AWS 白皮書

遊戲產業鏡頭

遊戲產業鏡頭: AWS 白皮書

Table of Contents

遊戲產業鏡頭 – AWS Well-Architected Framework

發佈日期：2025 年 12 月 9 日 (文件修訂)

AWS Well-Architected Framework 可協助雲端架構師為其應用程式和工作負載建置安全、高效能、彈性且高效率的基礎設施。Well-Architected 以營運卓越性、安全性、可靠性、效能效率、成本最佳化和永續性六大支柱為基礎，為客戶提供一致的方法來 AWS 評估架構、修復風險，並實作可提供商業價值的設計。

在此鏡頭中，我們著重於如何在 中設計、部署和架構您的遊戲工作負載 AWS 雲端。我們定義元件、探索常見的工作負載案例，並概述可協助您套用 Well-Architected Framework 的設計原則。我們建議您考慮 AWS Well-Architected Framework 白皮書中的最佳實務和問題，開始設計架構。本文件提供遊戲產業客戶的補充最佳實務。

此鏡頭指定最佳實務，旨在根據我們與遊戲產業開發人員和全球發佈者合作的經驗，解決在雲端中建置和操作遊戲的獨特特性。它提供如何設計和操作環境的指引，以便針對全球玩家需求的波動進行成本最佳化和可擴展性。此鏡頭也提供保護遊戲基礎設施和調校效能的指導方針，以提供正面的玩家體驗。

本文件適用於擔任技術角色的人員，例如技術長 (CTOs)、遊戲工作室技術主管、架構師、開發人員和營運團隊成員。閱讀本文件後，您將了解設計遊戲架構時要使用的 AWS 最佳實務和策略。

鏡頭可用性

自訂鏡頭延伸了 提供的最佳實務指引 AWS Well-Architected Tool。 AWS WA Tool 可讓您建立自己的自訂鏡頭，或使用其他人為您建立的鏡頭。

若要開始檢閱遊戲工作負載， AWS Well-Architected Tool 請從公有 AWS Well-Architected 自訂鏡頭 GitHub 儲存庫下載遊戲產業鏡頭並將其匯入 。

設計原則

AWS Well-Architected Framework 識別下列一般設計原則，以促進遊戲工作負載在雲端中的良好設計：

- 了解玩家行為和使用模式以發展遊戲並協助保護玩家：為了持續改善遊戲並有效管理玩家體驗，請務必了解玩家如何與遊戲本身和其他玩家互動。這可協助您了解如何改善遊戲、管理成本，以及監控和回應對玩家體驗構成風險的未經授權使用活動。
- 使用可簡化遊戲操作並提高開發速度的技術：優先採用可提高速度的技術，並減少為玩家提供新功能和改進的操作開銷。遊戲是命中驅動的，玩家有很多選擇可以考慮，因此快速移動和適應變化對於遊戲的成功至關重要。考慮您是否願意操作自己的軟體，或者您是否偏好採用來自 AWS、 AWS 合作夥伴或兩者的同等受管服務。
- 最佳化您的架構以改善反映實際玩家體驗的指標：隨著您隨著時間調整和發展架構，請考慮這些改進和變更將如何影響玩家體驗。遊戲工作負載應該能夠承受並盡量減少失敗的影響，以封鎖對遊戲的廣泛干擾。不嚴重依賴彼此的遊戲功能和系統應解除耦合，以減少故障的影響，並隔離影響問題的玩家。
- 設計基礎設施以滿足尖峰玩家並行並根據需要動態擴展：基礎設施應設計為擴展以滿足玩家需求。玩家工作階段並行和登入次數等指標可用於在系統過載之前先行擴展。被動系統使用率指標，例如 CPU 和記憶體耗用量，可用於在系統超載之後進行擴展。透過動態擴展您的基礎設施，您可以降低操作遊戲的成本。
- 實作 Runbook 以改善遊戲操作：應使用 Operational Runbook 來持續管理重複的遊戲操作任務。執行手冊應適用於常見的遊戲操作工作流程，例如調查和回應玩家報告、管理基礎設施預先擴展活動，以為新賽季啟動和遊戲內容發行等預期的大規模事件做好準備，以及處理典型的遊戲維護活動。

案例

在本節中，我們涵蓋遊戲架構中常見的多種案例。每個案例都包含推動設計的常見特性和範例參考架構圖表。

案例

即時同步遊戲體驗的遊戲託管

即時同步遊戲可讓兩個或多個玩家同時參與和互動遊戲，其中遊戲狀態會在連線的玩家之間共用，以盡可能建立接近即時體驗的遊戲。同步遊戲的範例包括第一人稱射擊遊戲、大量多玩家線上遊戲 (MMOG)、運動和動作遊戲，或是線上遊戲，其中必須有兩名或多名玩家連線才能近乎即時地共用遊戲體驗。

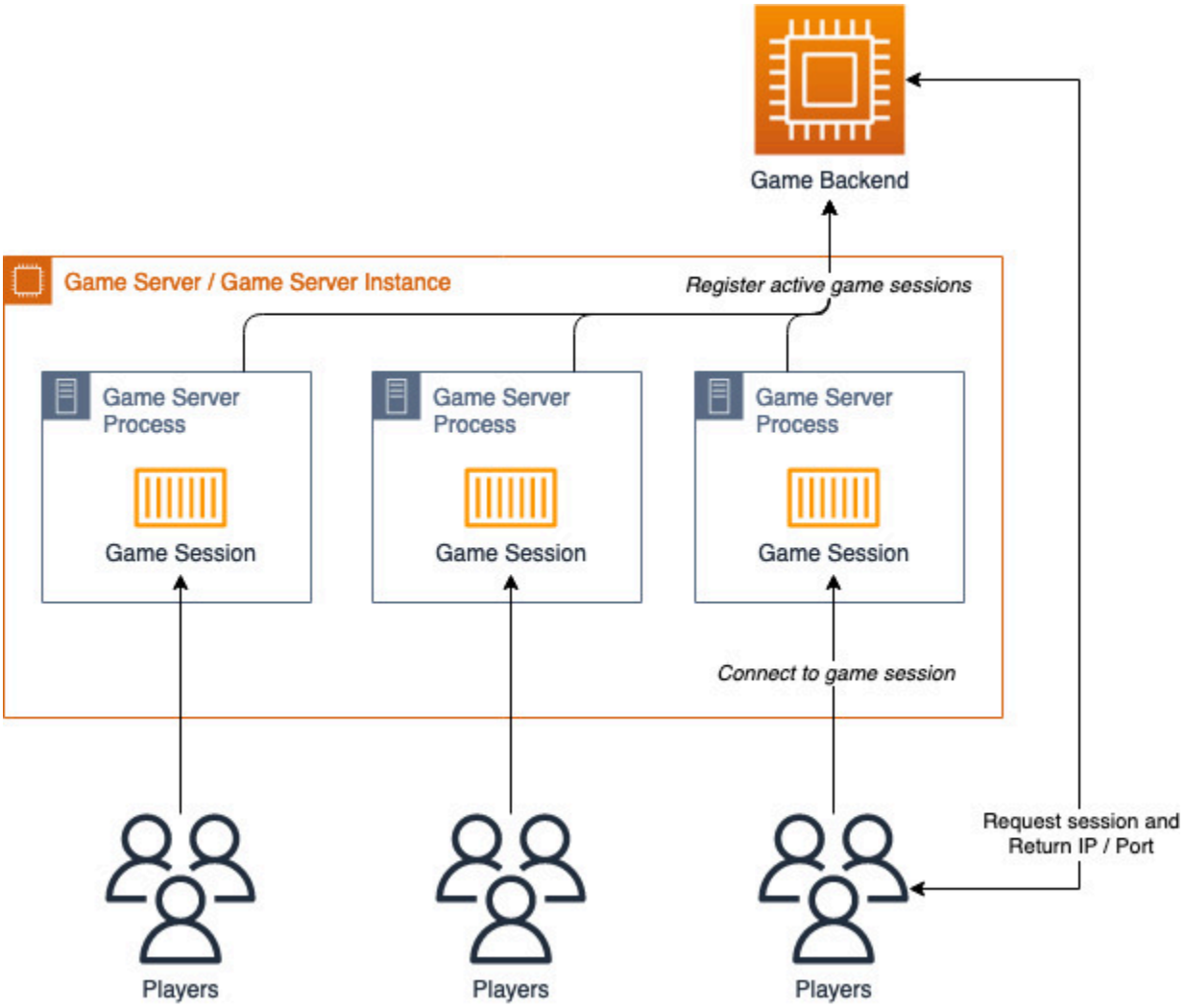
即時同步遊戲架構的特性包括：

- 某些遊戲可能會透過在專用伺服器上執行的遊戲伺服器程序，以遊戲工作階段的形式託管。有些遊戲可能會使用P2P-like 的架構，這些架構採用較輕量的工作階段周遊 NAT (STUN) 公用程式，或使用 NAT 伺服器周圍的轉送 (TURN) 周遊。無論涉及的伺服器類型為何，遊戲伺服器都會託管在多個資料中心和 AWS 區域 全球。
- 遊戲用戶端可以從遊戲後端系統託管的集中式配對服務請求配對，或從預先定義的可用遊戲伺服器清單中選擇配對，以加入遊戲工作階段。遊戲用戶端會獲得要連線的 IP 地址和連接埠。
- 許多同步遊戲對延遲敏感，例如第一人稱射擊遊戲和大量多玩家線上遊戲。它們可能包括倒轉和時間擴展等演算法，以將延遲效果降至最低，但也可能具有預先定義的延遲容錯能力，經過仔細測量和最佳化，以減少玩家在高延遲情況下有時會發生的延遲體驗。此延遲資訊是透過檢測遊戲用戶端來 ping 可用的遊戲伺服器 AWS 區域 ，以擷取延遲、網路抖動等指標，以及遊戲體驗的其他重要指標來決定。這些指標會傳送到遊戲後端系統中的中央指標收集服務，以便即時操作串流可以監控遊戲運作狀態。在配對過程中，遊戲用戶端在請求配對時提供其目前的延遲資料作為其中一個請求參數，而配對服務可以在選取遊戲伺服器來託管玩家時，使用該延遲資料作為其中一個變數。

- 一般而言，遊戲是透過多種通訊協定（例如使用較快速的 UDP 型傳訊搭配配對、身分驗證，以及使用 HTTPS 的其他用戶端伺服器流量）來執行。
- 遊戲伺服器採用演算法和設計，將轉換串流、差異和資料壓縮等用戶端伺服器流量降至最低。
- 遊戲伺服器是惡意活動的頻繁目標，應該使用類似 DDoS 保護解決方案進行保護 AWS Shield Advanced。

遊戲伺服器程序

下圖說明典型的遊戲伺服器架構。它描述遊戲伺服器執行個體與託管遊戲工作階段的遊戲伺服器程序之間的邏輯關係。



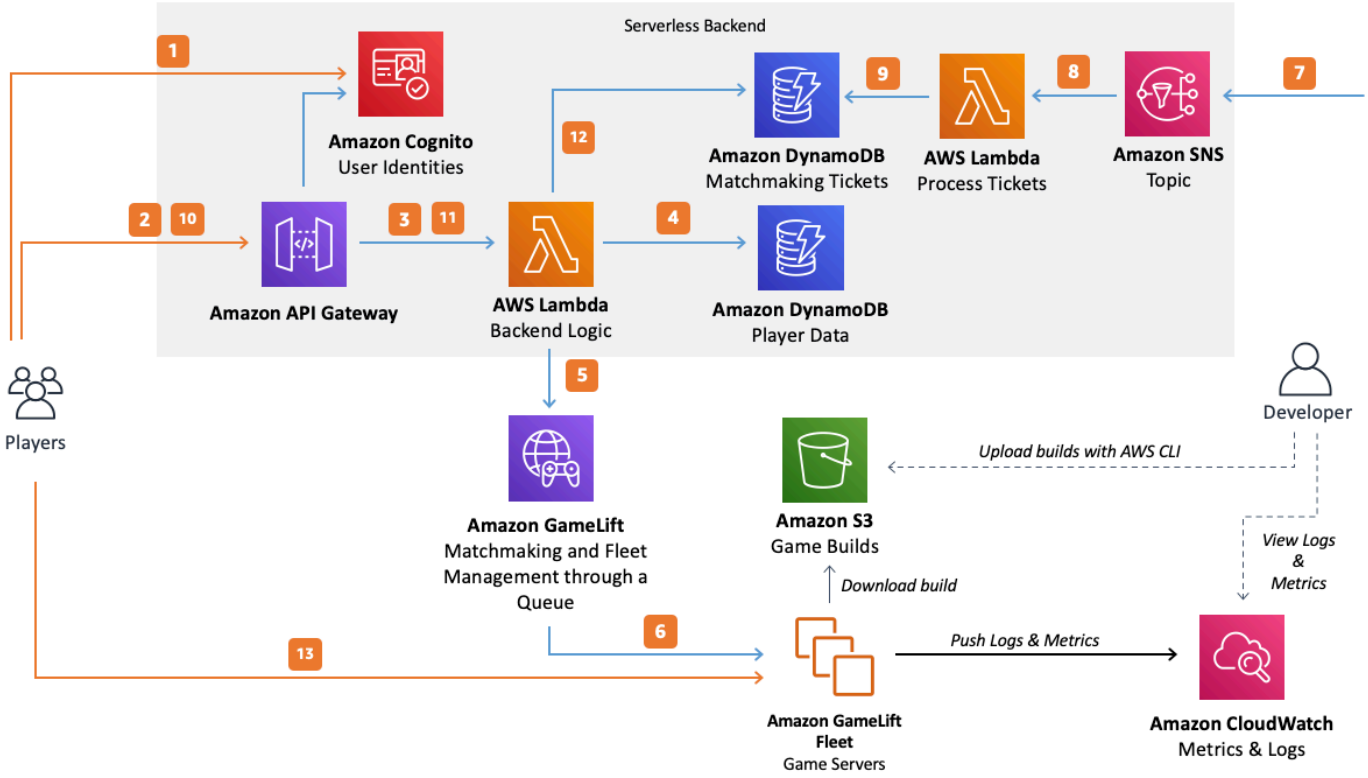
邏輯遊戲伺服器架構

- Amazon EC2 執行個體用作遊戲伺服器，也稱為遊戲伺服器執行個體。遊戲伺服器託管一或多個遊戲伺服器程序，每個程序都會執行遊戲伺服器建置的副本。一般而言，多個遊戲伺服器程序會在遊戲伺服器執行個體上執行，以有效率地利用運算資源並降低成本。當遊戲工作階段處於作用中狀態並準備好託管玩家工作階段時，其狀態會以遊戲後端（通常是配對服務）更新，以便開始用來託管玩家。
- 遊戲後端可以為玩家提供託管遊戲工作階段的 IP 地址和伺服器連接埠，讓他們可以連接到遊戲。

使用無伺服器後端託管工作階段型遊戲伺服器

為您的遊戲開發架構時，請考慮您需要的特性和功能，以及您準備擁有的操作管理開銷層級。為了在操作簡單性和靈活性之間取得最佳平衡，您可以使用雲端供應商的受管服務來建置遊戲。受管服務可讓您控制開發和自訂自己的自訂遊戲功能，同時減輕部署和管理基礎設施的負擔。

託管以工作階段為基礎的多玩家遊戲需要伺服器基礎設施來託管遊戲伺服器程序，以及可擴展的後端以進行配對和工作階段管理。下列參考架構顯示如何使用 Amazon GameLift 受管託管和無伺服器後端來管理您的工作階段型遊戲。



適用於工作階段型遊戲的 Amazon GameLift 受管託管

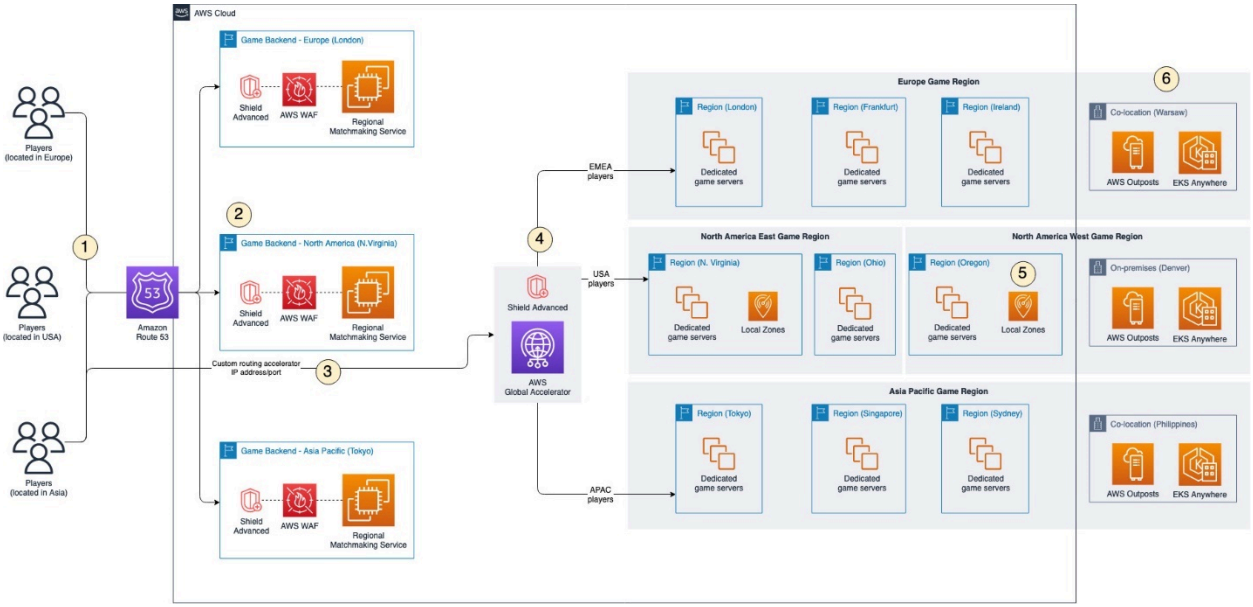
圖表說明讓玩家進入在 GameLift 受管遊戲託管上執行之遊戲的程序。它包含下列步驟：

1. 遊戲用戶端會從 Amazon Cognito 身分集區請求 Amazon Cognito 身分。這可以選擇性地連接到外部身分提供者。
2. 遊戲用戶端會收到臨時存取登入資料，並使用 Amazon Cognito 登入資料簽署請求，透過 Amazon API Gateway 請求遊戲工作階段。 Amazon Cognito
3. API Gateway 會叫用 AWS Lambda 函數。
4. Lambda 函數會從 Amazon DynamoDB 資料表請求玩家資料。Amazon Cognito 身分用於安全地請求正確的玩家資料，因為已驗證的身分是在請求內容資料中提供。
5. 使用正確的玩家資料取得其他資訊（例如玩家技能等級），Lambda 函數會透過 GameLift FlexMatch 配對來請求配對。您可以使用 JSON 型組態文件來定義 FlexMatch 配對組態。遊戲用戶端可以透過 ping 各個區域中的伺服器端點來產生延遲指標，而且延遲資料可用於支援以延遲為基礎的配對。
6. FlexMatch 將具有適當延遲的適當玩家群組配對至區域後，會透過 GameLift 佇列請求遊戲工作階段放置。佇列包含具有一或多個已註冊區域位置的機群。
7. 當工作階段放置在其中一個機群的位置時，事件通知會傳送至 Amazon SNS 主題。
8. Lambda 函數將接收並處理 Amazon SNS 事件。
9. 如果 Amazon SNS 訊息是 MatchmakingSucceeded 事件，Lambda 函數會使用伺服器連接埠和 IP 地址將結果寫入 DynamoDB。time-to-live (TTL) 值用於確保在不再需要配對票證時從 DynamoDB 中刪除。
10. 遊戲用戶端向 API Gateway 發出簽署請求，以在特定間隔檢查配對票證的狀態。
11. API Gateway 會叫用 Lambda 函數，以檢查配對票證狀態。
12. Lambda 函數會檢查 DynamoDB，以判斷票證是否成功。如果成功，Lambda 函數會將 IP 地址、連接埠和玩家工作階段 ID 傳回用戶端。如果票證失敗，Lambda 函數會傳送回應，宣告相符項目尚未就緒。
13. 遊戲用戶端會使用後端提供的連接埠和 IP 地址，使用 TCP 或 UDP 連線至遊戲伺服器。它會將玩家工作階段 ID 傳送至遊戲伺服器，而遊戲伺服器會使用 Amazon GameLift Server SDK 進行驗證。

