列表GameLift，請參閱亞馬遜GameLift區域。

Amazon GameLift 版本說明

Amazon GameLift 版本說明提供與服務相關的新功能、更新和修正的詳細資訊。

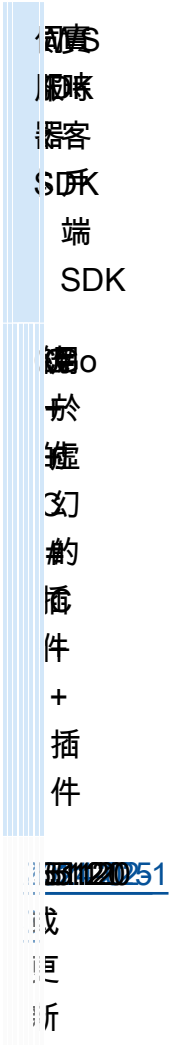
軟體開發套件版本

下表列出了包含 SDK 版本資訊的所有 Amazon GameLift 版本。您的遊戲伺服器和用戶端整合不需要使用可比較的 SDK。但是，某個 SDK 的早期版本可能無法完全支援另一個 SDK 中的最新功能。

如需 Amazon GameLift 開發套件的詳細資訊，請參閱Amazon 的開發支持 GameLift。

若要取得最新的 Amazon GameLift 開發套件，請參閱 Amazon GameLift 軟體開發套件下載網站。

目前版本



舊版本

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2023-12-14	1.11.225 或更新版本	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2023-11-02	1.11.193 或更新版本	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-09-28	1.11.144 或更新版本	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-08-17	1.11.144 或更新版本	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-07-27	1.11.111 或更新版本	5.1.0	5.1.0	5.0.2	5.0.0	1.2.0
2023-06-29	1.11.111 或更新版本		5.0.4	5.0.2	5.0.0	1.2.0

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2023-06-15	1.11.87 或更新版本		5.0.4	5.0.2	5.0.0	1.2.0
2023-05-25	1.11.87 或更新版本		5.0.3	5.0.2	5.0.0	1.2.0
2023-04-20	1.11.63 或更新版本				5.0.0	1.2.0

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2023-04-13	1.10.21 或更新版本				5.0.0	1.2.0
2023-02-09	1.10.21 或更新版本			3.4.0	5.0.0	1.2.0
2023-01-31	1.10.21 或更新版本			3.4.0	5.0.0	1.2.0
2022-12-01	1.10.21 或更新版本			3.4.0		1.2.0
2022-08-25	1.9.333 或更新版本		3.4.2	3.4.0		1.2.0

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2021-10-28	1.9.133 或更新版本		3.4.2	3.4.0		1.2.0
2021-06-03	1.8.168 或更新版本		3.4.2	3.4.0		1.2.0
2021-03-23	1.8.168 或更新版本		3.4.1	3.3.3		1.1.0
2021-03-16	1.8.163 或更新版本		3.4.1	3.3.3		1.1.0
2021-02-09	1.8.139 或更新版本		3.4.1	3.3.3		1.1.0
2020-12-22	1.8.95 或更新版本		3.4.1	3.3.3		1.1.0
2020-11-24	1.8.95 或更新版本		3.4.1	3.3.2		1.1.0
2020-11-11	1.8.36 或更新版本		3.4.1	3.3.2		1.1.0
2020-09-17	1.8.36 或更新版本		3.4.1	3.3.2		1.1.0
2020-08-27	1.7.310 或更新版本		3.4.0	3.3.1		1.1.0

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2020-04-16	1.7.310 或更新版本		3.4.0	3.3.1		1.1.0
2020-04-02	1.7.310 或更新版本		3.4.0			1.1.0
2019-12-19	1.7.249 或更新版本		3.4.0			1.1.0
2019-11-14	1.7.210 或更新版本		3.4.0			1.1.0
2019-10-24	1.7.210 或更新版本		3.4.0			1.1.0
2019-09-03	1.7.175 或更新版本		3.4.0			1.1.0
2019-07-09	1.7.140 或更新版本		3.3.0			1.0.0
2019-04-25	1.7.91 或更新版本		3.3.0			1.0.0
2019-03-07	1.7.65 或更新版本		3.3.0			
2019-02-07	1.7.45 或更新版本		3.3.0			

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2018-12-14	1.6.20 或更新版本		3.3.0			
2018-09-27	1.6.20 或更新版本		3.2.1			
2018-06-14	1.4.47 或更新版本		3.2.1			
2018-05-10	1.4.47 或更新版本		3.2.1			
2018-02-15	1.3.58 或更新版本		3.2.1			
2018-02-08	1.3.52 或更新版本		3.2.0			
2017-09-01	1.1.43 或更新版本		3.1.7			
2017-08-16	1.1.31 或更新版本		3.1.7			
2017-05-16	1.0.122 或更新版本		3.1.5			
2017-04-11	1.0.103 或更新版本		3.1.5			

服務版本	AWS SDK	伺服器 SDK				實時客戶端 SDK
		統一的 C# 插件	C++	用於虛幻的 C++ 插件	Go	
2017-02-21	1.0.72 或更新版本		3.1.5			
2016-11-18	1.0.31 或更新版本		3.1.0			
2016-10-13	1.0.17 或更新版本		3.1.0			
2016-09-01	0.14.9 或更新版本		3.1.0			
2016-08-04	0.12.16 或更新版本		3.0.7			

版本備註

下列版本說明依時間先後順序排列，最新更新會先列出。Amazon GameLift 於 2016 年首次發布。如需早於此處列出的版本備註，請參閱軟體開發套件版本中的發行日期連結。

2024 年 4 月 24 日：Amazon GameLift 推出集裝箱艦隊

Amazon 現 GameLift 在提供容器叢集的預覽功能，可讓您改善可攜性、可擴展性、容錯能力和靈活性。

在容器叢集中，Amazon EC2 執行個體託管一或多個容器。這些容器包括您的遊戲伺服器以及所需的任何內容，包括相依性和設定。相依性的範例包括 SDK 和軟體套件。將容器上傳到私有 Amazon 彈性容器註冊表後，Amazon 會將容器 GameLift 植入您的叢集。

若要在容器叢集中運作，您的遊戲伺服器必須在 Linux 中執行，並與伺服器 SDK 5.x 整合。在容器叢集中，您可以微調主機資源的控制權，以便最佳化 CPU 單元和記憶體等資源消耗。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。

在容器叢集中，您可獲得與其他叢集類型相同的許多優點，例如隨需執行個體類型、擴展（自動和手動）、佇列和配對。您也會取得與其他叢集類型相同的指標，以及一些容器的新指標。貨櫃車隊可讓您觸及以下地區的玩家遍及全球：

- 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地區域，請建立多位置容器叢集。

進一步了解：

- 使用 Amazon GameLift 容器管理託管，Amazon GameLift 開發人員
- CreateContainerGroupDefinition，Amazon GameLift API 參考