或者，您可以修改上述架構，將 API Gateway WebSockets 與 Amazon GameLift 搭配使用。在此方法中，遊戲用戶端與遊戲後端服務之間的通訊會使用以 WebSocket 為基礎的實作進行。可以使用此實作，讓遊戲後端 Lambda 函數透過 WebSocket 向遊戲用戶端啟動伺服器端訊息，而不是實作輪詢模型。

適用於低延遲遊戲的多區域混合架構

本節說明低延遲遊戲的多區域和混合架構。



透過全球部署的網路加速和遊戲伺服器來降低延遲

1. 全球可用遊戲中的玩家可以來自任何地方。當玩家請求遊戲工作階段或配對時，其遊戲用戶端會將請求傳送至向 Amazon Route 53 註冊的遊戲後端服務。Route 53 延遲型路由可用來將玩家路由到最接近的可用遊戲後端。
2. 遊戲後端部署在最 AWS 區域 接近玩家群體的多個位置。每個遊戲後端都包含區域配對服務，可尋找來自整個遊戲區域的遊戲工作階段。雖然玩家的配對請求是由他們附近的區域配對服務處理，但配對服務可以視需要將玩家路由到另一個遊戲區域中的遊戲工作階段。此動作可改善彈性和效能。此外，每個遊戲後端服務都會使用AWS WAF 和 AWS Shield Advanced 來提供 layer-7 Web 篩選和機器人控制，並防止分佈拒絕服務 (DDoS) 攻擊。您有許多建置遊戲後端服務的選項，例如無伺服器、容器、EC2 執行個體，或在您自己的資料中心託管遊戲後端服務。
3. 為了透過減少網路延遲和抖動來改善玩家的體驗，使用 AWS Global Accelerator 部署自訂路由加速器，這會使用 AWS 全域網路自動最佳化從遊戲用戶端到遊戲伺服器的流量路由。您可以設定自訂路由加速器，將 Global Accelerator 接聽程式連接埠映射至遊戲伺服器的 EC2 執行個體連接埠。遊戲用戶端會連線至做為代理的 Global Accelerator IP 和連接埠，並確定將玩家路由到託管遊戲工作階段的正確遊戲伺服器 IP 和連接埠。
4. 您的遊戲包含玩家友好的邏輯遊戲區域，代表地理上彼此接近的遊戲伺服器託管位置集合，例如北美洲或亞太區域。為了減少延遲並提高地理覆蓋範圍，您可以使用不同遊戲伺服器託管解決方案的組合來改善玩家體驗。 AWS 區域 盡可能優先使用 ，因為這些位置完全特色化，且包含最大的容量使用量。
5. 使用 AWS Local Zones 將遊戲伺服器託管在您沒有現有託管設施或 AWS 區域 無法使用的未受服務玩家地理位置。

6. 使用全受管機架和機架掛載的伺服器，部署 AWS Outposts 到現有的現場部署資料中心和主機代管供應商，以跨每個部署位置建立無縫的控制平面和管理體驗。如果您的現場部署環境中沒有現有的伺服器容量， 也 AWS Outposts 很有幫助。不過，如果您想要在現有的伺服器基礎設施上執行軟體的混合式實作，您應該使用 Amazon EKS Anywhere，這可讓您在自己的基礎設施上建立和執行 Kubernetes 叢集，並連線至 Amazon EKS。這可提供現場部署中 Kubernetes 叢集的一致主控台檢視。

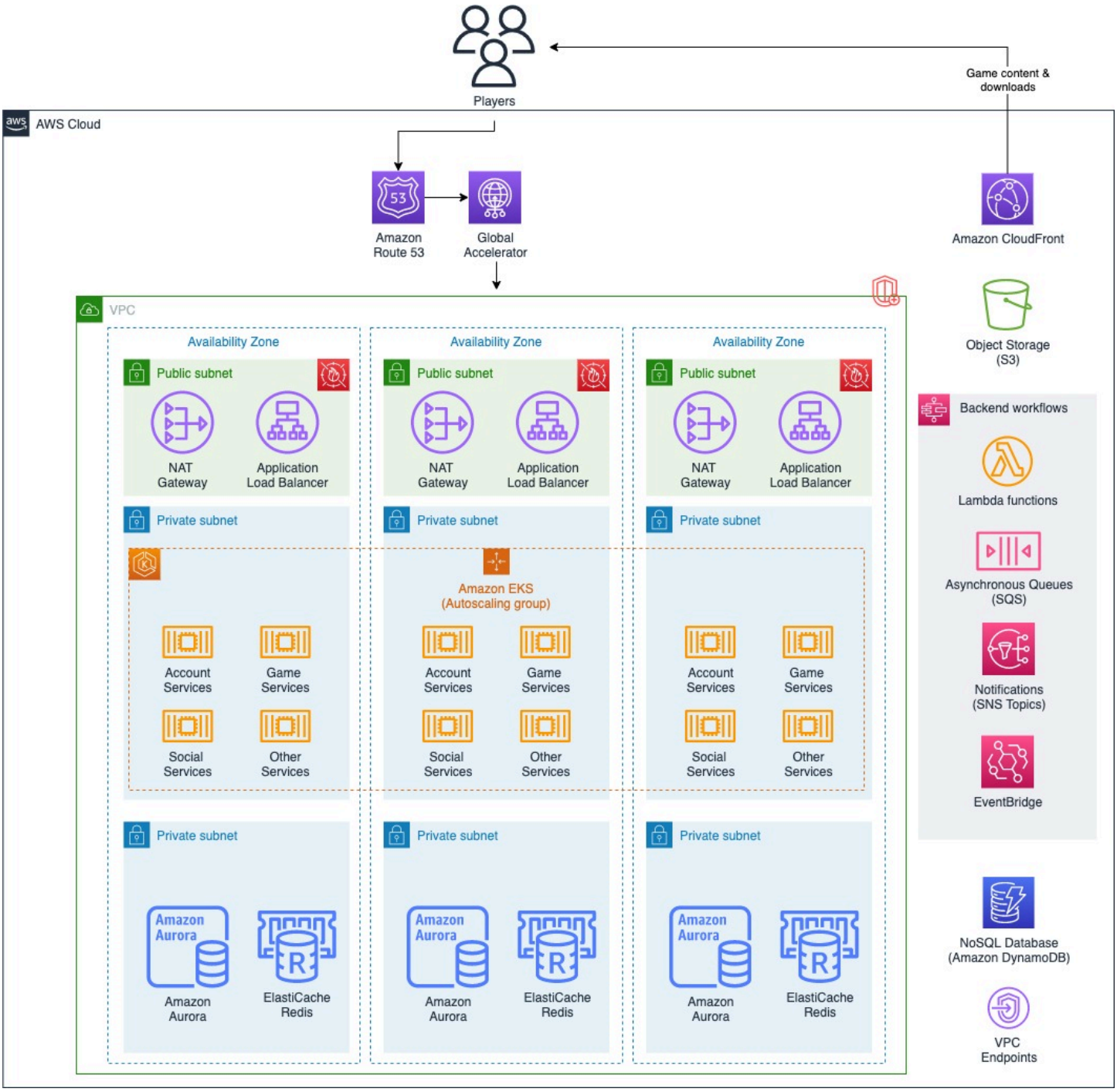
遊戲後端

遊戲後端用於管理遊戲和玩家狀態，並將社交和系統層級功能整合到支援遊戲體驗的遊戲中。玩家設定檔管理、項目和庫存儲存，以及統計資料和排行榜是遊戲後端中託管的服務範例。

遊戲後端通常會建置為由用戶端使用 HTTPS 存取的 REST APIs。不過，其他方法也很常見，例如為使用案例提供雙向通道的 WebSockets例如遊戲內聊天和存在的用戶端通知。您可以使用各種不同的部署架構來部署遊戲後端，包括使用執行個體、容器或無伺服器架構。

容器型遊戲後端架構

本節概述容器型遊戲後端架構。



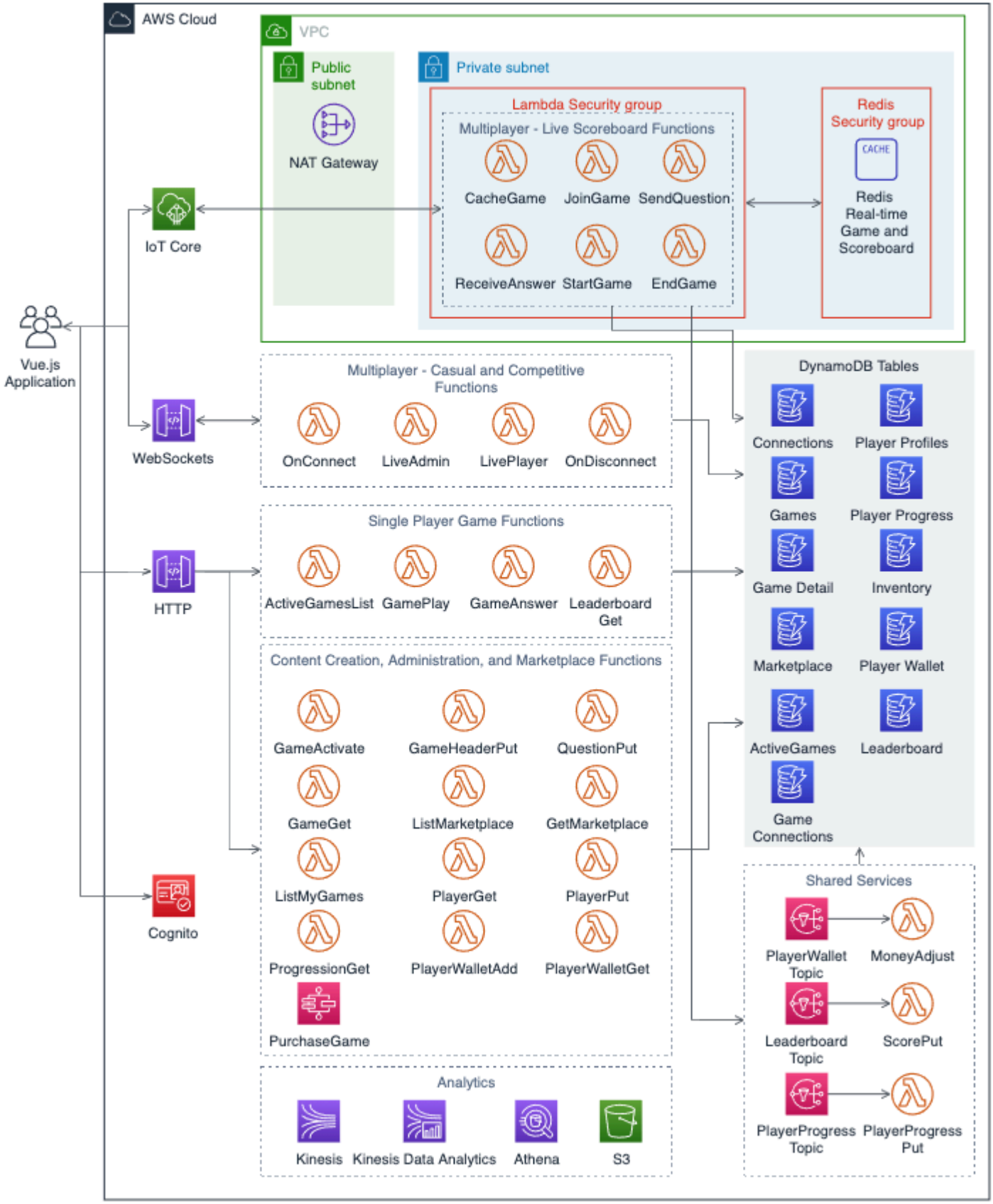
使用容器託管遊戲後端

- 玩家可以使用遊戲用戶端軟體存取遊戲，可透過遊戲系統、數位商店或直接從內容交付網路 (CDN)，例如 Amazon CloudFront 進行下載。CDNs 在節點提供快取，以加速使用者下載內容的效能。例如，CloudFront 可用來將遊戲用戶端軟體分發給玩家，以及遊戲資產和其他內容。

- AWS Global Accelerator 提供流量加速和可自訂的控制，可將流量從玩家遊戲用戶端路由到負載平衡器，以及將流量路由到區域以用於多區域和容錯移轉目的。遊戲後端 REST APIs的自訂網域名稱是在 Amazon Route 53 中設定，以將流量路由至 Global Accelerator 端點。未顯示在圖表中，AWS Shield Advanced 可以為您的加速器和遊戲後端提供額外的 DDoS 緩解措施。
- NAT Gateway 和 Application Load Balancer 會部署到遊戲後端使用的每個可用區域中的公有子網路，以提供 區域的高可用性。AWS WAF 會部署在 Application Load Balancer 上，以提供 layer-7 Web 流量篩選。
- 遊戲後端託管為個別容器型微服務集合，這些服務部署至跨可用區域分佈的私有子網路中的 Amazon EKS 叢集，以提供彈性。自動擴展會根據資源使用率動態調整服務和叢集節點的容量，這通常與玩家需求相關。雖然 Cluster Autoscaler 會自動調整叢集中的節點數量，但 Horizontal Pod Autoscaler 會自動擴展部署到叢集中的 Pod。
- 遊戲和玩家資料存放在後端資料庫和快取中，這些快取會部署到跨可用區域的私有子網路，並在主要節點和複本節點之間進行複寫。Amazon Aurora 是熱門的使用案例選擇，例如玩家設定檔、權利和遊戲內購買，其可能具有更複雜的查詢需求，並可能受益於 MySQL 和 PostgreSQL 的關聯式資料建模功能。Amazon ElastiCache 適用於建置高效能排行榜、pub/sub 訊息，以及快取經常存取的資料，以減少資料庫的延遲和負載。Amazon DynamoDB 是全受管的 NoSQL 資料存放區，非常適合不可預測的存取模式，以及針對玩家和遊戲狀態資料、工作階段資料、庫存和項目存放區等使用案例，或您希望全域資料庫負荷降到最低的使用案例，擴展到幾乎無限制的輸送量。
- 非同步處理工作流程應用於執行可在背景完成的工作，例如更新排行榜或傳送好友請求。設定您的遊戲後端，將此類工作推送至 Amazon SQS 佇列，隨著遊戲的成長進行擴展，或考慮使用 Amazon SNS 主題在許多取用者應用程式佇列之間分配工作以進行平行處理。使用 AWS Lambda 函數 以事件驅動的方式執行處理，以減少運算基礎設施成本和管理開銷。對於長期或需要任務協調多個步驟的工作流程，請考慮使用 協調整個工作流程AWS Step Functions。Amazon EventBridge 可用來啟動函數，以回應 AWS 服務和自訂應用程式事件。

無伺服器型遊戲後端架構

許多遊戲開發人員不想管理基礎設施，而是偏好使用允許他們專注於軟體的技術來建置遊戲。在此案例中，建議使用無伺服器架構，因為它可讓您更快速地建置和發行功能，並降低營運開銷。無伺服器架構的設計使用雲端服務，可根據需求動態擴展，而不需要設定、管理和擴展伺服器。下列參考架構說明如何使用無伺服器架構建置遊戲。


AWS Cloud
VPC
Public subnet
NAT Gateway
Private subnet
Lambda Security group
Multiplayer - Live Scoreboard Functions
CacheGame
JoinGame
SendQuestion
ReceiveAnswer
StartGame
EndGame
Redis Security group
CACHE
Redis Real-time Game and Scoreboard
IoT Core
Vue.js Application
WebSockets
HTTP
Cognito
Multiplayer - Casual and Competitive Functions
OnConnect
LiveAdmin
LivePlayer
OnDisconnect
Single Player Game Functions
ActiveGamesList
GamePlay
GameAnswer
Leaderboard Get
Content Creation, Administration, and Marketplace Functions
GameActivate
GameHeaderPut
QuestionPut
GameGet
ListMarketplace
GetMarketplace
ListMyGames
PlayerGet
PlayerPut
ProgressionGet
PlayerWalletAdd
PlayerWalletGet
PurchaseGame
Analytics
Kinesis
Kinesis Data Analytics
Athena
S3
DynamoDB Tables
Connections
Player Profiles
Games
Player Progress
Game Detail
Inventory
Marketplace
Player Wallet
ActiveGames
Leaderboard
Game Connections
Shared Services
PlayerWallet Topic
MoneyAdjust
Leaderboard Topic
ScorePut
PlayerProgress Topic
PlayerProgress Put

無伺服器型遊戲後端參考架構

此參考架構說明提供單一玩家和多玩家功能的 Web 型三角遊戲。

- 玩家身分驗證：玩家使用 Amazon Cognito 進行身分驗證，其提供安全身分驗證與玩家身分管理的使用者目錄。
- 遊戲邏輯作為無伺服器函數：遊戲功能和後端商業邏輯作為回應事件而啟動的 AWS Lambda 函數執行，這會降低成本，因為您只在函數執行時付費。Lambda 可讓您靈活地使用您選擇的程式設計語言，將每個遊戲功能寫入為單獨的微服務。例如，如果您有使用 C# 建置 Unity 遊戲的經驗，您可以選擇開發 .NET Lambda 函數，或者如果您想要在 JavaScript 中編寫 Web 遊戲的前端和後端程式，可以選擇開發 Node.js Lambda 函數。
- 遊戲和玩家資料的 NoSQL Data Store：使用 DynamoDB 來存放您的玩家和遊戲資料，因為它專為從微服務存放大量資料而打造。如此架構所示，最佳實務是針對每個遊戲功能的資料儲存需求使用個別的資料儲存，這可讓您直接獨立監控和管理功能。如果團隊中的功能或服務擁有權發生變更，這也會建立分離界限。在此參考架構中，DynamoDB 資料表用於存放連線狀態、遊戲詳細資訊、玩家進度和排行榜資訊等資料。
- 單一玩家遊戲：單一玩家功能可讓玩家執行動作，例如選取和玩遊戲，以及檢視排行榜。這些功能會實作為使用 Amazon API Gateway HTTP API 託管的 RESTful 後端服務，該 API 會叫用適當的 Lambda 函數，以取得和設定 DynamoDB 資料表中的資料。遊戲完成時，後端也會將通知傳送至 Amazon SNS 主題，以非同步方式啟動 Lambda 函數來儲存玩家的進度和統計資料。
- 多玩家遊戲：多玩家遊戲功能需要玩家能夠與遊戲互動以進行point-to-point通訊，以及廣播和接收來自其他連線玩家的更新。WebSockets 實作適用於輕量型遊戲中的point-to-point通訊，例如 trivia。玩家可以建立與 Amazon API Gateway WebSockets 的 WebSockets 連線，以管理連線，並只在有要為玩家傳送或接收的訊息時叫用 Lambda 函數。 WebSockets 對於玩家之間需要one-to-many通訊的使用案例， AWS IoT Core 支援透過 MQTT 使用 WebSockets 傳送訊息，這允許用戶端訂閱主題並對收到的訊息採取行動。在此架構中，透過 MQTT 的 WebSockets用於支援使用案例，例如廣播即時遊戲內更新，以及向連線的玩家提出問題。或者 AWS IoT，如果您需要訊息保留，您可以選擇 Redis Pub/Sub 進行訊息傳遞，或選擇 Redis Streams。
- 使用啟用 VPC 的 Lambda 函數來存取私有子網路中的資源：設定啟用 VPC 的 Lambda 函數來存取 VPC 私有子網路中的資源，例如 Amazon ElastiCache，用於縮短即時排行榜等低延遲資料集的查詢時間。

如需詳細資訊，請參閱 上的自訂遊戲後端託管指南 AWS。

雲端遊戲開發 (CGD)

雲端遊戲開發 (CGD) 是指遊戲開發生命週期建置、測試和開發遊戲所需的基礎設施和工具。遊戲開發是在使用者與基礎設施需求之間協作的，在開發階段經常變更。

許多遊戲開發人員都採用全球分散式和遠端開發團隊，這需要支援這種開發類型的技術。遊戲開發人員可以在 中託管這些環境的全部或部分， AWS 並使用 AWS 區域 的全域可用性，視需要擴展運算和儲存，以更符合成本效益的方式將資源放置在更接近使用者的地方，並管理其開發環境。

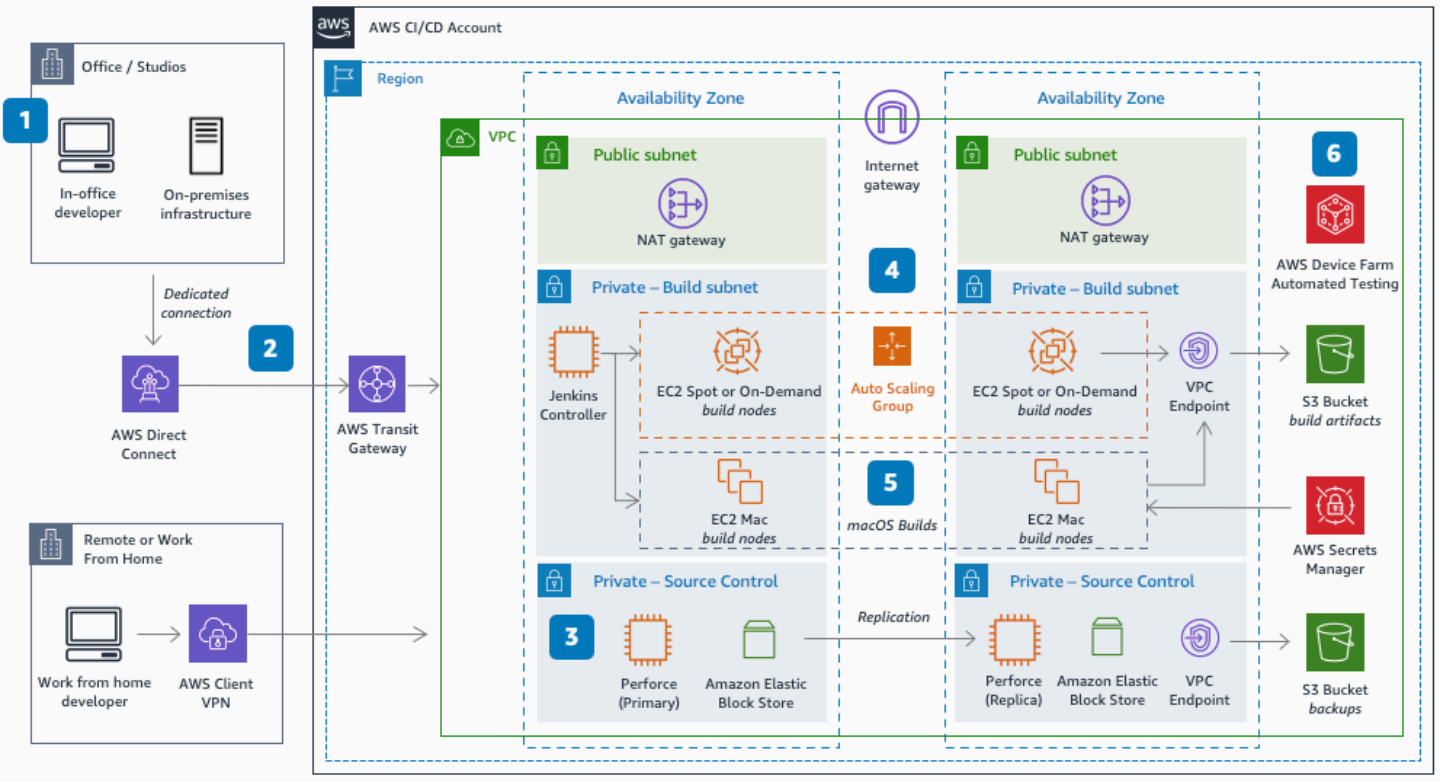
環境可能會因遊戲開發人員需求而有所不同，但通常會包含開發人員工作站，讓藝術家、設計師、工程師、QA 測試人員、承包商和其他人員執行其工作。這些環境通常也包含由原始程式碼儲存庫組成的建置陣列，供使用者檢查其變更，以及用於建置、封裝和測試已開發成品的 CI/CD 基礎設施。

這些遊戲生產架構具有下列特性：

- 使用者應能夠透過 Web 瀏覽器或本機桌面用戶端存取虛擬工作站，例如 Amazon DCV，該用戶端提供低延遲串流工作階段，以存取他們在辦公室或開發工作室的機器上工作時可存取的相同軟體和工具。這些虛擬工作站，通常是雲端伺服器，應允許使用者透過 LAN 或 WAN，在雲端環境中完全協作和處理其專案。當使用者未主動使用機器時，其工作應備份至耐用的雲端儲存體，例如來源控制儲存庫或檔案系統，例如 Amazon Elastic File System (EFS) 和 Amazon FSx，且其機器應關閉以降低成本。
- 來源控制儲存庫，例如 Perforce，應使用可用區域之間的複寫或在內部部署環境之間進行高可用性設計，並將備份存放在 Amazon S3 等雲端儲存體中。例如，雲端型 Perforce 伺服器應包含託管在一個可用區域中的主要遞交伺服器，並複寫到相同區域中另一個可用區域中託管的待命伺服器。
- 遊戲開發組建陣列資源應採用自動擴展設計，以便視需要佈建運算資源，並且應使用 EC2 Spot 執行個體來降低擴展組建所需的伺服器數量時所產生的成本。

雲端遊戲開發：CI/CD

無論團隊大小為何，開發遊戲時，CI/CD 基礎設施都很重要，以改善迭代時間、建置可靠性、高效部署，以及更好地控制開發和發行程序，為玩家提供高品質的遊戲體驗。遊戲開發 CI/CD 管道通常包含高可用性的來源控制伺服器和儲存體、執行建置的運算資源，以及執行自動化測試的軟體，以及來自開發機器的適當網路連線。下列參考架構示範如何將遊戲組建從遠端或內部部署遊戲開發環境卸載至 AWS 雲端 ，以協助開發人員遷移或建置新的組建陣列。



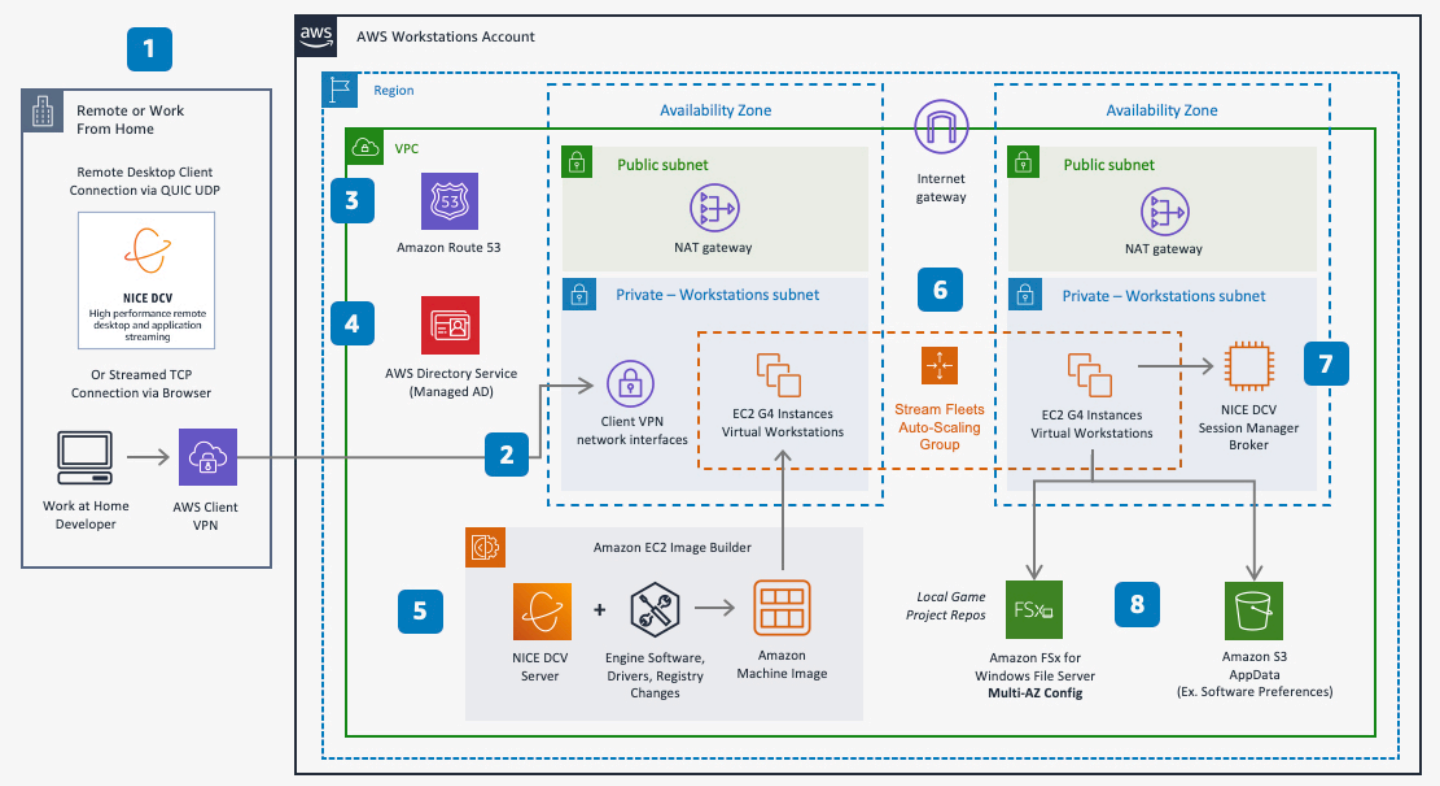
將遊戲組建卸載至雲端

1. AWS Direct Connect AWS 為辦公室開發人員提供低延遲、私有的專用 連線。遠端開發人員使用零信任技術 AWS Verified Access，例如 或虛擬私有網路 (VPN)，例如 AWS Client VPN。
2. AWS Transit Gateway 簡化 VPCs與內部部署網路之間連線的網路管理。
3. Perforce 管理 Amazon EBS 儲存體支援的來源和版本控制 (CI)，以快速存取持久性資料。Perforce Helix Core 可在 中取得AWS Marketplace。
4. 當開發人員將變更推送至繫結至分支的 Perforce 時，遞交會在 Jenkins 中啟動組建 (CD)。Perforce 會對 Jenkins 啟動 POST JSON 承載。Jenkins 控制器會呼叫引擎無周邊 CLI 命令，以跨暫時性 Docker 節點 （例如 Amazon EC2 Spot 執行個體或 Amazon EC2 隨需執行個體） 執行和平行化建置程序。開發人員可以使用兩個 Jenkins 控制器來提高可用性，每個可用區域中一個控制器位於負載平衡器後方。對於某些遊戲引擎，開發人員可能需要在其他子網路中設定的額外授權基礎設施，才能在每次執行並行建置時為建置內容提供授權。
5. iOS 建置的 Xcode 部分會卸載至 Amazon EC2 Mac 執行個體，以簽署、建置和匯出 .IPA 檔案，分割程序並縮短建置時間。AWS Secrets Manager 會保留佈建設定檔、私有金鑰和憑證。
6. 建置成品會交付至 Amazon S3，其會傳送成功或失敗的通知。 會 AWS Device Farm 啟用行動裝置的自動化測試。

雲端遊戲開發：工作站

遊戲開發通常涉及從不同位置遠端工作的分散式團隊，需要存取共用基礎設施，以及支援協作、地理上分散開發的能力。這種通往更分散式遊戲開發程序的趨勢，包括使用work-for-hire工作室和遠端工作，需要實作強大的技術和工作流程，以促進生產力、共享專業知識和敏捷的開發實務。

下列參考架構示範如何使用 託管 AWS 使用 Amazon DCV 通訊協定的遠端遊戲開發工作站。



使用 Amazon DCV 從任何地方串流遊戲開發

1. Amazon DCV 是一種支援 4K, 60-FPS串流的串流通訊協定。使用瀏覽器的開發人員透過 TCP 連線進行連線，而桌面用戶端可以透過連接埠 8443 使用 QUIC UDP 以提高效能。
2. 開發人員使用 AWS Client VPN 與具有來源網路位址轉譯 (SNAT) 的工作站子網路中的網路介面進行安全連線。
3. Amazon Route 53 為 VPC 中的資源以及傳入和傳出 DNS 轉送提供私有 DNS。
4. Directory Service 提供受管 Microsoft Active Directory，以啟用映射至個別使用者的本機遊戲專案儲存體。
5. 使用使用 Image Builder 建置的 Amazon Machine Image (AMI) 建立工作站。影像包括 Amazon DCV Server、開發人員軟體、登錄檔變更，以及 NVIDIA 遊戲驅動程式或周邊驅動程式等驅動程式。 AWS Marketplace 包括用於工作站AMIs。

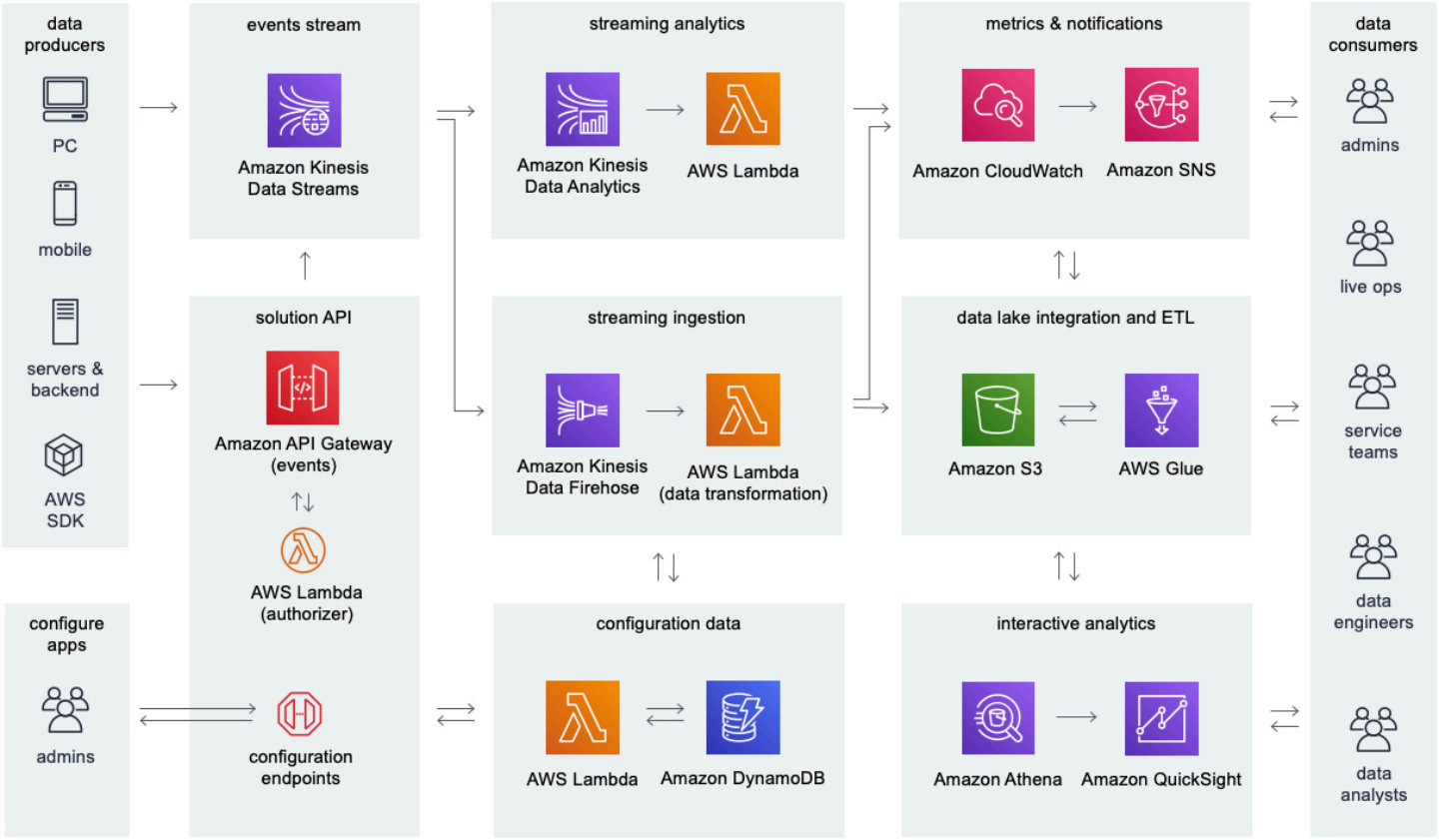
6. 工作站機群使用圖形 Amazon EC2 執行個體類型提供 GPUs，並使用 EC2 Auto Scaling 群組進行擴展。
7. Amazon DCV Session Manager Broker 可啟用 Amazon DCV 工作階段的管理。
8. 專案的本機檔案儲存體託管於 Amazon FSx for Windows File Server。開發人員會從本機儲存推送至來源控制，以遞交至個別的 CI/CD 管道。

遊戲分析管道

遊戲開發人員越來越正在尋找更了解玩家行為的方法，以便他們可以改善遊戲體驗，以保留和成長玩家群。遊戲分析代表了解和分析遊戲和相關服務所產生的資料所需的技術基礎設施和程序。這通常需要使用可支援此end-to-end程序的分析管道架構，例如 Game Analytics 管道解決方案實作。

遊戲分析架構具有下列特性：

- 資料來源會以 JSON 等常見格式傳送資料，通常包括遊戲伺服器和遊戲後端服務，以及遊戲用戶端，包括 PC、行動裝置和遊戲主控台。
- 遊戲分析管道會自動化擷取和儲存原始資料的整個工作流程，並將其處理為可用的輸出格式，以便資料消費者能夠以有效且經濟實惠的方式進行分析，例如最終使用者和分析應用程式。
- 遊戲分析管道支援擷取和處理大量即時資料，以隨著遊戲成長進行擴展。
- 支援即時和批次報告使用案例。例如，即時操作團隊通常會使用即時儀表板和警示來監控遊戲基礎設施和玩家行為，以偵測問題。資料分析師團隊通常依賴必要的和批次報告來了解一段時間的趨勢。



遊戲遙測的無伺服器遊戲分析管道

遊戲資料會從遊戲用戶端、遊戲伺服器和其他應用程式擷取。串流資料會擷取至 Amazon S3，以進行資料湖整合和互動式分析。串流分析會處理即時事件並產生指標。資料消費者分析 Amazon CloudWatch 中的指標資料和 Amazon S3 中的原始事件。

- 解決方案 API 和組態資料：使用 Amazon API Gateway 提供 REST API，以使用 Lambda 函數管理遊戲分析管道，並將組態資料儲存在 Amazon DynamoDB 中。您可以在此 API 之上建置內部入口網站或自訂命令列界面進行管理。REST API 也提供伺服器身分驗證，用於從資料來源擷取遊戲體驗資料，並將遙測資料轉送至 Amazon Kinesis Data Streams，以進行即時處理和擷取至儲存體。
- 事件串流：Amazon Kinesis Data Streams 會從遊戲擷取串流資料，並允許 Amazon Data Firehose 和 Amazon Managed Service for Apache Flink 進行即時資料處理。
- 串流分析：Managed Service for Apache Flink 會分析來自 Kinesis Data Streams 的串流事件資料，並可以使用 Lambda 函數產生發佈至 CloudWatch 的自訂指標和提醒。
- 指標和通知：使用 Amazon CloudWatch 監控解決方案的指標、日誌和警示。使用 Amazon SNS 將通知傳送給待命工程師和其他資料消費者。
- 串流擷取：使用 Firehose 從 Kinesis Data Streams 取用串流資料，並將其交付至 Amazon S3 中的資料湖，以便長期儲存、轉換和與其他資料整合。

- 資料湖整合和 ETL： AWS Glue 用於 ETL 處理工作流程，並在 中組織中繼資料 AWS AWS Glue Data Catalog，這為資料湖提供了與彈性分析工具整合的基礎。
- 互動式分析：最終使用者可以使用 Amazon Athena 對存放在 Amazon S3 中的資料集執行臨機操作互動式查詢，而 Quick Suite 可用於建置儀表板。