2024 年 2 月 13 日：Amazon GameLift 推出對 SDK 的改進，並簡化了虛幻引擎 Amazon GameLift 插件的安裝

更新的 SDK 版本：

- 轉到服務器 SDK，版本 5.1.0
- C# 服務器開發套件，版本 5.1.2
- C ++ 服務器開發套件，版本 5.1.2

我們做了以下改進：

- 通過在網絡中斷時添加自動重新連接來提高 SDK 的可靠性。

- [Go] 您現在可以`InitSDK()`使用或不使用服務器參數進行調用。在 Amazon GameLift 受管 EC2 叢集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 Anywhere 叢集上的遊戲伺服器必須`InitSDK()`使用伺服器參數呼叫。

更新插件版本：

- Amazon GameLift 插件虛幻引擎，版本 1.1.0
- Amazon GameLift 插件統一，版本 2.1.0
- 用於虛幻的 C ++ 服務器 SDK 插件，版本 5.1.1
- 用於統一的 C# 服務器 SDK 插件，版本 5.1.2

我們做了以下改進：

- [虛幻引擎的 Amazon GameLift 插件] 更新了安裝說明並簡化了包裝。這個外掛程式現在包含最新版本的 C++ 伺服器 SDK 的虛幻。
- 升級了插件以支持最新版本的 GameLift 服務器 SDK。

進一步了解：

- 將遊戲與虛幻引擎的 Amazon GameLift 插件集成，Amazon GameLift 開發人員指南
- Amazon GameLift 插件和 SDK 下載

2023 年 12 月 14 日：Amazon GameLift 增加了更新活動遊戲會話遊戲屬性的功能

您已經可以在建立遊戲工作階段時設定遊戲屬性，以及在遊戲工作階段中搜尋指定的屬性。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遊玩的地圖上投票。您的遊戲用戶端呼叫`UpdateGameSession`將`GameProperty`值修改為`{"Key": "map", "Value":"jungle"}`。然後，您的遊戲會在遊戲工作階段中為玩家實作新地圖。

遊戲管理員還可以通過使用`SearchGameSessions`操作從遊戲屬性中檢索有用的數據。例如，管理員可以列出`Status`值為的遊戲工作階段，`ACTIVE`且此遊戲屬性為：`{"Key": "map", "Value":"desert"}`。

進一步了解：

- the section called “將 Amazon 添加 GameLift 到遊戲客戶端”Amazon GameLift 開發人員指南
- GameProperty，Amazon GameLift API 參考
- UpdateGameSession，Amazon GameLift API 參考
- SearchGameSessions，Amazon GameLift API 參考

2023 年 11 月 21 日：Amazon GameLift 推出對基礎設施作為代碼工具的支持，例如 Terraform 和普魯米 AWS Cloud Control 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 資源堆疊。這些工具還包括第三方工具 AWS CloudFormation，例如地形和普魯米。有了這項額外的支援，您現在可以專注於建置遊戲，並運用 DevOps 策略來處理資源管理、CI/CD 和部署給客戶。

您現在也可以使用 AWS 雲端控制 API 佈建和設定所有 Amazon GameLift 資源類型。您可以繼續使用 Amazon GameLift API 或 Amazon 的 AWS CloudFormation 模板使用資源 GameLift。

有關可透過 IaC 取得的 Amazon GameLift 資源的詳細資訊，請參閱 Amazon GameLift 資源類型參考 Amazon GameLift 資源類型參考。

此外，您現在可以使用新的叢集屬性，使用 AWS CloudFormation 範本或 AWS Cloud Control API 來自動調整叢集的規模：`ScalingPolicies`。

雲端控制 API 為開發人員提供一組標準的 API，以建立、讀取、更新、刪除和列出數百個 AWS 服務和多個第三方工具 (例如 Terraform 和 Plumi) 的資源 (CRUDL)。

進一步了解：

- AWS CloudFormation
- AWS 雲端控制 API
- AWS CC 地形表單提供者
- 普盧米

2023 年十一月 16 日：Amazon GameLift 更新 Unity 的獨立插件

更新 SDK 版本：Amazon GameLift 插件統一，2.0.0 版

用於 Unity 的 Amazon GameLift 插件提供了工具和工作流程，簡化了讓您的 Unity 遊戲啟動和運行雲託管與 Amazon 的步驟 GameLift。Amazon GameLift 是一項全受管服務，可讓遊戲開發人員管理和擴展工作階段型多人遊戲的專用遊戲伺服器。

使用此版本，Unity 的插件將更新為使用最新的 Amazon GameLift 功能，包括服務器 SDK 版本 5.x 和對 Amazon GameLift 任何地方進行本地測試的支持。該插件是與統一版本統一 2021.3 LTS 和 2022.3 LTS 兼容。

主要插件功能包括：

- Unity 編輯器中的引導式 UI 工作流程，適用於下列案例：
  - GameLift 使用本機工作站做為主機測試您與 Amazon 的遊戲整合。此工作流程可協助您為本機電腦設定 Amazon GameLift Anywhere 叢集、啟動遊戲伺服器和用戶端的執行個體、透過 Amazon 請求遊戲工作階段 GameLift，以及加入遊戲。
  - 使用 Amazon GameLift 受管 EC2 和支援 AWS 資源，為整合式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主機設定遊戲，並提供三種部署方案：
    - 將遊戲伺服器部署至單一叢集。
    - 將遊戲伺服器部署到多個 AWS 區域中的一組低成本競價型車隊。
    - 使用 FlexMatch 分房系統部署遊戲伺服器。
- 能夠設置鏈接到帳戶用戶的用 AWS 戶配置文件並設置默認 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 整合和部署程序的特殊便利設施，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 ID 和玩家驗證的 Amazon Cognito 使用者集區。這些解決方案還包括用於儲存的 Amazon S3 儲存貯體、Amazon SNS 事件通知、AWS Lambda 功能和其他資源。
  - 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 Amazon GameLift 服務之間，並管理對 Amazon 服 GameLift務的所有直接呼叫。
- 用於集成測試的內容，包括用於簡單示例多人遊戲的資產和代碼，用於說明遊戲服務器和遊戲客戶端集成。
- 插件文檔與詳細的集成指導和示例代碼。

所有部署案例 (包括Anywhere和 Amazon EC2 叢集) 都使用 AWS CloudFormation 範本來描述和部署遊戲解決方案的 AWS 資源。這些模板包含在 Amazon GameLift 插件下載。您可以按原樣使用它們，也可以為您的遊戲自定義它們。

進一步了解：

- Amazon GameLift 插件統一指南服務器 SDK 5.xAmazon GameLift 開發人員指南
- 從以下位置下載插件 GitHub
- 關於 Amazon GameLift 託管
- Amazon GameLift 論壇

2023 年 11 月 2 日：Amazon GameLift 增加了對共享登入資料的支援

更新的開發套件版本： AWS 軟體開發套件

新的 Amazon GameLift 共用登入資料功能可讓部署在受管 EC2 叢集上的應用程式與其他 AWS 資源互動。此更新會影響您捆綁和部署的應用程式，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()` 動作來要求認證。)

例如，如果您想要使用 Amazon CloudWatch 代理部署遊戲伺服器組建以收集 EC2 執行個體指標和其他資料，則代理程式需要與您的 CloudWatch 資源互動的許可。若要這麼做，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叢集。Amazon 將遊戲伺服器組建 GameLift 部署到每個 EC2 執行個體時，會產生共用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認證。Amazon GameLift 會在執行個體生命週期內自動重新整理臨時登入資料。