請參閱 Game Analytics 管道，了解可使用 部署至您帳戶的分析管道的自動參考實作 AWS CloudFormation。

定義

AWS Well-Architected Framework 以六大支柱為基礎：卓越營運、安全性、可靠性、效能效率、成本最佳化和永續性。 AWS 提供多個核心元件，可讓您為遊戲工作負載設計state-of-the-art架構。在本節中，我們將介紹金鑰定義的概觀。

基於本文的目的，遊戲架構包含建置和操作遊戲所需的後端技術基礎設施。有些遊戲可能沒有社交、多玩家或其他線上功能，而且可能不需要使用本文所述的後端技術基礎設施的某些層面。如需經常部署以支援遊戲架構之不同類型的工作負載的詳細討論，請參閱案例。

AWS 雲端 基礎設施是以區域和可用區域為基礎建置。

- 區域是世界上一個實體位置，我們擁有多個可用區域。
- 可用區域由一或多個離散資料中心組成，每個資料中心都具有備援電源、聯網和連線能力，並存放在不同的設施中。

根據遊戲的特性，您可能想要將遊戲架構的特定元件部署到多個區域，例如改善玩家的效能，或根據玩家的位置為玩家提供自訂體驗。

遊戲有許多不同類型的遊戲，支援遊戲所需的後端技術基礎設施會因開發的遊戲類型而有所不同。例如，熱門的遊戲類型可能包括第一人稱射擊遊戲 (FPS)、角色遊戲 (RPG)、大量多玩家線上遊戲 (MMOG)、戰鬥角色 (BR)、運動遊戲、拼圖遊戲等。也有不同的遊戲互動模式會影響遊戲的架構，例如輪換式和同時播放，具有不同的效能特性。

遊戲的開發是在一或多個遊戲系統上進行，包括桌上型電腦、Web、行動裝置、主控台和較新的互動模式，例如擴增實境 (AR)、虛擬實境 (VR) 和遊戲串流解決方案。遊戲通常支援跨系統遊戲，這表示玩家可以儲存其遊戲進度，並在其他系統上繼續遊戲，以及與其他系統上的玩家啟動遊戲工作階段。

影片遊戲獲利可讓遊戲發行者使用不同的策略來產生收入，例如廣告、數位和零售遊戲購買、遊戲內購買稱為微交易的可下載內容 (DLC)，以及透過必要的付費訂閱來玩遊戲。遊戲產業中最常見的一些關鍵效能指標 (KPIs) 包括：

- 每日作用中使用者 (DAU)
- 每月作用中使用者 (MAU)
- 並行使用者 (CCU)
- 工作階段持續時間
- 每次安裝成本 (CPI)

- 玩家生命週期值 (LTV)
- 每個使用者的平均營收 (ARPU)

遊戲系統

視訊遊戲是在提供用戶端輸入控制、圖形、用戶端軟體（稱為遊戲用戶端）和硬體的遊戲系統上開發的，在某些情況下，還有支援遊戲的系統專屬功能。

遊戲系統通常分為以下類別：

- 主控台：專為遊戲設計的專用娛樂系統，包括熱門範例，例如 Sony PlayStation、Microsoft Xbox 和 Nintendo Switch。主控台透過將實體或數位分散式遊戲內容安裝到遊戲系統供應商製造的主控台硬體上，提供遊戲的能力。在此定義中，主控台可以是手持或固定式，且旨在用於家庭娛樂案例。
- 個人電腦 (PC)：使用安裝在可由玩家自訂之用戶端機器上的電腦軟體播放的遊戲。因此，PC 遊戲在玩家之間很熱門，因為它提供彈性和控制。
- Web：旨在使用 Web 瀏覽器播放的遊戲，這通常有利於讓玩家無論其作業系統如何都能存取遊戲。
- 行動裝置：開發為在行動電話上播放的遊戲，通常是智慧型手機作業系統。行動遊戲通常從數位應用程式存放區下載，並安裝在手機上。

除了先前提到的系統之外，還有一些較新且持續成長的系統，相較於較主要的系統，其市佔率要小得多。此類別中的遊戲系統範例包括 AR、VR 和遊戲串流，有時稱為雲端遊戲。

遊戲串流涉及在雲端渲染遊戲體驗，以及串流到精簡型用戶端，通常是瀏覽器。遊戲串流可讓玩家玩完全由遠端託管的遊戲，通常是由遊戲串流服務供應商在雲端。在遊戲串流中，玩家會透過瀏覽器或雲端遊戲服務供應商（遊戲系統）提供的精簡型用戶端連線至雲端遊戲。

遊戲伺服器

遊戲伺服器代表遊戲運算基礎設施最重要的層面之一。遊戲伺服器有時稱為專用遊戲伺服器，會在開發多玩家遊戲或需要伺服器授權處理遊戲事件時使用。遊戲伺服器位於遊戲架構的中心，做為核心邏輯執行的位置，其中包括管理玩家和遊戲狀態，以及管理連線遊戲用戶端和遊戲伺服器之間的互動。遊戲伺服器通常是遊戲架構中效能最敏感的層面之一，因為它負責處理玩家遊戲用戶端的輸入，並即時將其正確分發給其他連線的玩家。效能不佳的遊戲伺服器會影響遊戲體驗的整體效能。因此，您應該最佳化遊戲伺服器效能並提供足夠的容量，尤其是在遊戲啟動或尖峰遊戲期間。

基於本文件的目的，遊戲伺服器或遊戲伺服器執行個體是指託管一或多個遊戲伺服器程序的運算，例如虛擬機器。遊戲伺服器程序代表遊戲伺服器組建的單一執行個體託管遊戲工作階段，這是您執行中遊戲

的執行個體，玩家可以透過玩家工作階段連線到此執行個體。因此，由於遊戲工作階段與託管遊戲伺服器程序之間的隱含一對一關係，我們通常會互換地參考遊戲伺服器程序或遊戲工作階段。在 中 AWS，有多個運算選項可託管遊戲伺服器，可透過彈性佈建資源來存取可擴展的雲端容量。

Amazon EC2 提供雲端型虛擬伺服器，稱為執行個體，並支援多個版本的 Linux 和 Windows。您可以像其他伺服器或虛擬機器一樣，直接建立執行個體並進行管理。一般而言，多個遊戲伺服器程序會部署到執行個體，以提高效率並降低成本。如果您想要對運算基礎設施進行最大控制，Amazon EC2 是遊戲伺服器的最佳選擇。

Amazon GameLift 為雲端中的專用遊戲伺服器託管提供全受管解決方案，以及其他功能，例如使用 GameLift FlexMatch 進行配對。GameLift 在 Amazon EC2 之上提供一層抽象，讓遊戲伺服器管理變得簡單明瞭，並可在大多數 中使用， AWS 區域 因此您可以託管靠近玩家的遊戲伺服器，以減少延遲、實現高可用性，並使用 Spot 執行個體大幅降低成本。雖然 GameLift 可以整合到現有的遊戲後端，但對於不想開發自己的遊戲伺服器管理和配對解決方案，並偏好由 管理 AWS 且可以隨著遊戲成長擴展的解決方案的遊戲開發人員來說，此功能特別有用。

Amazon Elastic Container Service (Amazon ECS) 是一種全受管容器協同運作服務，可用來執行 Docker 型容器。您也可以使用 Amazon Elastic Kubernetes Service (Amazon EKS) 來執行使用 Kubernetes 建置的 Docker 型容器。使用 Amazon ECS 和 Amazon EKS 提供的容器技術，可透過將許多遊戲伺服器程序或其他遊戲應用程式執行個體有效地封裝到 EC2 執行個體，協助您改善運算使用率。

使用容器也可以透過使用開發人員在開發期間本機機器上使用的相同 Docker 映像操作執行時間來託管應用程式，從而提高開發人員的生產力。您可以使用 進一步降低營運開銷 AWS Fargate，這是用於執行容器的無伺服器運算解決方案，並且與 Amazon EKS 和 Amazon ECS 相容。Fargate 最適合您想要在容器中執行遊戲伺服器的使用案例，無需負責操作容器執行的基礎執行個體。

您可以使用 在資料中心或內部部署設施中 AWS Outposts 執行 AWS 基礎設施和服務，這可讓遊戲在內部部署環境中執行，並使用 AWS 相同的基礎設施來支援混合雲端採用策略。 AWS 本機區域做為 的延伸 AWS 區域 ，可讓您的遊戲伺服器和其他延遲敏感工作負載更接近您的玩家或開發團隊。此外，若要降低遊戲伺服器的全域網路延遲，您可以使用 AWS Global Accelerator 來改善玩家流量到遊戲伺服器的效能。

AWS Lambda 是一種無伺服器運算服務，可在不佈建或管理伺服器的情況下執行程式碼，因此適用於非同步遊戲伺服器使用案例，例如輪換型遊戲或具有輕量型運算需求的遊戲、小型程式碼庫，以及可使用無狀態微服務架構設計遊戲功能。請務必記住，Lambda 函數是以事件驅動的每次請求為基礎執行，而不是執行長時間執行的遊戲伺服器程序。Lambda 提供本白皮書中選項執行時間最多的摘要，因為基礎應用程式可供開發人員選擇託管程式碼。

選擇遊戲伺服器託管的方法時，請考慮各種需求，包括操作開銷、舊版程式碼庫、效能需求和規模。EC2 執行個體和容器是舊版程式碼庫的良好選項，因為它們需要最小的變更才能移至雲端，而且您可以使用 EC2 執行個體來專用運算執行個體的資源，而容器可以讓管理和高使用率更容易實現。無伺服器函數提供最高層級的抽象，可用來定義僅回應事件而執行的程式碼，進而降低成本。

遊戲用戶端

遊戲用戶端代表玩家用來玩遊戲的軟體和硬體裝置。遊戲用戶端提供軟體，將玩家的輸入轉換為傳送至伺服器進行處理的訊息，並負責處理來自伺服器的傳入回應，並將圖形等輸出轉譯給玩家。在即時網路多玩家遊戲中，遊戲用戶端通常會在遊戲工作階段期間維持與遊戲伺服器的持久性網路連線，以減少網路延遲並將處理時間降至最低。不過，遊戲用戶端也可能使用 REST 與遊戲伺服器或後端服務互動。

簡訊

遊戲中通常有三種主要訊息類別：

- 針對特定使用者或使用者群組的玩家參與訊息，例如遊戲邀請或推播通知
- 玩家之間的群組傳訊，例如遊戲內聊天
- Service-to-service傳訊，例如用於整合兩個或多個應用程式的 JSON 訊息

傳送和接收這類訊息的常見策略是使用發佈者訂閱者和非同步處理架構模式。 AWS 提供多種服務，可協助您在遊戲中實作訊息。

- Amazon Simple Notification Service (SNS)：使用 pub/sub 架構模式在發佈者和訂閱者之間傳遞訊息的受管服務。發佈者使用 API 將訊息傳送到 Amazon SNS，以非同步方式將訊息傳送到訂閱應用程式，並且可以直接將推播通知傳送到行動用戶端或桌面，並支援一些最常用的推播通知服務。Amazon SNS 可用於推送通知給用戶端，以及service-to-service傳訊使用案例。
- Amazon Simple Queue Service (SQS)：全受管佇列服務，無論每個 中使用的程式設計語言為何，都能直接整合遊戲伺服器和您的遊戲。許多遊戲任務可以在背景解耦和處理，例如更新資料庫中的排行榜或播放時間值。這種方法是一種有效的方法，可分離遊戲的各個部分，並從後端處理獨立擴展面向玩家的功能。
- Amazon Managed Streaming for Apache Kafka (MSK)：一種全受管服務，使用熱門的開放原始碼解決方案 Apache Kafka，簡化建置資料串流和生產者或消費者應用程式。Kafka 通常用於擷取和處理即時串流資料，並可用於service-to-service傳訊。

- Amazon ElastiCache (Redis OSS)：提供全受管記憶體內資料存放區，其中包含對 Redis 熱門 pub/sub 功能的支援，該功能通常用於開發聊天室應用程式和高效能service-to-service簡訊。Redis 也支援豐富的資料類型，例如清單和集，以便開發人員可以使用 Redis 進行高效能佇列。
- Amazon Pinpoint：透過電子郵件、簡訊、語音和推送通知提供使用者互動訊息。例如，Amazon Pinpoint 可用來將使用者參與訊息傳遞給玩家，邀請他們返回遊戲，並可用於交易使用案例，例如支援多重驗證字符、訂單確認和密碼重設電子郵件。

即時遊戲操作（即時操作）

即時遊戲操作（即時操作）是一種遊戲管理和操作風格，可將遊戲視為即時服務，並持續為已啟動的遊戲提供新功能、更新、促銷、遊戲內事件和改進，以改善玩家社群的體驗。

傳統上，遊戲以產品而非 服務的形式交付，而新內容和功能經常納入後續版本或後置版本，而非已啟動的產品。透過遊戲管理的即時操作方法，遊戲操作團隊可以透過實驗、提升、遊戲內活動和創新來啟動遊戲並維護參與的玩家社群，讓玩家享受娛樂。

雖然這種方法具有解鎖新玩家參與策略和交付經常性收入串流的優勢，但它需要更多的操作專業知識。例如，若要實作成功的即時營運策略，開發人員可能需要與雲端服務整合或操作自己的後端技術基礎設施。他們還需要一種有效的方法來識別和回應遊戲或玩家社群中出現的問題，這些問題可能會對玩家體驗產生負面影響。

卓越營運

卓越營運支柱著重於大規模部署和操作雲端遊戲的最佳實務。請務必專注於卓越營運，以維持正面的玩家體驗，並實作預防措施，以準備並從影響其體驗的問題中復原。

重點領域

- 設計原則
- 即時操作
- 帳戶結構
- 遊戲部署
- 運作狀態監控
- 負載測試
- 隨時間優化
- Resources

設計原則

除了 Well-Architected Framework 白皮書的設計原則之外，下列設計原則可協助您在建置和操作遊戲方面實現卓越營運：

- 為遊戲操作團隊定義可衡量且可實現的目標，並視需要進行調整：由於遊戲的命中驅動性質，很難預先判斷在遊戲啟動時有多少玩家會玩遊戲，或玩家會對您持續的遊戲操作有何期望。請務必與利益相關者一起設定遠大但可實現的目標，並設計可在遊戲超過投影和縮減規模時擴展的方法，同時讓遊戲開發團隊最佳化玩家體驗。事先做好充分準備和測試，以符合這些要求，並讓您的業務和技術利益相關者符合操作遊戲的目標目標。定義目標後，遊戲團隊可以在規劃、設計、佈建、測試、部署和操作遊戲後端基礎設施時，在成本和效能之間取得適當的平衡。
- 使用操作 Runbook 來規劃與遊戲啟動和特殊事件相關的擴展活動：遊戲操作團隊應與業務利益相關者協調，為事件的預期尖峰玩家並行建立投影模型，並執行主動規劃，提前預先擴展基礎設施容量。由於活動期間玩家流量的波動性質，先前的規劃和預先擴展活動應擴大您現有的自動化擴展系統，以提高您在活動期間成功的機會，並確認您有足夠的資源可提供正面的玩家體驗。實作效能工程實務，以開發 資源的基準，以及對系統容量的資料驅動型了解，這將有助於引導預先擴展活動和自動化擴展組態。開發營運 Runbook 以在程序中提供一致性。此進階規劃和對玩家需求的回應對於即時服務遊戲特別重要，因為即時服務遊戲必須維持可靠的效能和基礎設施，才能在延長的時間內支援主動參與的玩家群。

- 建立用於接收、調查和回應玩家支援請求的操作模型：啟動後，監控遊戲的投訴和問題報告。實作適當的系統，以安全且有效的方式與玩家互動，以充分解決玩家問題，例如社群論壇、社交媒體、電子郵件、票證系統、呼叫中心或自動化聊天機器人解決方案。這對於即時服務遊戲來說特別重要，因為即時服務遊戲需要與玩家群持續通訊、回應玩家意見回饋以因應不斷變化的需求，以及在更長的生命週期內維護參與社群。

即時操作

GAMEOPS01：如何定義遊戲的即時操作 （即時操作） 策略？

在諮詢業務利益相關者後，根據定義的目標和效能指標，為您的遊戲制定即時操作 （即時操作） 策略。

最佳實務

- GAMEOPS01-BP01 使用遊戲目標和業務效能指標來開發您的即時操作策略

GAMEOPS01-BP01 使用遊戲目標和業務效能指標來開發您的即時操作策略

諮詢商業利益相關者，例如遊戲生產者和發佈合作夥伴，以確定遊戲的目標和效能指標。這可協助您制定如何管理遊戲的計劃，包括定義維護時段、軟體和基礎設施更新排程，以及系統可靠性和可復原性目標。

未建立此最佳實務時的曝險等級：高

實作指引

這些指標也可協助您判斷您應該在遊戲生命週期的哪個階段納入即時操作蒸汽 （即時操作），以監控遊戲運作狀態、收集直接遊戲意見回饋，以及建置簡化的自動發行程序。例如，在設定專用即時操作團隊之前，新遊戲可能會等待達到特定規模，以作用中玩家計數、收入或其他一組指標進行測量。已建立的遊戲開發工作室可能已經有即時操作經驗，可能適用於其他遊戲，因此他們只需要加入新遊戲。

實作步驟

- 您可以定義玩家並行 (CCU) 和遊戲基礎設施應能夠有效支援之每日和每月作用中使用者 (DAU 和 MAU) 的目標、基礎設施預算、財務目標和其他效能目標，例如內容和功能發佈的頻率，以提高玩家參與度。這些目標和指標會納入有關遊戲設計、版本管理、可觀測性和有效操作所需支援的決定。

- 您的遊戲可能有一個目標是至少每個月發行一次新的內容更新，在發行期間不會停機。此資訊可協助您定義發行部署策略，並協調在一個月內其他時間可能需要停機時間的必要維護排程，並對您的可用性 SLA 做出貢獻。

帳戶結構

GAMEOPS02：如何建構 AWS 帳戶 以託管遊戲環境？

實作多帳戶策略來隔離不同的遊戲環境，並增強安全性、營運效率和可擴展性。使用 AWS Organizations 來管理帳戶階層、將護欄套用至帳戶，以及在部署的資源上強制執行標籤政策和標記。

最佳實務

- GAMEOPS02-BP01 採用多帳戶策略，將不同的遊戲和應用程式隔離到自己的帳戶中
- GAMEOPS02-BP02 使用資源標記組織基礎設施資源

GAMEOPS02-BP01 採用多帳戶策略，將不同的遊戲和應用程式隔離到自己的帳戶中

設計可引導基礎設施部署的帳戶結構，以符合每個環境的安全、隔離和操作需求。透過限制存取環境並只允許在其中使用必要的 AWS 服務來進行環境隔離至關重要，因為生產環境會遭到鎖定，而開發和測試環境則足以允許實驗。強烈建議進一步隔離每個環境中的主要子系統，以及由多個環境用來自行 AWS 帳戶 託管和管理的常見服務。

未建立此最佳實務時的曝險等級：高

實作指引

將不同環境（例如開發、測試、預備、生產和共用服務）AWS 隔離至個別環境，藉此在 上採用多帳戶策略 AWS 帳戶，進而減少事件範圍。 AWS Organizations 考慮集中管理 的階層 AWS 帳戶 ，以進一步簡化操作，以及選擇性地定義和套用帳戶層級和組織單位層級 (OU 層級） 政策。透過設計符合您開發和生產工作流程需求的適當 OU 和 AWS 帳戶 結構，您可以最佳化成本並增強可擴展性。

- 採用多帳戶策略：隔離環境以減少事件半徑並簡化操作。
- 使用 AWS Organizations：以階層方式管理帳戶、套用政策，以及啟用集中式控管。
- 針對可擴展性進行規劃：設計精細的帳戶結構，並實作節省成本的措施，以因應未來的成長。

實作步驟

部署於 的遊戲系統 AWS 應使用多個邏輯組織的帳戶來提供適當的隔離，這可降低問題的爆量半徑，並在您的遊戲基礎設施擴展時簡化操作。 AWS 帳戶 該主機遊戲基礎設施通常會分組為下列邏輯環境：

- 開發人員使用遊戲開發環境來開發遊戲的軟體和系統。
- 測試或品質保證 (QA) 環境用於執行整合測試、手動 QA 和其他必須執行的自動化測試。
- 預備或生產前環境用於託管已完成的軟體，以便在啟動生產之前執行負載和煙霧測試。
- 即時或生產環境用於託管即時軟體和基礎設施，並為來自玩家的生產流量提供服務。
- 共用服務或工具環境可讓您存取許多不同團隊所使用的常用系統、軟體和工具。例如，中央自我託管來源控制儲存庫和遊戲建置陣列可能託管在共用服務帳戶中。
- 安全環境用於整合集中式日誌和安全技術，供專注於雲端安全的團隊使用。

對於 上的遊戲基礎設施 AWS，建議為每個遊戲環境 （開發、測試、預備和生產） 建立單獨的帳戶，以及安全、記錄和中央共用服務的帳戶。

一般而言，管理有限數量基礎設施資源的較小遊戲開發工作室，通常為數百部伺服器或更少，可以 AWS 帳戶 為每個環境建立一個 （例如一個生產帳戶、一個開發帳戶和一個預備帳戶）。不過，隨著您的遊戲基礎設施或團隊大小隨時間增加，這個簡化的模型可能無法妥善擴展。

設定這些環境時，請考慮許多 AWS 服務會為特定區域內的整個帳戶共用資源和 API 層級 Service Quotas。在判斷如何以邏輯方式組織帳戶時，必須考量這一點。 AWS 帳戶 只會產生使用部署到其中之服務的成本。因此，這提供了一種有效減少資源爭用和服務配額的方法，特別是隨著您的遊戲增加，並且更多開發人員需要建立和管理資源的存取權。

根據我們處理大型遊戲開發工作室的經驗，這些工作室通常操作數千部伺服器，讓數百名開發人員存取資源，我們建議您設計更精細的帳戶結構，其中支援遊戲的個別應用程式擁有自己的開發、測試、預備和生產帳戶。由於由於規劃和遷移即時系統的複雜性，在您啟動遊戲後重新設計 AWS 多帳戶策略相當困難且耗時，因此在決定正確的多帳戶結構時，請考慮您未來的擴展需求。

您可以使用 AWS Organizations來設定階層和分組 AWS 帳戶，並定義組織單位 (OUs)，以透過服務控制政策 (SCPs) 將常見的 OU 層級政策套用到它們。隨著資源的成長和擴展， 會 AWS Organizations 集中管理和控管您的環境。您可以以程式設計方式建立新帳戶並配置資源、將帳戶分組以組織您的工作流程、將政策套用至帳戶或群組以進行控管，以及使用您帳戶的單一付款方式簡化帳單。此外，Organizations 與其他 服務整合，因此您可以定義組織中帳戶間的中央組態、安全機制、稽核需求和資源共用。

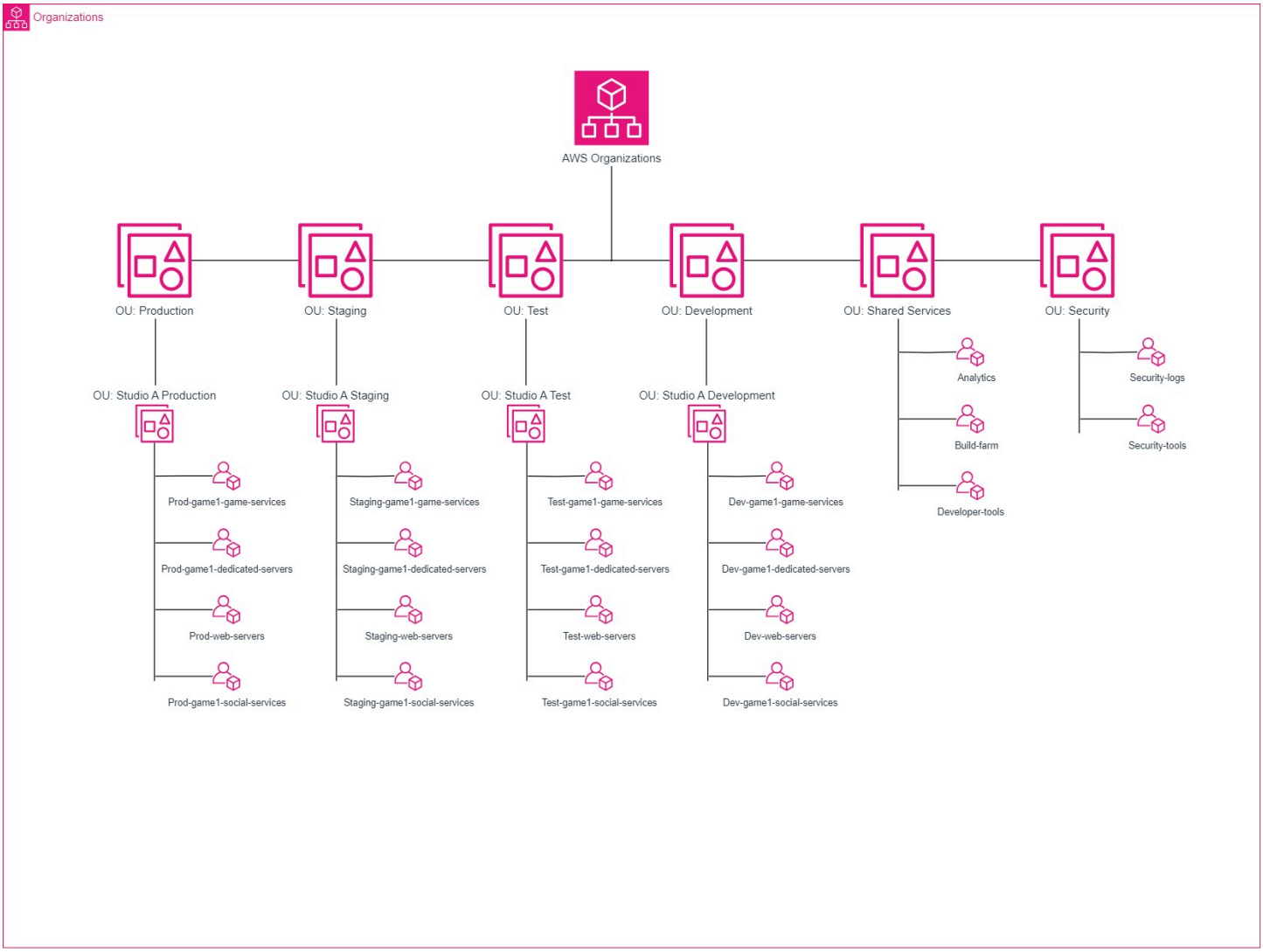
AWS Control Tower 提供直接的方式來設定和管理稱為登陸區域的安全多帳戶環境。Control Tower 會使用 建立您的登陸區域 AWS Organizations，根據數千位客戶在遷移至雲端時 AWS的經驗，提供持續的帳戶管理和控管，以及實作最佳實務。、 AWS Trusted Advisor和 AWS Config AWS Security Hub CSPM是提供帳戶衛生彙總或集中檢視的服務。

此隔離可協助您設定每個遊戲環境的自訂或個別許可和護欄。生產帳戶應具有必要的護欄、存取限制、監控和提醒，以及安全工具，而非生產帳戶可能不需要相同層級的護欄和許可。非生產環境可以在數小時後自動關閉資源並節省成本。以這種精細程度分隔帳戶，可以直接監控支援遊戲的每個環境的基礎設施成本。

以下是遊戲公司使用 AWS Organizations 和組織單位 (OUs) 以邏輯方式分組 AWS 帳戶 到不同環境和工作室的多帳戶結構範例。在此範例中，OUs 用於根據帳戶的環境分組，然後根據操作環境的工作室分組帳戶。這示範了如何建立巢狀階層，以允許將個別應用程式和遊戲部署到其環境中自己的帳戶（顯示為 OUs)，這在您開發和操作多個遊戲時非常有用。請參閱本支柱資源一節所提供的文件和白皮書，以了解您可以考慮用於組織多帳戶策略的其他策略。

根據上述討論，以下範例圖表假設遊戲工作室（組織）具有由 4 個階段（開發、測試、預備和生產）組成的開發管道。對於指定的遊戲 (game1)，每個環境 (OU) 都有個別 AWS 帳戶 的遊戲服務、專用遊戲伺服器、社交服務和 Web 伺服器。每個 中執行的資源 AWS 帳戶 都與個別子系統相關。一般而言，每個使用這種開發管道的個別遊戲都會為其複寫此結構或類似結構 AWS 帳戶。

除了這些以遊戲為中心的環境 OUs 之外，還有共用的服務 OU 和安全 OU。這些 OUs 應該是整個組織的，而不是針對每個個別遊戲。如此一來，遊戲就會使用開發工具、資料和分析的共用服務，如本範例所示。然後，將應用程式和系統日誌傳送至安全 OU 中為日誌 AWS 帳戶 設定的 。



遊戲環境的帳戶結構範例

GAMEOPS02-BP02 使用資源標記組織基礎設施資源

若要有效地管理和追蹤 中的基礎設施資源 AWS，請使用適當的資源標記和分組來識別每個資源的擁有者、專案、應用程式、成本中心和其他資料。標記的資源可以使用資源群組分組在一起，以協助操作支援。

未建立此最佳實務時的曝險等級：高

實作指引

定義標記政策。典型的策略包括用於識別資源擁有者的資源標籤，例如團隊名稱或個別名稱、遊戲、應用程式或專案的名稱、工作室名稱、環境（例如開發或生產），以及資源的角色（例如資料庫伺服器、Web 伺服器、專用遊戲伺服器、應用程式伺服器或快取伺服器）。您可以新增其他標籤，以協助

處理業務和 IT 需求。 AWS Config也可以在資源建立和更新時間強制執行標記政策。標籤和資源群組可從 AWS 管理主控台 AWS CLI、 和 API 操作取得。

實作步驟

- 標記資源以識別其擁有者、專案、應用程式、成本中心和其他相關資料。
- 實作標記政策，包括擁有者、專案、工作室、環境和資源角色的標籤。
- 使用 AWS Config 強制執行標記政策，並透過 CLI 和 API AWS 管理主控台管理標籤。

遊戲部署

GAMEOPS03：如何管理遊戲部署？

透過徹底驗證重複使用的元件、執行定期效能工程，以及在整個開發生命週期實作定期負載測試，來管理遊戲部署。

最佳實務

- GAMEOPS03-BP01 在遊戲中重複使用現有核心遊戲系統和基礎設施之前，先進行驗證和測試
- GAMEOPS03-BP02 在每次發行之前執行效能工程 （或至少針對主要版本）
- GAMEOPS03-BP03 提早且經常載入測試
- GAMEOPS03-BP04 採用部署策略，將對玩家的影響降至最低
- 支援尖峰需求所需的 GAMEOPS03-BP05 預先擴展基礎設施

GAMEOPS03-BP01 在遊戲中重複使用現有核心遊戲系統和基礎設施之前，先進行驗證和測試

組織傾向於重複使用先前遊戲的現有元件和原始程式碼，以節省開發時間和成本。這些舊版元件和程式碼可能不會受到徹底審查，或具有詳細的整合測試，而是依賴其過去的效能。

未建立此最佳實務時的曝險等級：高

實作指引

雖然重複使用有助於提高生產力，但它也可能帶來將過去的效能和穩定性問題重新引入新專案的風險。因此，在重複使用先前遊戲的現有元件和原始程式碼時，應實作強大的測試。

實作步驟

- 識別重複使用的程式碼和元件：為先前遊戲重複使用的原始程式碼、程式庫和元件編製目錄。明確區分主動維護和淘汰的程式碼
- 記錄原始行為和已知問題：記錄與重複使用元件相關聯的原始效能特性、功能限制和已知錯誤或生產事件。
- 執行徹底的程式碼審查：對重複使用的元件進行詳細的技術審查，特別是過去發生問題或文件記錄不佳的元件。
- 取代或重構高風險舊版元件：優先取代或更新具有問題歷史記錄或不再可維護的舊版元件，而不是依賴生產中的解決方法。
- 執行整合和相容性測試：在新遊戲系統的環境中驗證重複使用的元件。確認它們與新模組、工具和APIs正確互動。

GAMEOPS03-BP02 在每次發行之前執行效能工程 （或至少針對主要版本）

效能工程是監控應用程式多個關鍵操作指標的程序，以探索可進一步改善應用程式效能的最佳化機會。這是從測試開始的反覆程序，接著最佳化程式碼、其相依性、相關聯的程序、其主機作業系統和基礎基礎設施。

未建立此最佳實務時的曝險等級：高

實作指引

若要對應用程式效能進行更深入的分析，請在應用程式程式碼中整合應用程式效能監控 (APM) 或偵錯工具，透過追蹤應用程式流程中的異常行為來隔離問題並縮短故障診斷時間。APM 工具也能夠識別執行緩慢的方法和外部操作。

AWS X-Ray 協助開發人員進行效能工程活動，例如識別效能瓶頸，以及分析和偵錯生產錯誤。您可以使用 X-Ray 來了解應用程式及其基礎服務的效能，以及識別和疑難排解效能問題和錯誤的根本原因。透過數輪的負載測試，應用程式及其基礎設施會逐漸載入合成玩家流量、各種系統瓶頸、應用程式錯誤、例外狀況、作業系統問題，以及其他在其他 QA 測試期間可能尚未發現的問題。

對於遊戲啟動、內容發行、促銷和主要遊戲內事件等重要事件，請使用 AWS Countdown，根據遊戲專家建置的手冊提供實作指導，以驗證操作準備程度、降低潛在風險，以及規劃容量需求。AWS Countdown 也有進階支援選項，可提供增強的支援和選項，例如工程師來最佳化您的基礎設施。

實作步驟

- 效能工程涉及評估和監控關鍵操作指標，以驗證應用程式的程式碼、程序、作業系統和基礎設施是否如預期般運作。生產前檢閱也有助於定義不同層級模擬用量的基準效能。
- 使用 sar、top、vmstat、sysstat、netstat 和 Performance Monitor 等系統工具，探索和追蹤使用率、服務、I/O、程序等關鍵指標。
- 使用 等 APM 工具來追蹤應用程式的效能和行為， AWS X-Ray 以隔離問題、識別瓶頸和偵錯生產錯誤。
- 對於遊戲啟動等重要事件，請 AWS 訂閱 Countdown (IEM) 以取得架構和操作指導、隨需操作支援，以及識別風險和計劃緩解措施。

GAMEOPS03-BP03 提早且經常載入測試

負載測試是在系統上模擬真實流量的程序，以評估其可靠性和效能。

未建立此最佳實務時的曝險等級：高

實作指引

負載測試是為您的資源開發效能基準和了解系統容量的關鍵因素，可以引導財務預測、架構設計、資源配置、自動化擴展組態和啟動後預先擴展活動。其他優點包括：

- 最佳化基礎設施：資源可能過度佈建或佈建不足。了解所需的資源將可降低成本，並減少要管理的基礎設施。
- 可擴展性準備：某些機制和功能可讓使用者快速進入遊戲。知道何時和如何擴展可能是適當滿足增加的需求和失去玩家之間的差異。使用負載測試結果來準備具有不同擴展層級的系統閾值、警示點和關鍵警示點的 Runbook。
- 更高品質的程式碼：例如服務之間過度的串擾、資料庫呼叫不中斷、效率低的演算法、記憶體流失和服務降級等問題，有時更容易大規模識別。
- 行為驗證：將不同類型的失敗注入您的測試可以驗證系統的預期行為，或發現需要更正的錯誤處理問題。

理想情況下，開發人員應該在整個開發過程中的多個時間點執行負載測試，因為每個時間點都可能產生不同的優勢：早些時候，他們會引導架構決策和重構工作，同時更便宜且直接地進行變更。在每次衝刺或反覆運算結束時，他們會使用最新的功能來驗證應用程式的效能。

在部署到生產環境之前，大規模負載測試模擬預期的實際使用模式，可確認系統處理生產工作負載的能力。部署之後，定期負載測試會監控系統效能，並識別隨著時間推移可能發生的變更或瓶頸。

若要模擬玩家流量，您需要輕量型用戶端或機器人來模擬遊戲用戶端流程，並與遊戲後端交易，以模擬真實世界的玩家行為。此資料通常透過遊戲日誌和人類驅動的 QA 測試產生的資料擷取，以及透過真實世界的有限擴展 Alpha 或 Beta 測試擷取，其中邀請真實玩家玩遊戲的早期存取組建。

請務必在操作 Runbook 中記錄系統的行為，以協助故障診斷未來的可能故障，並保留未來負載測試可以比較的效能指標。也建議人類 QA 人員在載入測試時測試遊戲，因為他們可能會發現機器人無法識別且指標無法反映的問題。

AWS Fault Injection Service 是一項全受管服務，可執行故障注入實驗，以直接改善應用程式的效能、可觀測性和彈性。故障注入實驗用於混沌工程，這是透過建立破壞性事件在測試或生產環境中對應用程式施加壓力的做法，例如 CPU 或記憶體消耗突然增加、觀察系統回應的方式，以及實作改進。故障注入實驗可協助團隊建立所需的實際條件，以發現分散式系統中難以發現的隱藏錯誤、監控盲點和效能瓶頸。

實作步驟

- 使用 Kubernetes-Bases 遊戲負載測試指南來設定分散式負載測試環境。
- 使用提供的部署檔案，在 EKS 叢集內自訂和部署 Locust 控制和工作者 Pod，實現可擴展且可管理的負載產生。
- 在操作 Runbook 中記錄負載測試期間的系統行為和指標，以協助未來故障診斷並建立效能基準。
- 使用錯誤注入實驗來模擬真實世界的中斷，並發現系統效能、可觀測性和彈性方面的隱藏問題。

GAMEOPS03-BP04 採用部署策略，將對玩家的影響降至最低

為您的遊戲軟體和基礎設施整合部署策略，將讓玩家無法進入遊戲的停機時間降至最低。雖然某些類型的更新可能需要在遊戲用戶端安裝新的更新，但請設計遊戲，以盡可能減少或避免在部署期間停機。

未建立此最佳實務時的曝險等級：高

實作指引

開發遊戲部署策略時需要考慮的最重要步驟之一，就是判斷如何管理遊戲基礎設施。使用基礎設施即程式碼 (IaC) 工具管理遊戲基礎設施，例如 AWS CloudFormation或 Hashicorp 的 Terraform，以減少環境準備期間的人為錯誤。基礎設施範本可以在自動化管道中部署和測試，這可在不同遊戲環境的組態中建立一致性。

有數種部署策略可用於遊戲：

滾動替換

部署的滾動替換主要目標是執行版本，而不關閉遊戲，也不會影響玩家。要執行的升級或變更必須回溯相容，且將與先前版本的系統相鄰運作。

在此部署中，執行更新版本的執行個體會逐步取代（取代或推出）伺服器執行個體。此滾動替換可以透過幾種不同的方式執行。例如，若要對專用遊戲伺服器機群實作滾動更新，典型的方法是建立新的 EC2 執行個體 Auto Scaling 群組，其中包含部署到這些執行個體的新遊戲伺服器建置版本，然後逐步將玩家路由到這個新伺服器機群上託管的遊戲工作階段。如果有相關聯的遊戲用戶端更新是使用新遊戲伺服器建置的必要條件，則您必須包含驗證檢查，以確認只有安裝了此新遊戲用戶端更新的玩家才能路由到這些遊戲工作階段。

包含舊遊戲伺服器建置版本的伺服器機群（例如 EC2 Auto Scaling 群組）只有在以正常方式耗盡作用中玩家工作階段後才會從服務中移除，通常是透過設定個別化伺服器指標來允許遊戲操作團隊自動化此程序。或者，若要減少執行滾動部署的基礎設施量和時間，可以執行替代方法，其中現有生產執行個體從服務中移除、使用新的遊戲伺服器建置進行更新，然後放回生產機群。這種方法可減少所需的基礎設施數量，但也會增加風險，因為隨著伺服器被取代，玩家可用的即時遊戲伺服器數量也會減少。

此模型也可用於對後端服務執行滾動部署，例如未託管遊戲的資料庫、快取和應用程式伺服器。只要這些服務在多個叢集執行個體中以高可用性的方式部署，那麼部署到這些服務的複雜性就應該低於部署到專用遊戲伺服器。