您可以在使用下列方法建立受管 EC2 叢集時啟用共用登入資料：

- 在 Amazon 主 GameLift 控台叢集建立工作流程中。
- 使用新參數調`CreateFleet`用 Amazon GameLift 服務 API 操作時`InstanceRoleCredentialsProvider`。
- `aws gamelift create-fleet`使用參數呼叫 AWS CLI 作業時`instance-role-credentials-provider`。

進一步了解：

- Amazon GameLift 開發人員指南與叢集的其他 AWS 資源進行通訊

- CreateFleet, InstanceRoleCredentialsProvider, Amazon GameLift API 參考
- 設定 IAM 服務角色, Amazon GameLift 開發人員指南

2023 年 9 月 28 日：Amazon 為虛幻引擎 GameLift 發布新的獨立插件

更新 SDK 版本：Amazon GameLift 插件虛幻引擎版本 1.0.0

適用於虛幻引擎的 Amazon GameLift 外掛程式提供工具和工作流程，可簡化您的步驟，讓您透過 Amazon 啟動並執行遊戲進 GameLift 行雲端託管。Amazon GameLift 是一項全受管服務，可讓遊戲開發人員管理和擴展工作階段型多人遊戲的專用遊戲伺服器。該插件支持用戶界面 5.0，5.1 和 5.2 版本。主要特色包括：

- 虛幻編輯器中的引導式 UI 工作流程] 逐步執行以下路徑：
  - GameLift 使用本機工作站做為主機測試您與 Amazon 的遊戲整合。此工作流程可協助您為本機電腦設定 Amazon GameLift Anywhere 叢集、啟動遊戲伺服器和用戶端的執行個體、透過 Amazon 請求遊戲工作階段 GameLift，以及取得新遊戲工作階段的連線資訊。
  - 為您的整合式遊戲伺服器部署 Amazon EC2 雲端託管解決方案。此工作流程可協助您為雲端主機設定遊戲，並提供三種不同的部署案例：部署至單一叢集、部署到多個地區的一組 Spot 叢集，或使用分房系統部署至一 FlexMatch 組叢集。每個部署案例的解決方案包括 Amazon GameLift 資源和支援 AWS 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 整合和部署程序的特殊便利設施，包括：
  - 每個託管解決方案都包括支援 AWS 資源，包括提供唯一播放器 ID 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 功能。
  - 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使用 Lambda 函數提供內建的用戶端後端服務。後端服務會接收來自遊戲用戶端的請求，並將其傳遞至 Amazon GameLift 服務。
- 用於整合測試的內容，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。
- 插件文檔與詳細的集成指導和示例代碼。

所有部署案例 (包括適用於Anywhere和 Amazon EC2 叢集) 都會使用 AWS CloudFormation 範本來描述解決方案。該插件在為您的遊戲部署 Amazon GameLift 資源時使用這些模板。這些模板包含在 Amazon GameLift 插件下載中，並且可以編輯。您可以按原樣使用它們，也可以為您的遊戲修改它們。

進一步了解：

- 將遊戲與虛幻引擎的 Amazon GameLift 插件集成Amazon GameLift 開發人員指南
- 從以下位置下載插件 GitHub
- 關於 Amazon GameLift 託管
- Amazon GameLift 論壇

2023 年 8 月 17 日：Amazon GameLift 提供帶有 AWS 重力子處理器的遊戲服務器託管

更新的開發套件版本： AWS 軟體開發套件

透過 Amazon GameLift ，您現在可以使用具有 AWS Graviton 處理器的 EC2 執行個體在雲端託管遊戲。Graviton 執行個體採 AWS 用 ARM64 處理器設計，可為使用 EC2 的雲端工作負載提供最優惠的價格效能，相較於同類 x86 執行個體，最多可提升 40%。與舊版相比，最新的 Graviton3 處理器提供高達 25% 的運算效能。

透過 Amazon GameLift，您現在可以從 AWS 引力子系列中的這些新執行個體中進行選擇：

- 以重力翁為基礎的執行個體：c6 公克、六分之六公克、R6 公克、m6 公克
- 以重力翁 3 為基礎的執行個體：c7 公克、r7 公克、m7 克

進一步了解：

- AWS 重力發處理器：了解以重力為基礎的 EC2 執行個體的優點和實際用途。
- 開始使用 Graviton：取得以重力為基礎的執行個體概觀，並深入瞭解應用程式如何依據作業系統、語言和執行時間執行。

> **Note**
>
> 重力臂執行個體需要在 Linux 作業系統上建置 Amazon GameLift 伺服器。C ++ 和 C # 需要服務器 SDK 5.1.1 或更新版本。轉到需要服務器 SDK 5.0 或更新版本。這些執行個體不 out-of-the-box 支援 Amazon Linux 2023 (AL2023) 或 Amazon Linux 2 (AL2) 上的單聲道安裝。

2023 年 7 月 27 日：Amazon GameLift 發布服務器 SDK 5.1.0，並增加了對統一開發的支持

更新 SDK 版本：服務器 SDK for C++，C＃/統一，虛幻 5.1.0

Amazon GameLift 服務器 SDK 的最新版本提供了 C ++，C＃ 和虛幻插件的更新，以及與 Unity 遊戲引擎一起使用的新插件。遊戲開發人員將 Amazon GameLift 伺服器 SDK 整合到他們部署以便在 Amazon 上託管的遊戲伺服器中 GameLift。

最新的伺服器 SDK 版本包含下列更新，其中包括許多客戶要求：

- 下載特定語言的 SDK 套件 — 更新後的 Amazon GameLift 下載網站包含每種語言的 SDK 套件。您可以下載當前或以前的版本。
- 新 C# 伺服器 SDK 外掛程式的統一 — 新的伺服器 SDK 套件的統一包含內建 C# 程式庫，您可以使用 Unity 編輯器中的套件管理員安裝 (請參閱新的統一整合指南)。這些庫通過包括所需的依賴關係 UnityNuGet。您可以將此外掛程式與統一 2020.3 LTS、2021.3 LTS 和 2022.3 LTS 一起使用，適用於視窗和 Mac 作業系統。它支持統一的 .NET 框架和 .NET 標準配置文件，與 .NET 標準 2.1 和 .NET 4.x。
- C# 的整合式 .NET 解決方案 — C# 的伺服器 SDK 現在支援 .NET 框架 4.6.2 (從 4.6.1 升級) 和 .NET 6.0 在單一解決方案中。.NET 標準 2.1 可與單位構建的庫一起使用。
- 伺服器開發套件 5.1.0 更新
  - [C++，C＃ ，虛幻] 您現在可以`InitSDK()`使用或不使用服務器參數進行調用。在 Amazon GameLift 受管 EC2 叢集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 Anywhere 叢集上的遊戲伺服器必須`InitSDK()`使用伺服器參數呼叫。
  - [C ++，C＃ ，虛幻] 服務器 SDK 調用改進了錯誤消息傳遞。
  - [C++ SDK] 為了改善服務器 SDK 構建時間，默認情況下禁`-DRUN_CLANG_FORMAT`用構建標誌。您可以使用啟用它`-DRUN_CLANG_FORMAT=1`。
  - [C++ SDK] 在沒有標準庫（`-DGAMELIFT_USE_STD=0`）的情況下構建庫時，不`InitSDK()`再使用`std::`數據類型。
- 擴充伺服器 SDK 5.x 文件
  - 更新了 C ++，C＃/ 統一和虛幻的服務器 SDK 參考指南，包括擴展了所有數據類型的涵蓋範圍。
    - 用於 C＃ 和統一的亞馬遜GameLift服務器 SDK 5.x 參考
    - 亞馬遜GameLift服務器 SDK 5.x 參考 C ++
    - 亞馬遜GameLift虛幻引擎服務器 SDK 5.x 參考
  - 服務器 SDK 的新版本 5 集成指南統一和虛幻插件

    - GameLift 將亞馬遜整合到統一項目中
    - GameLift 將 Amazon 集成到虛幻引擎項目中
- 其他文件更新
    - 針對 Amazon GameLift 服務 API 操作的修訂文件，GetComputeAccess並GetInstanceAccess根據使用中的 Amazon GameLift 伺服器 SDK 版本釐清遠端存取程序。
    - 修訂說明，GameSessionPlacement以記錄放置位置處於「擱置中」狀態時，遊戲工作階段資訊是如何暫時的。

2023 年 7 月 13 日：Amazon GameLift 新增車隊硬體指標

您現在可以追蹤 Amazon GameLift 受管 EC2 叢集的硬體效能指標。指標包括用於 CPU 使用率的 EC2 執行個體指標、網路流量和磁碟讀取/寫入活動。對於 Amazon GameLift，這些指標描述叢集位置中的所有作用中執行個體。您可以使用中的 Amazon CloudWatch 儀表板檢視這些叢集硬體指標 AWS Management Console。您也可以在 Amazon GameLift 主控台中的叢集詳細資訊中檢視它們。

進一步了解：

- 監控亞馬遜GameLift與亞馬遜 CloudWatch（艦隊指標），Amazon GameLift 開發人員指南

2023 年 6 月 29 日：Amazon GameLift 推出對 Amazon Linux 的支持 2023

更新的開發套件版本： AWS 軟體開發套件

Amazon GameLift 客戶現在可以使用 Amazon Linux 2023 操作系統來託管他們的遊戲服務器。AL2023 比 AL2 提供了一些改進，包括安全性。此作業系統在所有地區均可使 AWS 區域 用，但中國地區除外。

當 Amazon Linux (AL1) 的支援於 2023 年 12 月終止時，客戶可以使用較新的 Linux 作業系統，並繼續收到重要的安全性更新。直到 2025 年，對 Amazon Linux 2 的 Support 將繼續。

進一步了解：

- Amazon GameLift Linux 服務器常見
- 比較 Amazon Linux 2 和 Amazon Linux 2023
- Amazon GameLift API 參考鏈接：
    - AWS SDK 動作 `CreateBuild`
    - CLI 命令 `upload-build`

- CLI 命令 `create-build`

2023 年 5 月 25 日：Amazon GameLift FleetIQ 新增篩選器，以排除耗盡執行個體上的遊戲工作階段展示位置

更新的開發套件版本： AWS 軟體開發套件

如果您使用 Amazon GameLift FleetIQ 進行遊戲託管，您現在可以防止在目前耗盡的執行個體上放置遊戲工作階段。排空執行個體會被標記為關機，但如果沒有其他主機資源可用，仍然可以選取這些執行個體來託管新的遊戲工作階段。使用此新功能，您可以完全排除排水例證的使用。

撥打電話`ClaimGameServer`尋找可用的遊戲伺服器時，請使用此功能。新增`FilterOption`參數，並僅將允許的例證狀態設定為 ACTIVE。作為回應，Amazon GameLift FleetIQ 只會在搜尋和聲明可用的遊戲伺服器時查看作用中執行個體。

進一步了解：

- ClaimGameServer在 Amazon GameLift API 參考
- FleetIQ 如何在 Amazon FleetIQ 開發人員指南 GameLift 中運作

2023 年 5 月 16 日：Amazon GameLift 支持車隊的成本分配標記

Amazon GameLift 客戶現在可以使用 AWS Billing 成本分配標籤來整理他們的遊戲託管成本。您可以將成本分配標籤指派給個別 Amazon GameLift EC2 叢集資源，以追蹤叢集對整體託管成本的貢獻。

進一步了解：

- 管理您的遊戲託管成本
- 使用 AWS 成本分配標籤, 使用AWS Billing 者指南

2023 年 4 月 20 日：Amazon GameLift 推出對視窗服務器 2016 年的支持

更新的開發套件版本： AWS 軟體開發套件

Amazon GameLift 客戶現在可以使用視窗伺服器 2016 年作業系統來託管他們的遊戲伺服器。該操作系統在所有可用 AWS 區域。客戶可以使用較新的 Windows 作業系統，並繼續收到重要的安全性更新，因為 Microsoft 於 2023 年 10 月終止對 Windows 伺服器 2012 的支援。

從今天開始，需要 Windows 執行階段環境的新客戶在建立新的遊戲伺服器組建以進行裝載時，必須指定 Windows 伺服器 2016 年。現有的客戶可以繼續使用 Windows 伺服器 2012 年建立新的組建和叢集，但必須在 Microsoft 於 2023 年 10 月 10 日結束支援日期之前完成移轉。

此更新包含下列服務變更：

- 使用 Amazon GameLift 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。若要在視窗伺服器 2016 上部署您的遊戲伺服器，請使用值`WINDOWS_2016`。
- 使用 Amazon GameLift 主控台建立遊戲伺服器組建時，您必須從可用值中選取作業系統。如果您是擁有作用中 Windows 伺服器 2012 年叢集的現有客戶，您可以選擇`WINDOWS_2012`或`WINDOWS_2016`。

進一步了解：

- Amazon GameLift API 參考鏈接：
  - CLI 命令 `upload-build`
  - CLI 命令 `create-build`
  - AWS SDK 動作 `CreateBuild`
- 視窗 2012 的 Amazon GameLift 常見問題

2023 年 4 月 13 日：Amazon 為虛幻 GameLift 啟動服務器 SDK 5.x

更新 SDK 版本：虛幻的服務器 SDK 5.0.0

虛幻引擎的 Amazon GameLift 輕量級插件的最新版本現在基於 Amazon GameLift 服務器 SDK 5.x。要開始將虛幻引擎環境與 Amazon 集成， GameLift 請參閱以下鏈接。

進一步了解：

- GameLift 將 Amazon 集成到虛幻引擎項目中
- 添加亞馬遜GameLift到您的遊戲服務器
- 亞馬遜GameLift服務器 SDK 5.x 參考 C ++

2023 年 3 月 14 日：Amazon GameLift 推出全新的主控台體驗

全新的 Amazon GameLift 主控台包含下列改良功能：

- 改進的導航 — 新的導航窗格有助於在 Amazon GameLift 資源之間進行導航。
- Amazon GameLift 登陸頁面 — 新的登陸頁面提供實用文件的連結、顯示 Amazon 的高階概觀 GameLift，並透過文件、常見問題和連結提供支援 AWS re:Post。
- 改進的 Amazon CloudWatch 指標 — Amazon GameLift 指標現在可在 Amazon GameLift 控制台和 CloudWatch儀表板中使用。此更新還包括效能、使用率和玩家工作階段的新指標。