藍/綠部署

遊戲中藍/綠部署的主要目標是將停機時間降至最低，同時在發現問題時允許安全回復到先前的部署。它適用於兩個版本的遊戲後端相容且可同時為玩家提供服務的部署。

在藍/綠部署策略中，會設定兩個相同的環境（藍和綠）。現有的遊戲版本會標示為藍色，而做為部署目標的新遊戲版本則會標示為綠色。當綠色環境準備好進行遷移時，您可以將路由層設定為將流量翻轉至綠色環境，同時在需要容錯回復時保持舊環境（藍色）可用。在此案例中，路由更新可能需要更新配對服務，以將其設定為開始將遊戲工作階段傳送至新機群，或者如果是遊戲後端服務，這可能會更新您的服務 Amazon Route 53 中的 DNS 記錄，或轉移應用程式負載平衡器權重以將流量傳送至新目標群組。

由於執行部署時所需的額外基礎設施，藍/綠部署策略的缺點之一是待命環境的固有成本。降低此額外基礎設施成本的選項是考慮採用藍/綠部署的變體，其中新的遊戲軟體會部署到已部署到生產環境的相同伺服器上。在此案例中，新的綠色伺服器程序可以使用新軟體和現有的藍色伺服器程序啟動，並在伺服器程序之間進行切換，而不是在單獨的實體基礎設施之間進行切換。這種方法也可以透過消除在雲端等待新伺服器啟動的需求，加速大量基礎設施的遊戲部署。如需此部署方法的最佳實務，請參閱 上的藍/綠部署 AWS。

Canary 部署

Canary 部署對於遊戲開發人員很有用，因為策略可以套用來發行遊戲的早期 Alpha 或 Beta 建置，或遊戲功能，例如新遊戲模式、地圖或挑戰生產中受限或少數的玩家。這種部署稱為 Canary。此版本可能有額外的追蹤和報告，因此當真正的玩家玩該遊戲或功能時，會收集和分析其遊戲遙測是否有異常和問題。

對於新功能，不會一致地通知玩家，而遊戲遙測是用來判斷玩家是否遇到問題的主要來源，並且應該復原發行版本。同時，如果未發現重大問題，則可以進一步將此功能推出給更多玩家以取得其他資料。如果玩家收到通知，則可以要求他們定期提供有關其體驗的意見回饋。這類測試活動最好是由即時操作團隊協調。

作為策略，Canary 部署也可以用於標準版本，逐步為玩家提供新功能。標準藍/綠環境的潛在優勢是不需要完整規模的第二個環境。新的縮減規模環境的容量決定要加入多少玩家加入新功能。在新增更多玩家之前，必須適當擴展容量。即使此自訂藍/綠技術預期成本低於標準藍/綠，但仍估計會產生的成本可能高於金絲雀部署的滾動替代技術。

只對生產環境執行單一 Canary，並針對其資料和意見回饋聚焦。如果部署了多個 Canary，它會使生產環境中問題的疑難排解和隔離更為複雜，並損害所收集資料集和意見回饋的品質。

Canary 的變化是當一或多個實驗（通常是 UI 測試）透過目標部署執行時，其中一組遊戲後端伺服器提供一個版本的功能，另一組相同大小的套件提供另一個版本的相同功能。系統不會為此建立其他或特殊基礎設施，而且只有所選的後端伺服器插槽會收到這些更新。實驗的結果是觀察玩家如何對相同特徵的每個版本做出反應，判斷整體上是否贊同，並觀察其可用性或功能是否發現問題。這種策略實驗也稱為 A/B 測試，而整體程序稱為 A/B 測試。完成這些實驗後，在還原至用於測試的伺服器上遊戲後端系統的目前版本之前，會收集必要的測試資料。

傳統傳統部署

在傳統的部署風格中，在排程維護時段期間，遊戲會關閉，並在遊戲後端內的伺服器執行個體更新為最新的程式碼建置之前，先捨棄或耗盡連線的玩家。此部署每次執行時都會影響玩家，而且必須提前通知玩家。因此，此模型對玩家的影響最大，應盡可能避免。

部署遊戲更新之後，遊戲可以在向玩家開放遊戲之前進行煙霧測試，玩家會等待遊戲重新開啟。當玩家嘗試在短時間內登入和玩遊戲時，這可能會導致流量激增。因此，如果遊戲的設計無法處理這類流量峰值，您可以選擇逐步允許玩家分批返回遊戲。

或者，您可以選擇過度佈建基礎設施，以維持流量的高峰，並在遊戲流量穩定後，將資源縮減。如有必要，請在玩家數量最低的離峰時段執行此類部署。經常排程的維護以及延長的維護本質上具有玩家流失和潛在收入損失的風險。玩家也預期在新版本之後會有變更，而且在停機一段時間後返回時，可能會失去對遊戲的信任。

實作步驟

- 將停機時間降至最低：實作部署策略，以減少停機時間並讓玩家留在遊戲中。
- 基礎設施即程式碼 (IaC)：使用 AWS CloudFormation 或 Terraform 等工具來管理遊戲基礎設施並減少人為錯誤。
- 部署策略：使用滾動替換、藍/綠和金絲雀部署的一個或組合，以提供順暢的更新並減少玩家的影響。

支援尖峰需求所需的 GAMEOPS03-BP05 預先擴展基礎設施

在大規模遊戲事件之前擴展基礎設施，以確保您可以處理玩家需求的突然增加。

未建立此最佳實務時的曝險等級：高

實作指引

除了推出新的遊戲之外，即時遊戲通常會執行遊戲內活動、促銷、新內容和賽季版本，作為維持和改善玩家參與度的範例。在事件或提升期間，這類活動會遇到大量玩家流量。企業預期會達到或超過活動預期的目標，而且遊戲基礎設施必須透過該目標來維持和支援這些目標。

事先準備您的基礎設施，以便能夠支援您在大規模事件期間將遇到的預期玩家負載。為了做好準備，遊戲營運團隊應與銷售和行銷領域的利益相關者協調，透過查看過去的玩家並行、參與指標和銷售資料，來估計即將在近期事件中產生的預計需求。如果事件是用於新的遊戲啟動，遊戲操作團隊應該與這些利益相關者合作，以識別他們預期規模的實際預測。雖然很難預測遊戲的成功程度，但每個人都必須了解對成功的期望，才能擴展和測試基礎設施以支援這些目標。

許多遊戲選擇分階段啟動，從向少數玩家開啟遊戲開始，然後在完全公開啟動之前，在每個階段以有機方式擴展玩家。在軟啟動期間，監控、識別、追蹤和解決問題，同時精簡公有啟動的投影。

若要正確估算基礎設施需求，請在遊戲啟動之前，透過針對在生產或類似生產的預備環境上執行的遊戲後端執行的負載和效能測試來收集資料。應執行這些測試的多個回合來模擬遊戲的不同條件，並驗證後端在大多數情況下可以承受負載。

為了達成此目的，開發人員可以撰寫遊戲機器人，在遊戲中周遊各種工作流程，並模擬不同的條件。這些測試應檢查遊戲後端的不同系統層，以便測試每個層和元件並記錄詳細資訊。使用這些測試收集的資料來佈建遊戲啟動的計劃。

應盡可能透過提高應用程式可用性和容錯能力來識別和移除單一故障點 (SPOF)。使用負載測試，透過模擬不同上游和下游層的故障，以及驗證遊戲和其他元件行為來識別 SPOFs。

除了為遊戲啟動、遊戲內事件或提升準備佈建的必要預估基礎設施之外，請設定系統以自動隨需擴展。定義、設定和監控擴展事件閾值，以允許遊戲後端擴展以維持大量玩家流量。對於可變流量，預先佈建是最好的，因為可能沒有足夠的時間橫向擴展。在初始遊戲啟動期間，可能需要手動擴展，以比自動化系統更快的速度驅動高於預期的需求，進而擴展資源。

在 上 AWS，組織應為他們在遊戲後端中使用的服務請求更高的 Service Quotas。為帳戶設定 Service Quotas，以防止客戶意外站立或擴展比預期更多的基礎設施。當帳戶中執行的遊戲達到該區域中設定之服務配額的上限時，服務會調節超出佈建配額和爆量佈建的請求。調節可能會導致意外或非預期的錯誤，並損害玩家體驗。監控、追蹤和定期檢閱遊戲生產中所使用的服務的服務配額閾值，以避免限流。當用量超過可容忍的服務配額閾值時，可以從主控台支援中心提出支援案例、登入受影響的帳戶，或使用支援 API 來請求增加配額。

對於遊戲啟動、內容發行、促銷和主要遊戲內事件等重要事件，請使用 AWS Countdown。Countdown 會根據 Games 專家建置的手冊提供實作指導，以提供操作準備、降低潛在風險，以及規劃容量需求。AWS Countdown 也有進階支援選項，可提供增強的支援和選項，例如工程師，以最佳化您的基礎設施。

如果您要啟動 Amazon GameLift 上託管的遊戲，請檢閱啟動前檢查清單以進行準備。

實作步驟

- 向前擴展基礎設施：事先為大規模遊戲事件準備基礎設施，以處理玩家需求的突然增加。
- 預估需求：與銷售和行銷部門協調，使用過去的玩家資料和逼真的預測來預估預估需求。
- 負載測試和 SPOF 移除：執行多輪負載測試，以驗證後端容量、識別單點故障，並正確設定自動擴展。

運作狀態監控

GAMEOPS04：如何監控遊戲的運作狀態？

透過實作全面的檢測來監控遊戲運作狀態，以偵測和追蹤影響玩家的問題，包括用戶端活動記錄、後端服務監控和錯誤報告。結合使用 Amazon CloudWatch AWS X-Ray和 等 AWS 工具以及第三方解決方案，以協助快速識別和解決問題。

最佳實務

- GAMESOPS04-BP01 檢測遊戲以偵測和監控影響玩家的問題

GAMESOPS04-BP01 檢測遊戲以偵測和監控影響玩家的問題

除了回應社交媒體和玩家問題報告之外，您還可以使用監控解決方案來檢測您的遊戲，以偵測和調查影響玩家的問題。

未建立此最佳實務時的曝險等級：高

實作指引

沒有任何測試量可以識別遊戲中的每個問題。遊戲通常會以已知問題啟動，這些問題預計會隨著遊戲的下一個版本逐漸修正。已知且可重現的問題很容易解決和修正。為了協助識別此類問題，遊戲用戶端應該在各種策略位置實作玩家活動追蹤、應用程式記錄和報告，以協助後端團隊識別用戶端問題。儘早發現此類問題的能力，有助於遊戲開發人員在問題變得普遍之前進行故障診斷和修正。追蹤程式碼報告的資料和日誌絕不應包含個人身分識別資訊 (PII)，而且只應包含協助偵錯的遊戲特定中繼資料。

實作可觀測性解決方案，以偵測和回應遊戲當機或錯誤等問題。您可以使用 Amazon CloudWatch Synthetics 建立 Canary，以監控面向玩家的後端遊戲服務的運作狀態。您可以使用 檢測後端服務AWS X-Ray，以追蹤分散式服務的請求，並將自訂日誌和指標傳送至 Amazon CloudWatch。

第三方解決方案，例如 Backtrace.io 和 Sentry，是遊戲中錯誤報告的熱門解決方案。來自 New Relic、Splunk、Datadog 和 Honeycomb.io 等合作夥伴的應用程式效能監控 (APM) 解決方案也很受歡迎。

除了官方支援管道之外，遊戲的即時操作團隊和社群經理還應監控各種社交網路和管道，以檢查玩家意見回饋、投訴和錯誤報告。檢閱並嘗試重現每個遊戲特定的投訴，或將其傳送給 QA 團隊進行檢閱。如果可重現，請在影響較大的玩家群之前，將問題上報給遊戲開發人員以進行疑難排解和修正。

實作步驟

- 實作監控解決方案：使用監控工具偵測影響玩家的問題並快速回應。
- 追蹤玩家活動和日誌：檢測遊戲用戶端以記錄玩家活動和報告問題，並確認不包含個人身分識別資訊 (PII)。
- 使用第三方和 AWS 工具：使用 CloudWatch、X-Ray 和第三方解決方案等工具進行錯誤報告和效能監控，並監控社交媒體以進行玩家意見回饋和錯誤報告。

負載測試

GAMEOPS05：載入測試遊戲時應考慮什麼？

載入測試遊戲時，請考慮適當的測試階段、負載產生架構和測試架構，以有效地評估系統效能和可擴展性。選擇正確的時機組合（早期開發、衝刺、生產前或部署後）、基礎設施 (EC2、EKS、Fargate 或 Lambda) 和測試工具 (JMeter、Locust、Grafana K6 或 Gatling)，以符合遊戲的獨特特性和開發目標。

最佳實務

- GAMEOPS05-BP01 選擇正確的階段、架構和負載測試架構以符合您的目標

GAMEOPS05-BP01 選擇正確的階段、架構和負載測試架構以符合您的目標

載入測試遊戲的方法可能會因許多因素而有很大差異，包括其執行的開發程序階段、負載產生系統本身的架構，以及負載測試架構的選擇。執行時機，無論是在早期階段、反覆衝刺期間、生產部署之前或部署後，都會塑造測試工作的目標和重點。不同的負載產生基礎設施設計具有自己的優缺點，而負載測試架構的選擇會大幅影響測試程序可用的功能、易用性和整合。透過仔細調整這些元素，開發團隊可以根據遊戲的獨特特性量身打造負載測試方法、擷取最有價值的效能洞見，並為玩家提供流暢的體驗。

未建立此最佳實務時的曝險等級：高

實作指引

不同開發階段的負載測試

在開發階段的早期進行探索性負載測試，可以驗證基礎系統架構。這可協助開發人員在大量實作工作完成之前，做出有關遊戲基礎設施、資料庫設計和網路拓撲的明智決策。負載測試可識別風險並建立效能基準，從而可能最大限度地減少在開發生命週期稍後需要昂貴的返工和技術負債。他們也可以促進團隊之間對遊戲效能需求的共同理解，從而實現更好的協作和決策。最終，初始階段的負載測試為高效能、可擴展性和彈性遊戲奠定了堅實的基礎，有助於增強整體玩家體驗。

在每次衝刺或反覆運算結束時，負載測試可以評估最新週期中引入的新功能、錯誤修正和其他變更的效能影響。這種有針對性的方法可讓開發團隊快速識別最新更新引進的迴歸或效能降低，使其能夠在進一步向下傳播管道之前解決這些問題，並維持一致的品質和效能水準。

在部署到生產環境之前，強大的負載測試可協助團隊驗證系統處理預期實際流量和負載條件的能力。他們可以發現生產基礎設施中的可擴展性瓶頸或資源限制，並提供最佳化遊戲效能的機會，從第一天開始就建立順暢且回應靈敏的使用者體驗。從啟動前負載測試獲得的洞察可以緩解啟動日風險並通知持續的容量規劃，這為遊戲的長期永續性和可擴展性奠定了基礎。

負載測試已在生產環境中的遊戲，可讓團隊監控遊戲效能，並識別一段時間內可能發生的效能迴歸或降級。這使他們能夠在問題影響玩家體驗之前主動解決問題，並對使用者保留率產生負面影響。此外，生產中的負載測試會驗證已實作的效能最佳化工作或基礎設施擴展的有效性。即使遊戲不斷發展和成熟，此程序也為玩家提供高品質、回應能力高且可擴展的遊戲體驗。

產生負載的架構

遊戲負載測試的負載產生架構設計可以採用各種形式，每個形式都有自己的一組優點和考量。

在最基本的層級，可以佈建自我管理的 Amazon EC2 執行個體，並設定為充當負載產生器。透過控制節點和工作者節點方法，您可以設定多個負載產生執行個體，每個執行個體都會執行自己的測試指令碼，並由單一控制執行個體整體管理。架構可以擴展並產生更多負載，而不會增加額外的工作者節點的複雜性，但這種實作方法需要團隊處理基礎基礎設施的佈建、設定和管理。

如需更具可擴展性和協調性的方法，您可以使用 Amazon EKS Kubernetes 叢集來管理和分配容器型負載代理程式機群的負載測試工作負載。Kubernetes 自動擴展功能可用來處理負載產生型 Pod 的擴展，而團隊本身則可在託管 Pod 的叢集中設定和管理基礎 EC2 執行個體。

或者， 的無伺服器特性AWS Fargate可以透過抽象化基礎設施管理，同時仍提供必要的可擴展性和靈活性，來加速和簡化負載測試設定。對於現場部署、產生負載的 Kubernetes 叢集已存在，但可能需要額外容量的混合解決方案，EKS Anywhere 可以從 將兩個叢集管理為一個 AWS 管理主控台叢集。

您也可以根據您的需求和目標使用 AWS Lambda 函數。Lambda 函數相對簡單，無需佈建和管理其他資源即可設定和擴展。由於與其他 AWS 服務的深度整合，它們也允許建立更複雜和動態的測試案例。不過，Lambda 函數對並行函數和執行時間 (15 分鐘） 有限制，這可能會限制可達到的負載測試規模和長度。冷啟動延遲也會影響結果的準確性，Lambda 的資源限制可能不適合高要求負載測試工作負載。

想要使用預先建置解決方案的 Studio 可以使用 上的分散式負載測試 AWS。此解決方案在 上使用 Amazon ECS AWS Fargate 來部署容器，以執行數萬個連線使用者的模擬。您可以使用此功能，以 IAC 方式使用 快速啟動負載測試基礎設施 AWS CloudFormation。

負載測試架構

沒有兩個負載測試架構的建置方式相同。有些具有直覺式圖形界面來建立測試，有些則是完全以命令列為基礎。一個工具可能具有彈性和效能，但需要時間和精力來設定和管理，另一個工具可能沒有伺服器，但在可以建立和執行的測試中受到限制。一些人喜歡大型社群和大量的教學課程，同時在現場未經驗證，與可能在生產環境中經過戰鬥測試但缺乏社群支援或文件的社群形成強烈對比。選擇為您和團隊取得適當平衡的架構。一些熱門選項包括：

- Apache JMeter：基於 Java 的熱門開放原始碼負載測試架構，因為其強大的功能集和易用性。它能夠模擬複雜的使用者案例、各種支援的通訊協定、全面的報告和經過驗證的追蹤記錄，使得 JMeter 成為負載測試的可靠選擇。
- Locust：以事件驅動型架構建置的現代分散式負載測試架構，既具效能又具資源效率。 測試是以 Python 撰寫，可靈活地測試案例，利用數千個強大的第三方程式庫，同時保持友善且易於閱讀。

- Grafana K6：強大的負載測試架構，結合了易用性與進階功能。其支援分散式負載產生、彈性指令碼，以及與 Grafana 無縫整合以進行資料視覺化，讓 Grafana K6 成為吸引人的選擇。
- 遊戲：開放原始碼負載測試架構以其效能和可擴展性著稱。其以 Scala 為基礎的特定網域語言 (DSL) 可讓開發人員建立簡潔、可維護的負載測試指令碼，其強大的報告和分析功能可提供受測系統的詳細洞見。

實作步驟

- 負載測試階段：在各種開發階段（早期開發、衝刺、生產前和部署後）執行負載測試，以驗證系統效能並識別問題。
- 負載產生架構：根據可擴展性需求、管理偏好設定和特定測試需求，選擇適當的負載產生架構 (EC2、EKS、Fargate 或 Lambda)。
- 負載測試架構：選取負載測試架構（例如 JMeter、Locust、Grafana K6 或 Gatling)，在易用性、效能、彈性和社群支援之間取得平衡，以符合團隊的需求。

隨時間優化

GAMEOPS06：如何隨著時間最佳化遊戲？

透過監控關鍵指標和遙測資料來識別玩家趨勢、系統效能和需要改進的領域，隨著時間的推移最佳化您的遊戲。根據這些洞察持續更新遊戲設計、基礎設施和負載測試方法，同時適應新技術和架構，以隨著遊戲發展提供最佳效能和玩家體驗。

最佳實務

- GAMEOPS06-BP01 監控關鍵遊戲指標以識別玩家趨勢和模式，並使用資訊改善遊戲
- GAMEOPS06-BP02 在遊戲變更時更新和調整負載測試方法

GAMEOPS06-BP01 監控關鍵遊戲指標以識別玩家趨勢和模式，並使用資訊改善遊戲

除了遊戲用戶端系統用量、應用程式用量、例外狀況和當機資料之外，還會擷取傳送至遊戲後端系統的遊戲遙測資料。此資料應代表玩家活動，以便您可以了解玩家如何與遊戲中的各種功能互動。