進一步了解：

- 在主機中檢視遊戲資料
- 管理亞馬遜GameLift託管資源
- 建立 FlexMatch 媒人

2023 年 2 月 14 日：Amazon GameLift 現在支持 Amazon SNS 主題的服務器端加密

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：

- 設定遊戲工作階段位置的事件通知
- FlexMatch配對活動
- 靜態加密

2023 年 2 月 9 日：Amazon GameLift 服務器開發套件支持 .NET 6 與 C #10

更新 SDK 版本：適用於 .NET 6 的服務器 SDK 5.0.0。不需要 SDK 更新。

如果您使用統一即時開發平台，請繼續使用 Amazon GameLift 伺服器 SDK 5.0.0 與 .NET 4.6。團結不支援 .NET 6。

進一步了解：

- 在 Amazon 下載最新版本的 Amazon GameLift 服務器 SDK GameLift 入門
- 用於 C # 和統一的亞馬遜GameLift服務器 SDK 5.x 參考

2023 年 1 月 31 日：Amazon GameLift 服務器開發套件支持 Go 語言

更新 SDK 版本：適用於圍棋的服務器 SDK 5.0.0

進一步了解：

- 在 Amazon 下載最新版本的 Amazon GameLift 服務器 SDK GameLift 入門
- 適用於圍棋的亞馬遜GameLift服務器 SDK

2022 年 12 月 1 日：Amazon GameLift 推出 Amazon GameLift Anywhere 和 Amazon GameLift 服務器 SDK 5.0

更新的 SDK 版本： AWS SDK 1.10.21，用於 C ++ 和 C # 的服務器 SDK 5.0.0

Amazon GameLift Anywhere 使用您的遊戲服務器資源託管 Amazon GameLift 遊戲服務器。您可以使用 Amazon GameLift Anywhere 將自己的運算資源與 Amazon GameLift 受管 EC2 運算整合，以便將遊戲伺服器分配到多個運算類型。您還可以使用 Amazon GameLift Anywhere 迭代測試遊戲服務器，而無需每次迭代都將構建上傳到 Amazon GameLift 。

亮點：

- 全新的 Amazon GameLift Anywhere 叢集和運算類型
- Amazon GameLift Anywhere 運算資源註冊
- 改進的測試迭代週期

Amazon GameLift 服務器 SDK 5.0.0 引入了對現有服務器開發套件和新的資源類型運算的改進。服務器 SDK 5.0.0 支持 Amazon， GameLift Anywhere並使用您自己的計算資源進行遊戲服務器託管。

進一步了解：

- 亞馬遜GameLift服務器 SDK 參考
- 車隊位置
- 選擇 Amazon GameLift 運算資源
- 創建一個 Amazon GameLift Anywhere 車隊

2022 年 8 月 25 日：Amazon GameLift 推出對 Local Zones 的支持

更新的開發套件版本： AWS 開發套件 1.9.333

Amazon 現 GameLift 已在美國的八個 Local Zones 推出，因此您可以將車隊部署到更靠近玩家的地方。您可以將 Local Zones 新增至叢集，將所有受管 Amazon GameLift 功能與 Local Zones 搭配使用。

Local Zones 將 AWS 資源和服務延伸到雲端邊緣，鄰近大型人口、工業和資訊技術 (IT) 中心。這表示您可以將延遲時間需要 10 毫秒的應用程式部署到較接近使用者或內部部署資料中心的應用程式。

進一步了解：

- 本機區域
- 車隊位置
- 建立亞馬遜 GameLift 受管叢集

2022 年 6 月 28 日：Amazon GameLift 推出新的選擇加入控制台體驗

全新的 Amazon GameLift 主控台包含下列改良功能：

- 改進的導航 — 新的導航窗格有助於在 Amazon GameLift 資源之間進行導航。
- Amazon GameLift 登陸頁面 — 新的登陸頁面提供實用文件的連結、顯示 Amazon 的高階概觀 GameLift，並透過文件、常見問題和連結提供支援 AWS re:Post。
- 改進的 Amazon CloudWatch 指標 — Amazon GameLift 指標現在可在 Amazon GameLift 控制台和 CloudWatch儀表板中使用。此更新還包括效能、使用率和玩家工作階段的新指標。

進一步了解：

- 在主機中檢視遊戲資料
- 管理亞馬遜GameLift託管資源
- 建立 FlexMatch 媒人

2022 年 2 月 15 日： FlexMatch 添加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 人以下玩家的複合配對規則的支援。您現在可以使用邏輯陳述式建立複合規則以形成相符項目。如果規則集中沒有複合規則，若要形成相符項目，規則集中的所有規則都必須為 true。對於複合規則，您可以使用下列邏輯運算子選擇要套用的規則：`and`、`or`、`not`和`xor`。
- 靈活的團隊選擇 — 更新了配對屬性表達式，以支持選擇所有可用團隊的子集。

- 較長的字串清單 — 在播放程式屬性值的字串清單中，將字串的最大數目從 10 增加到 100。

進一步了解：

- Amazon GameLift FlexMatch 開發者指南：
  - FlexMatch 規則類型
  - FlexMatch 屬性表示式
- AttributeValue: SL

2021 年 10 月 28 日：Amazon 在亞太區域 (大阪) 區域 GameLift 增加了對多區域車隊的支援；Amazon F GameLift leetIQ 增加了對重力失 2 處理器的支援 AWS

更新的開發套件版本： AWS 開發套件 1.9.133

亞太區域 (大阪) 現已推出 Amazon GameLift 服務。遊戲開發人員現在可以使用 GameLift 多區域叢集在大阪部署執行個體。

與同等的 Intel 運算選項相比，您現在可以使用以 ARM 為基礎的處理器架構為基礎的 Graviton2 代管遊戲伺服器，以較低的成本提升效能。

亮點：

- 亞太區域 (大阪) 現已推出 Amazon GameLift 服務。
- Amazon GameLift FleetIQ 遊戲伺服器群組現在可以設定為管理重力 2 執行個體系列 c6g、m6g 和 r6g。

進一步了解：

- Amazon GameLift 多區域機隊
- CreateGameServerGroup
- AWS 重力子處理器

2021 年 9 月 20 日：Amazon GameLift 發布統一插件

統一版本 1.0.0 的 Amazon GameLift 插件包含庫和本機用戶界面，使得它更容易訪問 Amazon GameLift 資源，並 GameLift 將 Amazon 集成到您的統一遊戲。您可以使用適用於 Unity 的 Amazon

GameLift 外掛程式存取 Amazon GameLift API，並為常見的遊戲案例部署 AWS CloudFormation 範本。該插件還包括一個示例遊戲，可以與示例場景一起使用。您可以使用 Amazon L GameLift ocal 查看遊戲用戶端和遊戲伺服器之間傳遞的訊息，以了解典型遊戲如何與 Amazon GameLift 互動。