未建立此最佳實務時的曝險等級：高

實作指引

根據其實作，遊戲用戶端可以在遊戲世界中預先定義的遊戲功能或位置收集遙測資料。資料會傳送至後端擷取服務進行處理。如果無法連線後端服務，用戶端可以在本機裝置上本機存放資料，直到後端服務再次可用為止。遊戲設計人員使用此遙測資料來檢閱玩家如何玩遊戲，以及遊戲中是否有異常。

例如，玩家移動和與地圖中項目的互動可以從遙測資料中擷取，並繪製為玩家在遊戲中活動在設定的時段內的熱度圖。這類資料可協助遊戲設計人員識別平衡遊戲中各種元素的需求，例如武器的力量、遊戲內角色的力量或地圖的複雜性。原始遙測資料通常會儲存然後處理，以擷取分析師可視覺化的分析。

Game AnalyticsPipeline 解決方案實作可協助遊戲開發人員啟動可擴展的無伺服器資料管道，以擷取、儲存和分析從遊戲和服務產生的遙測資料。解決方案支援串流資料擷取，讓使用者在幾分鐘內就能從遊戲和其他應用程式取得洞見。

對於自訂遊戲遙測資料擷取、儲存、處理和分析， AWS 也為大數據處理和分析提供許多專業服務。

實作步驟

- 擷取遊戲遙測資料：收集玩家活動、系統使用量、例外狀況和當機的資料，以了解玩家互動並識別問題。
- 實作遙測收集：使用預先定義的遊戲功能或位置來收集遙測資料，並將其傳送至後端服務，如果後端無法連線，則儲存在本機。
- 使用 AWS 分析解決方案：使用 Game Analytics 管道等 AWS 服務進行可擴展的資料擷取、儲存和分析，以及專門的大數據處理和分析服務。

GAMEOPS06-BP02 在遊戲變更時更新和調整負載測試方法

最佳化負載測試方法是一個持續的過程，應隨著遊戲開發週期而發展。隨著遊戲的複雜性、使用者基礎和功能集的增長，負載測試策略必須進行調整，以確認其準確模擬真實世界條件並提供可行的洞見。

未建立此最佳實務時的曝險等級：高

實作指引

考慮下列各項：

測試案例遺失或過時

在開發過程中將新功能新增至遊戲時，請建立並執行新的負載測試案例，以驗證新功能的效能和可擴展性。同樣地，特徵和功能通常經過重構，以改善效能、解決玩家意見回饋，或符合新的設計目標，要求持續更新測試案例，以跟上變更的步伐，並真正測試和反映系統的狀態。

新的負載測試架構

開發人員可能會因為各種原因而需要變更負載測試架構：

- 初始架構可能不再能夠充分模擬使用者負載，或提供有關系統效能的必要洞見
- 新的遊戲功能可能需要支援新通訊協定、APIs負載測試
- 開發人員可能會想要更進階的功能，因為他們更熟悉負載測試程序
- 偏好更符合團隊技術專業知識、程式設計語言或現有工具鏈的架構

透過仔細評估和調整一段時間，開發人員可以將負載測試程序與遊戲不斷變化的需求保持一致，並繼續提供必要的洞見，以最佳化和改善整體使用者體驗。

最佳化成本

使用受管 AWS 服務的簡單性和便利性可能非常有益，尤其是在開發的早期階段。這些服務抽象基礎基礎設施管理，可讓團隊快速設定其解決方案，並僅專注於製作負載測試案例和分析結果。不過，由於受管服務提供的額外價值和便利性，例如佈建、設定和維護基礎設施，以及提供高可用性、擴展和監控功能，因此使用受管服務通常成本更高。

隨著團隊對其負載測試程序的成熟和成長更加舒適和自信，有時自我管理基礎設施可以提供額外的最佳化和節省成本。雖然這種實作方法會增加營運開銷，但直接控制運算資源、組態、擴展行為和資源使用率，可以釋放微調和降低成本的新機會。例如，團隊使用無 AWS Fargate 伺服器架構開始負載測試旅程可能很合理，之後再移至自我管理 Amazon EKS 叢集中的基礎節點。

實作步驟

- 更新測試案例：持續建立和更新負載測試案例，以驗證新功能和重構功能，並確認它們反映遊戲的目前狀態。
- 評估負載測試架構：視需要適應新的架構，以模擬使用者負載、支援新的通訊協定，並符合團隊的專業知識和工具鏈。
- 最佳化成本：從輕鬆方便的受管 AWS 服務開始，然後考慮自我管理基礎設施，以節省成本，因為團隊對負載測試程序越來越熟悉。

Resources

請參閱下列資源，進一步了解我們卓越營運相關的最佳實務。

文件和部落格

- Game Tech 的架構最佳實務
- 在 AWS pt.1 上管理您的遊戲 Studio
- 在 AWS pt 上管理您的遊戲 Studio。2
- 在 AWS pt 上管理您的遊戲 Studio。3
- 建立您的最佳實務 AWS 環境
- Control Tower 登陸區域的多帳戶策略
- 遊戲分析管道
- 使用遊戲分析管道將您的遊戲資料洞見最大化
- 利用 AWS Glue 和 Amazon Redshift Spectrum for Player Insights
- 如何在 上設定 CI/CD 管道
- 良好的任務遊戲如何使用 AWS 建置管道加速 43%
- 實作 Unity 行動應用程式的建置管道
- 其他相關的 CI/CD 部落格
- 遊戲伺服器 CD 管道部落格讓遊戲 DevOps 變得簡單
- Harmony Games 部署完全自訂的遊戲後端使用率 AWS Cloud Development Kit (AWS CDK) (AWS CDK)
- GameLift 準備啟動
- 使用 Amazon Gamelift Anywhere 託管混合遊戲伺服器
- 使用 Amazon Gamelift Anywhere 和 Amazon Gamelift Agent 加速遊戲伺服器開發
- 如何使用 Amazon Gamelift 以低於 1 USD 的價格託管 Unreal Engine 遊戲
- 在 上建置可擴展跨平台遊戲後端的新解決方案指引 AWS
- 在 上將 Pragma 後端遊戲引擎測試到 100 萬並行使用者的負載 AWS
- Code Wizards 如何透過 將 Heroic Lab 的 Nakama 載入兩百萬名並行玩家 AWS
- 組織單位的最佳實務
- AWS X-Ray
- AWS 倒數計時

- AWS for Games Solutions Hub

合作夥伴解決方案

- New Relic
- Splunk APM
- Backtrace.io
- Sentry
- Datadog APM
- Honeycomb.io

白皮書

- 使用多個帳戶組織您的環境
- 上的可擴展遊戲開發模式簡介 AWS

影片

- YouTube 系列：在 上建置遊戲 AWS
- AWS 適用於遊戲：Boss LEVEL Podcast
- Re：Invent 2023： AWS 為前 1，000 萬使用者擴展
- Re：Invent 2022：Riot Games 如何在 上每天處理 20TB 的分析 AWS
- Re：Invent 2022：Riot Games AWS 如何建置控管報告引擎
- Re：Invent 2023：在 上實作分散式設計模式 AWS
- Re：Invent 2023：使用 Mortal Kombat 1 將多玩家遊戲擴展到數百萬人
- Re：Invent 2022：Netflix 的混沌工程發展
- Re：Invent 2023：雲端控管的最佳實務
- Re：Invent 2023：在 上建立多區域架構的最佳實務 AWS

訓練資料

- 課程 – AWS 遊戲第 1 部分入門

安全

安全支柱包括保護資訊、系統和資產的能力，同時透過風險評估和緩解措施提供商業價值。由於全球可見性和大量玩家，遊戲是探索利用和濫用系統之方式的入侵者、駭客和其他人士的理想目標。如果沒有設定強大的安全基礎，這通常會導致令人沮喪的玩家體驗和遊戲開發人員的成本增加。

如共同責任模型中所述，了解哪些安全層面是客戶的責任 AWS ，以及哪些層面是客戶的責任很重要，以便您準備好維持強大的安全狀態。此支柱提供最佳實務雲端安全指導，供您在雲端開發和操作遊戲時考慮。

在架構系統之前，您必須建立一組包含存取控制的安全最佳實務。此外，您應該能夠識別安全事件並保護您的系統和服務，同時透過資料保護來維護資料的機密性和完整性。您應當具備界定完善且經過演練的程序，以因應安全事故。這些工具和技術很重要，因為它們支援業務目標，例如防止財務損失或遵守法規義務。

客戶範例

AnyCompany Games 是虛擬遊戲工作室，正在改善其安全狀態。安全可以直接了解何時說明其直接應用程式。本節使用 AnyCompany Games 來闡聯化 支柱中所述的安全最佳實務

重點區域

- 設計原則
- 安全基礎
- 持續的安全性
- 身分與存取管理
- 存取控制
- 偵測
- 基礎設施保護
- 事件回應
- 應用程式安全
- 自動化安全性
- 威脅建模
- Resources

設計原則

除了 Well-Architected Framework 白皮書安全支柱的設計原則之外，下列設計原則還可以增強雲端中遊戲工作負載的安全性：

- 監控和中等玩家使用行為：擷取和分析使用資料，以了解玩家如何與您的遊戲和社交功能互動。透過分析此資料，您可以偵測和回應可能降低玩家體驗的濫用和不適當的行為。

安全基礎

GAMESEC01：如何實作遊戲開發的安全基礎知識？

遊戲工作室需要獨特的安全方法，以保護開發環境和即時玩家服務。遊戲工作室的強大 AWS 安全策略需要三個互連元件：多帳戶結構、強式身分驗證，以及使用 IAM 政策的明確授權策略。多帳戶 AWS 結構可讓 Studio 分隔不同的遊戲專案、開發階段和工具環境。這可讓工作室更精細地控制某些事物，例如存取特定環境或服務。啟用強大的身分驗證可驗證團隊成員可以安全地存取開發資源，無論是在現場或遠端工作，同時對原始程式碼、遊戲組建和專屬工具維持嚴格的控制。Studio 也應該有明確的授權策略，以使用具有 IAM 許可和角色的最低權限原則授予許可。使用 IAM 角色在不同的開發團隊角色之間指派許可，例如讓開發團隊存取低階 AWS 服務，同時限制藝術家和設計師使用特定的資產管理和建置系統。這種專門的方法可驗證遊戲工作室可以保護其智慧財產權、維持高效的開發工作流程，並安全地擴展其團隊，同時讓開發人員能夠適當存取以快速迭代其專案。

最佳實務

- GAMESEC01-BP01 使用角色和聯合存取，而不是帳戶根使用者，在您的 AWS 環境中執行動作
- GAMESEC01-BP02 使用 AWS Control Tower 在 上快速設定多帳戶環境 AWS
- GAMESEC01-BP03 使用針對特定任務職能量身打造的最低權限角色政策
- GAMESEC01-BP04 將角色和聯合存取政策與帳戶層級存取政策搭配使用，以授予對 AWS 資源的存取權
- GAMESEC01-BP05 使用中央身分提供者

GAMESEC01-BP01 使用角色和聯合存取，而不是帳戶根使用者，在您的 AWS 環境中執行動作

第一次建立 時 AWS 帳戶，您會從稱為根使用者的身分開始，該身分是使用與帳戶相關聯的電子郵件地址和密碼存取。根使用者可完整存取該帳戶中 AWS 的服務和資源。在大多數情況下，您應該避免將根使用者用於day-to-day任務。需要根層級存取時，請確認這是絕對必要的，並確認有額外的記錄和護欄來追蹤其使用情況。

未建立此最佳實務時的曝險等級：高

實作指引

在 AWS Organizations 組態中，每個帳戶仍然有自己的根使用者，但day-to-day存取。建立針對遊戲生命週期階段和團隊量身打造的角色型存取。例如，即時操作團隊可能需要管理遊戲內事件的許可，而開發人員則需要推送更新的存取權。與第三方服務或合作夥伴合作時，請使用聯合存取來啟用安全協作，而不會暴露敏感的基礎設施。此方法會驗證每個使用者或合作夥伴是否只有他們所需的存取權，同時維護遊戲基礎設施和玩家資料的安全性。

客戶範例

AnyCompany Games 在開發新遊戲時實作以角色為基礎的存取控制。透過為不同的開發團隊使用特定的 IAM 角色，他們可避免使用共用的登入資料。此設定可讓開發團隊擔任核心遊戲系統的角色，而內容團隊的角色只能存取資產管理服務。

實作步驟

- 除非絕對必要，否則在設定帳戶後請勿使用根使用者。建立帳戶、保護根使用者，並立即建立所需的管理 IAM 角色，並將該角色指派給聯合身分使用者。
- 只有在您需要執行僅限根使用者可用的有限數量任務時，才使用根使用者。這些任務的範例包括變更您的根使用者電子郵件地址，以及變更您的 AWS 支援計劃。

GAMESEC01-BP02 使用 AWS Control Tower 在 上快速設定多帳戶環境 AWS

如果您只 AWS 以單一帳戶開始使用 ，您可能會發現遊戲工作室隨著遊戲開發程序的進展而成長。例如，使用單一 AWS 帳戶，您可能會開始達到服務限制，或者不同專案和工作負載的成本可能會變得更加複雜。為不同的遊戲標題和環境建立不同的帳戶，可讓團隊試驗新功能、繞過服務限制，並維護安全

狀態和合規性。透過在 中實作多帳戶策略 AWS，您可以受益於將服務限制分散到多個帳戶，並深入了解您的 AWS 成本。

未建立此最佳實務時的曝險等級：中

實作指引

使用多個 AWS 帳戶 會自動更令人困惑且耗時，這是常見的錯誤概念。相反地，使用旨在促進多個帳戶控管 AWS 的服務，可協助遊戲工作室減少管理帳戶的時間。

您可以使用 AWS Control Tower 服務來安全地佈建多帳戶 AWS 環境。如果您正在建置新的 AWS 環境、開始旅程 AWS或完全不熟悉，建議使用 Control Tower AWS。在簡短設定程序 期間，您可以與涉及管理帳戶和使用者存取權的其他 AWS 服務整合 AWS Organizations，例如 Service Catalog 和 AWS IAM Identity Center。

客戶範例

AnyCompany Games 最初從單一 操作 AWS 帳戶，當其中一個遊戲的開發團隊在重要的 Beta 測試期間達到 EC2 服務限制時，就會遇到多個障礙。同時，他們的不同遊戲開發團隊在自動化測試管道的資源配置方面遇到困難。當 AnyCompany Games 無法準確區隔專案之間的成本，使每個遊戲的開發難以編列預算時，情況就達到了重大點。

AnyCompany Games 接著使用 實作多帳戶策略 AWS Control Tower。他們為每個遊戲專案建立單獨的帳戶，具有不同的開發、QA 和生產環境。此帳戶層級分離會隔離每個專案的資料和資產，因此處理一個遊戲的團隊無法存取或修改另一個遊戲的資源。透過 AWS Organizations，他們建立了集中式帳單結構，可清楚顯示每個遊戲的基礎設施成本，並建立整個組織的存取政策。

實作步驟

- 使用 AWS 控制塔來設定自動化多帳戶環境。
- 根據環境（例如開發、QA 和生產）組織帳戶。
- 使用 AWS IAM Identity Center 和服務目錄來集中使用者許可，並簡化跨帳戶的資源佈建。

GAMESEC01-BP03 使用針對特定任務職能量身打造的最低權限角色政策

設定 IAM 政策是建立強大安全基礎的重要部分。設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。例如，QA 團隊需要存取權才能變更測試環境中的物件，但不應能夠修改生產環境。

未建立此最佳實務時的曝險等級：中

實作指引

您可以從廣泛的許可開始，例如 受管政策，同時探索工作負載或使用案例所需的許可。隨著使用案例的成熟，您可以設法減少授予的許可，以便朝向最低權限的目標邁進。

實作步驟

- 遵循為使用者和應用程式建立 IAM 角色的最低權限許可實務。
- 使用 AWS受管政策快速提供廣泛的存取，同時識別需要執行其任務的特定許可團隊或應用程式。
- Studio 也可以使用 IAM 存取分析器政策產生，根據 CloudTrail 事件產生自訂 IAM 政策，以識別 IAM 項目使用的動作和服務。
- 定期檢閱 IAM 政策並編輯過度寬鬆的政策。

GAMESEC01-BP04 將角色和聯合存取政策與帳戶層級存取政策搭配使用，以授予對 AWS 資源的存取權

新 AWS 使用者通常只會在將存取權授予其他人時使用 IAM 政策。不過，如果您使用的是 AWS Organizations，請考慮如何使用服務控制政策搭配 IAM 政策，授予您的 Studio 團隊成員和承包商必要的存取層級。

未建立此最佳實務時的曝險等級：中

實作指引

您可以建立 IAM 政策，以允許或拒絕存取 AWS 使用的服務或 API 動作 AWS Identity and Access Management。它們只能套用至 IAM 身分，例如使用者、群組或角色。例如，IAM 政策可用於提供使用者對 Amazon S3 的唯讀存取權。

服務控制政策 (SCPs) 是 的護欄 AWS 帳戶。SCP 不會授予許可，而是用來限制個別成員帳戶 AWS 的服務動作。例如，SCP 可以拒絕 AWS 帳戶 存取特定區域。

採取動作時，會結合 SCPs 評估相關的 IAM 政策。在上述範例中， 角色是嘗試執行 EC2 執行個體，IAM 表示是否允許 (ec2：RunInstances 為「允許」)，而 SCPs 會判斷其區域選擇是否有效 （ 允許「us-east-1」，但 SCP 拒絕「us-west-1」)。

分層 IAM 政策和 SCPs 可以驗證存取您 AWS 資源的任何人只會獲得他們所需的適當許可。這對於考慮您的 AWS 帳戶 和資源是否跨越多個區域特別重要，但並非遊戲工作室中的每個人都需要存取所有區域。

您可以量身打造 IAM 政策，授予特定團隊特定的許可，以更新遊戲組態、管理玩家資料、設定促銷活動，以及調節使用者產生的內容。同時，您可以使用 SCPs強制執行對遊戲操作至關重要的全組織控制。這些可能包括將部署限制為僅遊戲操作的核准區域、協助防止未經授權存取敏感玩家資料存放區、強制執行合規要求，以及透過限制跨開發帳戶的服務用量來控制成本。

實作步驟

- 使用 IAM 政策來管理個別使用者、群組或角色的許可。
- 在 中使用服務控制政策 SCPs) AWS Organizations 來強制執行帳戶層級許可。
- 結合 IAM 政策和 SCPs，僅授予特定使用者和帳戶所需的存取權。

Resources

- IAM AWS 中的政策和許可
- 服務控制政策
- 工作職能的AWS 受控政策

GAMESEC01-BP05 使用中央身分提供者

中央身分提供者可做為儲存和管理使用者登入資料、身分、許可和身分驗證的單一來源。

未建立此最佳實務時的曝險等級：中

實作指引

使用中央身分提供者簡化您的使用者身分驗證程序、強制執行一致的安全政策，並簡化您 AWS 帳戶和 應用程式的使用者管理。採用集中式方法可免除個別管理使用者身分和登入資料的需求，進而降低不一致、備援和其他安全漏洞的風險。將使用者身分和身分驗證合併到一個位置也可讓您的整個 AWS 環境有更好的可見性、控制性和可稽核性。

客戶範例

AnyCompany Games 在其快速擴展 AWS 的基礎設施中，在管理開發人員存取權方面面臨重大挑戰。他們的開發團隊在三個主要標題中從 50 人成長到 200 人。一開始，每個專案團隊都會管理自己的 AWS 存取憑證，導致不一致的安全實務、新開發人員的加入延遲，以及偶爾的安全事件。

Studio 實作 AWS IAM Identity Center 作為其中央身分提供者，將使用者管理合併為單一系統。他們將其與其現有的公司目錄連線，讓開發人員能夠使用相同的公司登入資料進行 AWS 存取。現在，開發人員使用其單一的現有公司登入來取得完成其工作所需的 AWS 存取權