該插件為統一支持統一 2019.4 LTS 和 2020.3 LTS。

亮點：

- 建置、執行和修改不同情境的範例遊戲，或建立您自己的遊戲。
- 針對典型遊戲 AWS CloudFormation 案例部署範例案例，包括僅授權驗證、單一區域叢集、具有佇列和自訂配對系統的多區域艦隊、具有佇列和自訂配對系統的競價型艦隊，以及。 FlexMatch

進一步了解：

- 用於 Unity 的 Amazon GameLift 插件集成遊戲

2021 年 6 月 30 日: FlexMatch 新增批次距離規則

您可以使用 BatchDistance 規則類型來指定字串或數字屬性，為每個區段帶來許多好處。

亮點：

- 對於大型比賽（>40 名玩家），您現在可以根據技能，模式和地圖獲得相同的平衡，而不是僅按技能平衡玩家。確保比賽中的每個人都處於技能樂隊中，樂隊多個數字屬性（例如聯賽或比賽風格），並根據地圖或遊戲模式等字符串屬性進行分組。您也可以隨時間建立資料片。例如，您可以建立資料片，讓更大的技能等級範圍進入比賽，玩家等待的時間越長。

  對於 40 名玩家以下的比賽，您可以使用新的簡化規則表達式。

2021 年 6 月 3 日：Amazon GameLift 實時客戶端 SDK 和服務器 SDK 更新

更新的 SDK 版本：實時客戶端 SDK 1.2.0，虛幻的服務器 SDK 3.4.0

有了這個最新的 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，並遵循架構的最佳做法。您現在也可以為虛幻版本 4.26 構建 Amazon GameLift 服務器開發套件。此更新包含與您的 Windows 或 Linux 遊戲伺服器整合的元件，包括 C++ 和 C# 版本的 Amazon GameLift 伺服器 SDK、Amazon GameLift 本機和虛幻引擎外掛程式。

亮點：

- 在 RTS 用戶端 SDK 中新增對 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。
- 您可以用DescribePlayerSessions()來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的資訊，或與單一玩家 ID 關聯的所有玩家工作階段的資訊。
- 您可以使GetInstanceCertificate()用擷取與叢集及其執行個體相關聯的 PEM 編碼 TLS 憑證的檔案位置。
- 為虛幻版本 4.26 創建了服務器 SDK 支持。
- 現有的 C# SDK，版本 4.0.2，已經過驗證與統一 2020.3 相容。不需要 SDK 更新。

進一步了解：

- Amazon GameLift 開發者指南：
  - DescribePlayerSessions()
  - GetInstanceCertificate()

2021 年 3 月 23 日：Amazon 在遊戲會話放置中 GameLift 添加了通知

更新的開發套件版本： AWS 開發套件 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放置活動。建立 Amazon Simple Notification Service (Amazon SNS) 主題以發佈事件通知，或使用 CloudWatch 事件設定事件追蹤。

亮點：

- 對於每個佇列，您可以設定要包含在所有事件訊息中的自訂文字字串。
- 使用 Amazon SNS 主題時，您可以設定限制發佈到特定佇列的其他存取條件。

進一步了解：

- Amazon GameLift 開發者指南：
  - 設定遊戲工作階段位置的事件通知 (新)
  - 遊戲工作階段安置事 (新)
- 應用程式介面參考 (AWS SDK)
  - 新的遊戲工作階段佇列參數`NotificationTarget`以及`CustomEventData`：GameSessionQueue、CreateGameSessionQueue、UpdateGameSessionQue

- Amazon GameLift 論壇

2021 年 3 月 16 日：Amazon GameLift 增加了六個新區域的多區域車隊

更新的開發套件版本： AWS 軟體開發套件

Amazon 託 GameLift 管現在可在 21 個 AWS 區域使用。新地區包括開普敦（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）。

有了全新的 Amazon GameLift 多地點叢集功能，您現在可以設定單一叢集，將您的遊戲伺服器託管在 20 個 Amazon GameLift 支援的區域 (北京地區除外) 中的任何一個或全部。此功能旨在大幅減少在全球設定和維護 Amazon GameLift 託管資源所需的工作。您可以在下列 AWS 區域建立多地點叢集：us-east-1(維吉尼亞北部)、us-west-2 (奧勒岡州)、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叢集。在此版本之前建立的所有艦隊都是單一位置的叢集。使用多地點艦隊不會影響您的託管成本。Amazon GameLift 定價是根據您使用的執行個體的類型、位置和數量而定。如需詳細資訊，請參閱 Amazon GameLift 定價。) AWS CloudFormation 即將推出對多地點車隊的支援。

> **Note**
>
> 中國地區不提供多地點機隊。位於中國區域的 Amazon GameLift 資源無法與其他 Amazon 區 GameLift 域的資源互動或使用。

亮點：

- 使用多位置叢集時，明確新增遠端位置清單。Amazon GameLift 會將相同類型和組態的執行個體 (包括建置和執行階段組態) 部署到叢集的本地區域和所有新增的位置。
- 分別調整每個位置的容量設定和調整規模。自動調整規模政策適用於整個叢集，但您可以依據位置開啟或關閉它們。
- 在特定艦隊位置開始新的遊戲階段。使用遊戲工作階段佇列或配對來放置遊戲工作階段時，您現在可以依據位置、代管費用和玩家延遲來排定新遊戲工作階段開始位置的優先順序。
- 在 Amazon GameLift 主控台取得託管指標，針對叢集中的所有位置彙總或按每個叢集位置劃分。

進一步了解：

- Amazon 遊戲技術博客
- 應用程式介面參考 (AWS SDK)
  - 新車隊位置作業：CreateFleetLocationsDescribeFleetLocationAttributes、、DescribeFleetLocationCapacity、Describe
  - 更新了車隊操作，具有新的多位置支持：CreateFleet, UpdateFleetCapacity, 說明 EC2 InstanceLimits,,, DescribeInstancesStopFleetActionsStartFleetActions
  - 更新了遊戲工作階段放置操作，具有新的優先順序和過濾功能：CreateGameSessionQueue、DescribeGameSessionQueues、UpdateGameSessionQueue
  - 更新了遊戲工作階段建立操作，支援新的位置：CreateGameSessionDescribeGameSessions、、DescribeGameSessionDetails、SearchGameSess
- Amazon GameLift 開發者指南：
  - Amazon GameLift 託管地點(已更新)
  - 亞馬遜GameLift車隊設計指南 (新)

    擴展亞馬遜GameLift託管容量(已更新)
  - 設計遊戲工作階段佇列 (新)
  - 檢視車隊詳情(已更新)
- Amazon GameLift 論壇

2021 年 2 月 9 日：Amazon GameLift 擴展了對獨立 AMD 實例的支持 FlexMatch

更新的開發套件版本： AWS 開發套件 1.8.139

此版本包含下列更新：

- Amazon GameLift 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。支援的 Amazon EC2 執行個體類型 (如中 GameServerGroup InstanceDefinition所列) 現在包括下列項目：
  - C5 米。大，c5a.12 倍大，c5a.16 倍大，c5a.4 倍大，c5a.8 倍大，c5a.12 倍大，c5a.16 倍大，c5a.24 倍大
  - m5a. 大, m5a.12 大, m5a.16 倍大, m5a.4 x 大, m5a.8 x 大, m5a.12 倍大, m5a.16 倍大, m5a.24 x 大
  - r5 米。大，立方米。大，5 A.16 倍大，5 A.4 倍大，r5 A.8 倍大，r5a.12 倍大，r5a.16 倍大，R5a.24 x 大

注意：適用於 FleetIQ 的 AMD 執行個體目前無法在中國 (北京) AWS 區域使用。請參閱中國的功能可用性和實施差異。

- Amazon 託 GameLift 管遊戲託管現在支持由 Sinnet 運營的中國（北京）區域的 AMD 執行個體。新的 AMD 執行個體系列包括 M5a 和 R5a。支援的 EC2 執行個體類型 (如叢集InstanceType所列) 現在包括下列項目：
  - m5a. 大, m5a.12 大, m5a.16 倍大, m5a.4 x 大, m5a.8 x 大, m5a.12 倍大, m5a.16 倍大, m5a.24 x 大
  - r5 米。大，立方米。大，5 A.16 倍大，5 A.4 倍大，r5 A.8 倍大，r5a.12 倍大，r5a.16 倍大，R5a.24 x 大
- Amazon 現在 GameLift FlexMatch 可以用作由 Sinnet 運營的中國（北京）區域的獨立配對解決方案。客戶可以在北京地區建立 FlexMatch 分房系統，並將FlexMatchMode參數設定為「獨立」。如需有關 FlexMatch使用 Amazon 託管或非 GameLift Amazon 託 GameLift 管解決方案的詳細資訊，請參閱 Amazon GameLift FlexMatch 開發人員指南。
- 現在，在為亞馬遜設定事件通知時 GameLift FlexMatch，您可以將 Amazon SNS FIFO 主題指定為通知目標。如需詳細資訊，請參閱：
  - MatchmakingConfiguration NotificationTarget，Amazon GameLift API 參考
  - 設置 FlexMatch 事件通知，Amazon GameLift FlexMatch 開發人員指南
  - 介紹 Amazon SNS FIFO — 發irst-in-first-out布/訂閱消息傳遞, 新聞博客AWS

十二月 22, 2020: Amazon GameLift 服務器 SDK 支持虛幻引擎 4.25 和統一 2020

更新 SDK 版本：Amazon GameLift 服務器 SDK 4.0.2，虛幻插件版本 3.3.3

最新版本的 Amazon GameLift 服務器開發套件包含以下組件：

- 更新的虛幻插件已更新與虛幻引擎 4.25 兼容性。API 並未變更。
- 現有的 C# SDK，版本 4.0.2，已被驗證與統一 2020 兼容。不需要 SDK 更新。

在 Amazon 下載最新版本的 Amazon GameLift 服務器開發套件 GameLift 入門。

2020 年十一月 24 日：Amazon GameLift FlexMatch 現在可用於任何地方託管的遊戲

更新的開發套件版本： AWS 開發套件 1.8.95

Amazon GameLift FlexMatch 是一個可定制的多人遊戲配對服務。最初為 Amazon 託 GameLift 管的使用者設計，現在 FlexMatch 可以整合到使用其他託管系統的遊戲中 peer-to-peer，包括專有的現場部署

運算和雲端運算原始類型。在 Amazon Amazon EC2 上使用 Amazon GameLift FleetIQ 進行遊戲託管的遊戲，現在可以使用實作配對功能。 FlexMatch

FlexMatch 提供強大的配對演算法和規則語言，讓您可以自訂配對流程，讓玩家根據關鍵的玩家特性和回報的延遲時間進行配對。此外，還 FlexMatch 提供配對要求工作流程，支援玩家派對、玩家接受度和比賽回填等功能。當您 FlexMatch 與 Amazon 託 GameLift 管主機或即時伺服器搭配使用時，分房系統會自動使用 Amazon GameLift 尋找託管資源，並為新組成的比賽開始新的遊戲工作階段。作 FlexMatch 為獨立服務使用時，分房系統會將比賽結果傳回您的遊戲，然後您可以使用您的主機解決方案開始新的遊戲階段。

的 API 操作 FlexMatch 是 Amazon GameLift 服務 API 的一部分，該服務 API 包含在 AWS SDK 和 AWS Command Line Interface （ AWS CLI ）中。此版本包含以下更新以支援獨立配對功能：

- API 資源`MatchmakingConfiguration`具有以下變更：
  - 新屬性，`FlexMatchMode`表示分房系統是否與 Amazon 託 GameLift 管主機搭配使用，還是作為獨立配對。
  - 當設定`GameSessionQueueArns`為獨立時`FlexMatchMode`，不需要內容。
  - 這些屬性不適用於獨立配對：`AdditionalPlayerCount`、、`BackfillMode`、`GameProperties`、`GameSessionData`。
- 自動回填功能不適用於獨立配對功能。

2020 年 11 月 24 日：AMD 執行個體現已在 Amazon 上推出 GameLift

更新的開發套件版本： AWS 開發套件 1.8.95

Amazon 支援的亞馬遜 EC2 執行個體類型清單 GameLift 現在包括三個新執行個體系列：C5a、M5a 和 R5a。這些系列包含 AMD 運算最佳化執行個體，這些執行個體由 AMD EPYC 處理器提供支援，頻率最高可達 3.3。GHz. AMD 執行個體與 x86 相容；目前在 Amazon 上執行的遊戲 GameLift 可以部署到 AMD 執行個體類型，而不需要進行任何變更。新執行個體在下列 AWS 區域提供：美國東部 (維吉尼亞北部和俄亥俄)、美國西部 (奧勒岡和加利佛尼亞北部)、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)、亞太南部 (孟買)、亞太東北部 (首爾和東京) 以及亞太區域東南 (新加坡和雪梨)。

新的 AMD 執行個體包括：

- C5 米。大，c5a.12 倍大，c5a.16 倍大，c5a.4 倍大，c5a.8 倍大，c5a.12 倍大，c5a.16 倍大，c5a.24 倍大
- m5a. 大, m5a.12 大, m5a.16 倍大, m5a.4 x 大, m5a.8 x 大, m5a.12 倍大, m5a.16 倍大, m5a.24 x 大

- r5 米。大，立方米。大，5 A.16 倍大，5 A.4 倍大，r5 A.8 倍大，r5a.12 倍大，r5a.16 倍大，R5a.24 x 大

進一步了解：

- Amazon 遊戲技術博客
- Amazon GameLift 實例定價
- 具有 AMD EPYC 處理器的 Amazon EC2 執行個體
- Amazon GameLift 論壇

十一月 11, 2020: Amazon GameLift 服務器 SDK 的版本更新

更新 SDK 版本：Amazon GameLift 服務器 SDK 4.0.2

新的服務器 SDK 版本 4.0.2 修復了 API 操作`StartMatchBackfill()`的已知問題。此作業現在會傳回符合回填要求的正確回應。

此問題不會影響比對回填程序，而且此功能的運作方式也沒有變更。此問題可能會影響相符回填要求的記錄訊息和錯誤處理。

在 Amazon 下載最新版本的 Amazon GameLift 服務器開發套件 GameLift 入門。

2020 年 11 月 5 日：新 FlexMatch 演算法自訂

FlexMatch 使用者現在可以調整配對程序的下列預設行為。這些自訂是在配對規則集中設定的。Amazon GameLift 開發套件沒有任何變更。

- 排定回填工單的優先順序：您可以選擇在搜尋可接受的相符項目時，提高或降低比賽回填票券的優先順序。啟用自動回填功能時，排定回填工單的優先順序很有用。使用演算法屬性`backfillPriority`。
- 預先排序以優化比賽的一致性和效率：配置您的分房系統在批處理票證以進行評估之前預先排序票庫。通過根據關鍵玩家屬性預先排序彩票，您的比賽結果通常會有在這些屬性中更相似的玩家。您也可以對比對規則中使用的相同屬性進行預先排序，以提高評估過程的效率。使用演算法屬性`sortByAttributes`，並將`strategy`屬性設定為「排序」。
- 調整擴充等待時間的觸發方式：根據不完整比賽中最新（預設）或最舊票證的年齡，在觸發資料片之間進行選擇。觸發最舊的彩票往往會更快地完成比賽，而觸發最新的彩票會導致更高的比賽品質。使用演算法屬性`expansionAgeSelection`。

九月 17, 2020: Amazon GameLift 更新服務器 SDK

更新 SDK 版本：Amazon GameLift 服務器 SDK 4.0.1

新的伺服器 SDK 包含下列更新：

- C# 應用程式介面版本 4.0.1
  - 不再支援 API 作業TerminateGameSession()。以ProcessEnding()結束遊戲工作階段和伺服器程序的呼叫取代。
  - 已修正作業的GetInstanceCertificate()已知問題。
  - 此作業GetTerminationTime()現在會傳回資料類型的值 AwsDateTimeOutcome。
- C++ 應用程式介面版本 3.4.1
  - 不再支援該作業TerminateGameSession()。將其替換為ProcessEnding()結束遊戲會話和服務器進程的調用。
- 虛幻引擎插件 3.3.2 版
  - 不再支援該作業TerminateGameSession()。將其替換為ProcessEnding()結束遊戲會話和服務器進程的調用。
  - 回呼作業`OnUpdateGameSession`會新增F ProcessParameters至以支援比對回填。

在 Amazon 下載最新版本的 Amazon GameLift 服務器開發套件 GameLift 入門。

2020 年 8 月 27 日：Amazon GameLift FleetIQ 使用 Amazon EC2 進行遊戲託管（正式推出）

更新的開發套件版本： AWS 開發套件 1.8.36

針對 Amazon EC2 上的低成本雲端遊戲託管而設的 Amazon GameLift FleetIQ 解決方案現已正式推出。Amazon GameLift FleetIQ 讓開發人員能夠將遊戲伺服器直接在 Amazon EC2 競價型執行個體上託管遊戲伺服器，方法是最佳化遊戲託管的可行性。遊戲開發人員可以將 Amazon GameLift FleetIQ 與新遊戲搭配使用，或為現有遊戲補充容量。此解決方案支援使用容器或其他 AWS 服務，例如 AWS Shield 牌和亞馬遜彈性容器服務 (Amazon ECS)。

此正式發行版本包含下列 Amazon GameLift FleetIQ 解決方案的更新：

- 新的 API 作業會`DescribeGameServerInstances`傳回 Amazon GameLift FleetIQ 遊戲伺服器群組的所有作用中執行個體的資訊，包括狀態。

- 新的平衡策略ON_DEMAND_ONLY，可將遊戲伺服器群組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，以便視需要在使用 Spot 執行個體和隨需執行個體之間切換。
- 已刪除以下預覽元素，以供正式使用：
  - 使用遊戲伺服器資源的自訂排序鍵。遊戲服務器可以根據註冊時間戳進行排序。
  - 標記遊戲伺服器資源。

2020 年 4 月 16 日：Amazon GameLift 更新用於統一和虛幻引擎的服務器 SDK

更新 SDK 版本：Amazon GameLift 服務器 SDK 4.0.0，Amazon 本 GameLift 地 1.0.5

最新版本的 Amazon GameLift 伺服器開發套件包含下列更新元件：

- C# SDK 版本 4.0.0 為統一 2019 更新。
- 虛幻插件版本 3.3.1 更新虛幻引擎版本 4.22，4.23 和 4.24。
- Amazon 本 GameLift 地版本 1.0.5 已更新，以測試使用 C# 服務器 SDK 版本 4.0.0 的集成。

在 Amazon 下載最新版本的 Amazon GameLift 服務器開發套件 GameLift 入門。

2020 年 4 月 2 日：Amazon GameLift FleetIQ 適用於 EC2 上的遊戲託管（公開預覽）

更新的開發套件版本： AWS 開發套件

Amazon GameLift FleetIQ 功能可將低成本競價型執行個體的可行性最佳化，以便與遊戲託管搭配使用。對於想要直接管理託管資源而非透過 Amazon 託管 GameLift服務的客戶，此功能現已擴展。此解決方案支援使用容器或其他 AWS 服務，例如 AWS Shield 牌和亞馬遜彈性容器服務 (Amazon ECS)。

進一步了解：

GameTech Amazon GameLift Fle@@ etIQ 上的博客文章

2019 年 12 月 19 日：改善了 Amazon AWS 資源的資 GameLift 源管理

更新的開發套件版本： AWS 開發套件

您現在可以利用 Amazon AWS GameLift 資源的資源管理工具。特別是，所有關鍵的 Amazon GameLift 資源 (建置、指令碼、叢集、遊戲工作階段佇列、配對組態和配對規則集) 現在都已指派 Amazon 資源名稱 (ARN) 值。資源 ARN 提供跨所有 AWS 區域唯一的一致標識符。它們可用來建立特定於資源的 AWS Identity and Access Management (IAM) 許可政策。現在會為資源指派 ARN，以及預先存在的資源識別碼 (不是區域特定)。

此外，Amazon GameLift 資源現在支持標記。您可以使用標籤來組織資源、建立 IAM 許可政策來管理資源群組的存取權限、自訂 AWS 成本明細等。管理 Amazon GameLift 資源的標籤時，請使用 Amazon GameLift API 動作 `TagResource()``UntagResource()`，和`ListTagsForResource()`.

進一步了解：

- TagResource在 Amazon GameLift API 參考
- AWS 一般參考中的標記 AWS 資源
- Amazon 資源名稱在一AWS 般參考

2019 年 11 月 14 日：新 AWS CloudFormation 模板，中國（北京）地區的更新

更新的開發套件版本： AWS 軟體開發套件

AWS CloudFormation Amazon 模板 GameLift

現在可以透過建立和管理 Amazon GameLift 資源 AWS CloudFormation。現有的組 AWS CloudFormation 建和叢集範本已更新，以符合目前的資源，而新範本現在可用於指令碼、佇列、配對配置和配對規則集。 AWS CloudFormation 範本可大幅簡化管理相關 AWS 資源群組的工作，尤其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 資源類型參考
- 使用管理資源 AWS CloudFormation在 Amazon 開 GameLift 發人員指南

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。