實作步驟

- 請考慮使用 AWS IAM Identity Center 做為您的中央身分提供者。這可在您的 之間提供一致的存取管理 AWS 帳戶、為您的員工提供單一登入身分驗證，並簡化使用者對 AWS 應用程式的存取稽核。IAM Identity Center 也會從支援的身分提供者連線到現有的公司身分。

持續的安全性

GAMESEC02：如何實現、維護和監控持續的安全最佳實務？

對於跨產業的企業而言，遵守最佳安全實務至關重要，尤其是在遊戲產業。遊戲產業倚賴培養和維持玩家信任並建立強大的評價，甚至輕微的安全問題都可能很快破壞該信任。

此外，遊戲產業的全球性質需要符合各種產業法規和標準，這些法規和標準適用於提供遊戲的區域中的資料保護、消費者隱私權和安全性。公平且安全的遊戲是強調強大安全措施重要性的另一個關鍵層面。作弊、駭客攻擊和其他形式的遊戲利用可能會中斷合法玩家的遊戲體驗，這使得強大的安全控制對於維持遊戲體驗的完整性至關重要，並為參與者培養關卡遊戲場。

最佳實務

- GAMESEC02-BP01 使用準備好部署範本以進行標準安全實務
- GAMESEC02-BP02 發生安全事件時使用自動修補技術

GAMESEC02-BP01 使用準備好部署範本以進行標準安全實務

Ready-to-deploy的範本提供主動且敏捷的方式，以評估您在雲端的安全狀態。預先設定的範本會評估您的雲端安全性，並立即實作必要的變更。範本包含各種技術和廣泛接受的安全架構的各種最佳實務。使用 範本可協助遊戲工作室維持一致的基礎設施組態，尤其是因為它們可以擴展和新增額外的 AWS 帳戶 以支援新的工作負載。

未建立此最佳實務時的曝險等級：中

實作指引

透過使用 AWS 服務和實作ready-to-deploy範本，遊戲開發人員可以主動評估和強化雲端安全狀態、保護其智慧財產權、保護玩家資料，並透過定期安全評估和持續監控來培養安全的遊戲環境，以快速識別和解決潛在的漏洞。

客戶範例

AnyCompany Games 在準備在歐洲產業推出下一個標題時面臨重大挑戰。他們意識到其現有的資料處理實務不符合 GDPR 要求。他們轉向 AWS Security Hub CSPM AWS Config 和 及其ready-to-deploy 的解決方案範本。團隊在其中實作了 GDPR 特定的一致性套件 AWS Config，該套件會根據 GDPR 標準自動評估其現有基礎設施。此初始掃描顯示數個重大差距，例如不當的資料保留政策，以及玩家資料存放位置的存取控制不足。AnyCompany Games 會使用範本的預先定義規則，快速實作必要的變更。此外，範本提供的持續自動化合規檢查可讓小型團隊輕鬆地維持 GDPR 合規，即使他們繼續更新和擴展遊戲也一樣。

實作步驟

- 將 範本用於標準安全實務，例如 中的受管規則和一致性套件 AWS Config ，以及 中的標準 AWS Security Hub CSPM。
- 檢閱 Security Hub CSPM 標準的詳細資訊，以判斷哪些符合遊戲工作室的安全需求。

GAMESEC02-BP02 發生安全事件時使用自動修補技術

遊戲開發人員可以使用自動化修補技術，主動保護和維護其遊戲基礎設施，並將安全事件可能帶來的潛在影響降至最低。如果偵測到安全問題，請使用 Runbook 引導您對情況的回應。盡可能將這些回應自動化，以更快地修復問題並減少其影響。這可透過減少遊戲停機和中斷的機會來改善玩家體驗。

未建立此最佳實務時的曝險等級：中

實作指引

準備回應安全問題不僅可以保護玩家的體驗，還可以滿足各種合規和法規標準。此外，使用自動化安全回應可在工作負載擴展時擴展您的安全操作。 AWS 提供服務，以協助識別和自動化對這些事件的回應。

客戶範例

AnyCompany Games 在例行資產管道更新期間意外公開 S3 儲存貯體，其中包含即將進行之遊戲的未發行角色模型和紋理時，面臨重大安全事件。自動化安全系統在修改後的幾分鐘內偵測到儲存貯體許可變更。系統會立即執行其修補 Runbook：將儲存貯體還原為私有狀態、記錄公開時段期間的存取嘗試、通知安全團隊，以及建立許可變更的詳細 CloudTrail 日誌。

實作步驟

- 使用 解決方案上的自動化安全回應 AWS來實作自動化 Runbook，以定義將在其中自動採取以回應安全事件的動作 AWS Security Hub CSPM。

Resources

- AWS for Games 部落格 — 在 上管理您的遊戲 Studio AWS：第 1 部分
- AWS for Games 部落格 — 在第 2 AWS 部分管理您的遊戲 Studio
- 向 註冊現有的組織單位 AWS Control Tower
- AWS 帳戶 根使用者
- 需要根使用者憑證的任務
- 上的自動化安全回應 AWS

身分與存取管理

GAMESEC03：如何管理玩家身分和存取管理？

開發遊戲時，您必須判斷如何讓玩家存取遊戲和相關服務。下一節探討設計考量事項，包括玩家身分驗證、授權和多重要素驗證。

最佳實務

- GAMESEC03-BP01 決定識別和控制玩家存取遊戲環境和資源的方法
- GAMESEC03-BP02 驗證傳送到遊戲後端服務的請求
- GAMESEC03-BP03 使用您的遊戲後端服務來驗證玩家加入多玩家遊戲的請求
- GAMESEC03-BP04 要求強式密碼，為玩家使用者帳戶強制執行嚴格的安全政策
- GAMESEC03-BP05 為玩家提供在其帳戶中設定多重要素驗證 (MFA) 的選項

GAMESEC03-BP01 決定識別和控制玩家存取遊戲環境和資源的方法

此決策會受到玩家取得和獲利策略、玩家體驗，以及其他因素的影響，例如遊戲發佈合作夥伴可能提供的現有功能。例如，遊戲可能需要購買，並要求玩家建立使用者設定檔，將實際的付款方式與其帳戶建立關聯。

未建立此最佳實務時的曝險等級：高

實作指引

或者，遊戲可能想要減少第一次玩家體驗進入的障礙，方法是消除在玩遊戲之前建立使用者帳戶的需求，從而提高玩家第一次嘗試遊戲的機會。一般而言，遊戲會針對其遊戲實作玩家身分和存取管理方法的一或多個組合。

未驗證或匿名存取

此存取層級在遊戲不需要玩家在社交網路和遊戲系統上建立新的使用者帳戶或與其身分連結的情況下非常有用。這是玩家開始玩遊戲最簡單且最快速的方式，在遊戲開發人員可能想要減少進入障礙以進行初始體驗的行動遊戲中特別有用。

在此存取案例中，如果您想要從遊戲安裝中識別用量，您可以編寫遊戲用戶端的程式，以產生唯一識別符並將其存放到玩家的裝置上。此唯一識別符用於在其裝置上跨遊戲工作階段識別玩家，並允許分析一段時間內的用量報告。稍後，如果玩家選擇建立帳戶，您可以將其新使用者帳戶與其先前產生的唯一識別符建立關聯。這會將其新玩家身分連結至其歷史用量，其中可能包括統計資料和遊戲成就。

如果玩家最終不會建立和連結帳戶，則玩家用來與遊戲互動的裝置可以唯一識別，但不會收集和儲存玩家的可復原資訊。因此，如果玩家中斷或遺失其裝置，與裝置相關聯的先前儲存資料也會遺失，而且可能無法復原。

使用使用者名稱和密碼進行身分驗證

遊戲可能允許玩家使用存放在遊戲後端的使用者名稱和密碼來建立自己的使用者帳戶。當遊戲開發人員與已存在玩家帳戶系統且開發人員可以整合的遊戲發行者合作時，可能會發生這種情況。或者，發佈自己的遊戲的開發人員可能想要透過允許玩家建立單一使用者帳戶，以便在他們發佈的遊戲中存取，來簡化玩家體驗。

與第三方社交網路和遊戲系統進行身分驗證和帳戶連結

線上遊戲和具有社交功能的遊戲通常會提供第三方身分提供者聯合，以簡化玩家體驗。與其要求玩家建立使用者名稱和密碼組合以進行身分驗證，您可以使用聯合身分來允許玩家使用其具有社交網路和遊戲系統的第三方帳戶進行身分驗證。此登入程序可簡化玩家的登入和註冊體驗。它還提供了強制性帳戶建立的便利替代方案，以及玩家存取遊戲的無摩擦方法。

對於遊戲開發人員，聯合登入程序可以提供簡化的玩家驗證工作流程。它也可以提供更可靠的方法來管理用於個人化的玩家資料。這是因為您不需要要求玩家提供他們可能已提供給第三方身分提供者的特定資料。此外，這些系統提供與其他社交功能的整合，例如將玩家與其朋友連結的能力。

實作步驟

- 使用未經驗證或匿名的存取，透過產生唯一的裝置識別符來追蹤用量，並在稍後啟用帳戶連結，來減少第一次玩家的障礙。
- 使用現有的玩家帳戶系統或跨遊戲建立統一的體驗，實作專用使用者帳戶的使用者名稱和密碼身分驗證。
- 整合第三方身分提供者以進行聯合身分驗證，簡化登入程序並允許存取社交功能和個人化資料。

GAMESEC03-BP02 驗證傳送到遊戲後端服務的請求

驗證傳送到您遊戲後端服務的請求，可能會阻止不必要的請求成功。

未建立此最佳實務時的曝險等級：高

實作指引

您應該提供身分驗證服務讓玩家登入，當玩家成功驗證時，應傳回安全的短期字符，例如 JSON Web Token (JWT) 給遊戲用戶端。

這些字符可以包含包含玩家屬性和其他相關中繼資料的宣告聲明。此相關中繼資料可用於從遊戲用戶端傳送至遊戲後端的後續請求，以驗證請求，並在已驗證玩家的內容中授權請求。

您可以選擇設計和建置自己的玩家身分驗證系統，這需要持續改進和維護，也可以使用 Amazon Cognito 提供的可擴展且安全的使用者註冊、登入和存取控制功能。

Amazon Cognito 使用者集區包含用於身分驗證和授權的使用者目錄。使用者集區提供 APIs，您可以將其整合到您的遊戲中，以進行註冊、登入和密碼重設工作流程，而這些工作流程可以與第三方身分提供者整合。Application Load Balancer 和 Amazon API Gateway 兩者都提供與 Cognito 的整合，以整合使用者身分驗證，用於透過這些服務託管傳送至自訂遊戲後端的請求。

如果您的遊戲支援匿名存取，而且您無法對玩家進行身分驗證，您可以使用用戶端身分驗證方法，在與遊戲後端整合時提供更安全的體驗。如果您的遊戲用戶端直接使用 AWS 服務，必須使用登入資料來簽署對這些服務的請求。若要為未經驗證的使用者提供登入資料給遊戲用戶端，您可以使用 SDK AWS 從 Amazon Cognito 身分集區擷取短期登入資料，這些登入資料可用來簽署您對 AWS 服務的請求。您可以從遊戲用戶端重新整理這些登入資料。

除了直接從遊戲用戶端與 AWS SDK 整合之外，您還可以使用支援自訂授權的 Amazon API Gateway 等服務來建置自己的遊戲後端。透過設計您自己的遊戲後端服務，您可以使用自訂伺服器端邏輯來取得對請求的授權控制。

如需為使用 Amazon GameLift 託管的遊戲建置後端服務的詳細資訊，請參閱設計您的遊戲用戶端服務。

客戶範例

AnyCompany Games 採用受管身分驗證和授權方法，增強了其下一個標題的安全性。他們不使用維護自訂使用者名稱和密碼系統，而是使用 Amazon Cognito 使用者集區來處理玩家註冊和登入，以及身分集區，以支援在建立帳戶之前嘗試訓練模式的玩家匿名存取。他們也在遊戲中實作自訂授權邏輯，以辨識 Cognito 中定義的管理員角色，授予這些使用者存取遊戲內特殊管理功能的權限。

實作步驟

- 使用 Amazon Cognito 使用者集區以 JWTs 等安全字符管理身分驗證，啟用註冊、登入和密碼重設等功能。
- 從 Amazon Cognito 身分集區擷取短期憑證，供匿名使用者安全地與服務互動 AWS 。
- 使用 Amazon API Gateway 實作自訂遊戲後端，以進行自訂伺服器端身分驗證邏輯。

GAMESEC03-BP03 使用您的遊戲後端服務來驗證玩家加入多玩家遊戲的請求

一般而言，在多玩家遊戲中，玩家會直接從可用工作階段清單中選擇一個選項來加入遊戲工作階段，或者提交請求來尋找配對。後者方法會讓遊戲開發人員負責尋找合格的遊戲工作階段，並將連線資訊（通常是 IP 地址和連接埠號碼）提供給玩家的遊戲用戶端。實作可能會因您正在開發的遊戲類型而有所不同，但無論如何，對玩家加入遊戲的請求執行伺服器端驗證都是安全最佳實務。

未建立此最佳實務時的曝險等級：中

實作指引

例如，在以工作階段為基礎的多玩家遊戲中，來自玩家加入遊戲工作階段的請求，應該先透過遊戲伺服器軟體與您的遊戲後端配對服務進行驗證，再授權其與伺服器的連線。當玩家請求加入遊戲工作階段時，遊戲伺服器應檢查請求是否有唯一識別符，例如玩家工作階段 ID 和先前由遊戲後端配對服務提供給遊戲用戶端的伺服器產生票證。

啟動與遊戲伺服器的連線時，您的伺服器端軟體可以使用此資訊與配對服務驗證玩家的連線請求是否有效，並確認玩家未加入先前為其他玩家在遊戲工作階段中預留的位置。

對於託管在 Amazon GameLift 上的遊戲，請參閱遊戲用戶端/伺服器與 Amazon GameLift 伺服器的互動，以取得如何實作此類伺服器端驗證的範例。

客戶範例

在 AnyCompany Games 初始 Beta 版推出期間，他們發現玩家直接連線到遊戲伺服器，繞過配對系統，導致嚴重的競爭完整性問題。當排名較高的玩家發現他們可以與好友共用伺服器 IP 地址時，他們開始規避以技能為基礎的配對系統，導致經驗豐富的玩家加入新手配對，並為新玩家創造令人沮喪的體驗。AnyCompany Games 回應的方式是實作伺服器端驗證系統，為每個配對請求產生唯一的工作階段票證。系統需要玩家 IDs 和配對請求票證，以及對其後端配對服務的驗證連線嘗試。

實作步驟

- 使用玩家工作階段 IDs 和伺服器產生的票證等唯一識別符，驗證玩家加入請求伺服器端。
- 使用配對服務確認連線請求的有效性，以封鎖未經授權的存取。
- 在驗證程序期間，確認未經授權的玩家無法存取遊戲工作階段中的預留位置。

GAMESEC03-BP04 要求強式密碼，為玩家使用者帳戶強制執行嚴格的安全政策

如果遊戲讓玩家能夠使用密碼建立使用者帳戶，您應該要求玩家的密碼遵守強大的政策。例如，Amazon Cognito 使用者集區可讓您定義使用者帳戶的密碼需求。建立強大的密碼政策可以保護玩家的帳戶不受社交工程和暴力攻擊的影響。

未建立此最佳實務時的曝險等級：高

實作指引

客戶範例

當 AnyCompany Games 的熱門標題因為密碼政策較弱而遇到一波帳戶劫持時，就會面臨危機。使用「password123」等簡單密碼的玩家成為自動暴力破解攻擊的受害者，導致物品遺失和遊戲貨幣遭到入侵。為了解決這個問題，AnyCompany Games 修訂了其登入系統，並強制要求先前未使用的密碼，至少包含一個大寫字母、一個數字、一個特殊字元，以及長度下限為 15 個字元。

實作步驟

- 玩家帳戶需要強式密碼政策來增強安全性。
- 使用 Amazon Cognito 使用者集區來定義和強制執行密碼要求。

GAMESEC03-BP05 為玩家提供在其帳戶中設定多重要素驗證 (MFA) 的選項

玩家帳戶可能是惡意人士的資產，尤其是在支援遊戲內貨幣和購買的遊戲中。由於玩家帳戶駭客入侵和社交工程攻擊的普遍性，為玩家提供透過設定多重要素驗證 (MFA) 增強其帳戶安全性的選項。

未建立此最佳實務時的曝險等級：中

實作指引

當玩家嘗試使用 MFA 存取其帳戶時，系統會將臨時密碼傳送至其電子郵件地址、電話號碼或專門建置的多重驗證行動應用程式。若要成功驗證，玩家必須在有限的時間範圍內將程式碼輸入登入系統。

MFA 也可用來協助保護嘗試從新地理位置進行身分驗證的帳戶、由玩家支援潛在惡意活動標記的帳戶，甚至是長時間未登入遊戲的帳戶。

例如，Amazon Cognito 使用者集區可以在使用者目錄上設定多重驗證。

實作步驟

- 啟用多重要素驗證 (MFA) 以增強玩家帳戶安全性。
- 使用透過電子郵件、電話或 MFA 應用程式傳送的臨時代碼來驗證帳戶存取權。
- 將 MFA 套用至新的地理位置、標記的帳戶，或具有長時間閒置的帳戶。

存取控制

GAMESEC04：如何封鎖對遊戲內容的未經授權存取？

現代遊戲包含大量內容，例如可下載的內容 (DLC)，這是玩家參與和遊戲獲利的重要層面。玩家期望持續串流新角色、關卡和挑戰，要求遊戲開發人員跟上對新內容的持續需求，以保留玩家。內容的多樣性和大小會因遊戲類型和遊戲的播放裝置而有所不同。無論遊戲的系統為何，請保護遊戲的內容不受未經授權的存取。

最佳實務

- GAMESEC04-BP01 將可下載內容的存取限制為授權用戶端和使用者
- GAMESEC04-BP02 限制對授權內容交付網路 (CDNs原始存取

- GAMESEC04-BP03 實作地理限制以限制未經授權的存取
- GAMESEC04-BP04 使用數位版權管理 (DRM) 解決方案限制對內容的存取

GAMESEC04-BP01 將可下載內容的存取限制為授權用戶端和使用者

限制授權應用程式和用戶端存取遊戲內容。考慮使用 Amazon S3 作為經濟實惠且可擴展的原始伺服器，以存放可下載的遊戲內容，並考慮使用 Amazon CloudFront 為玩家提供全域效能的內容交付。這兩種服務都提供內建機制，以限制對已儲存資料的存取，例如限制對已驗證使用者的存取。

未建立此最佳實務時的曝險等級：高

實作指引

授予存取存放在 Amazon S3 中的內容

當您需要授予存取存放在 S3 中的內容的權限時，需要考慮幾個因素。根據預設，只有建立 S3 儲存貯體 AWS 帳戶 的 可以存取其中存放的物件。若要授予內部應用程式的存取權和管理存放在 Amazon S3 儲存貯體中的內容，請使用 AWS Identity and Access Management (IAM) 建立提供適當存取權的政策。

IAM 角色可以與託管在 服務中的聯合身分使用者、系統或應用程式相關聯，例如 Amazon EC2 AWS Lambda，以及託管在 Amazon EKS 和 Amazon ECS 中的容器型應用程式。例如，您可以使用 AWS SDK 或 AWS CLI 在 S3 儲存貯體中發佈和管理遊戲內容資產。若要支援此使用案例，您可以建立具有適當存取權的 IAM 角色，以將遊戲內容讀取和寫入 S3 儲存貯體，並將其與託管軟體和指令碼的 EC2 執行個體建立關聯。

您可以為您的儲存貯體和特定物件定義資源型政策。S3 儲存貯體政策與 S3 儲存貯體相關聯，可用於限制對儲存貯體和其中物件的存取，以及授予來自其他帳戶的 Amazon S3 資源存取權。例如，在多個團隊或個別遊戲開發工作室正在處理相同遊戲內容，且需要對 Amazon S3 中集中託管內容的相同存取權的情況下，您可以使用 S3 儲存貯體政策來定義跨帳戶存取 S3 資源的許可。考慮使用 S3 存取點，這可以透過建立具有每個應用程式或一組應用程式特定名稱和許可的存取點，簡化對共用資料的資料存取管理。Amazon S3 文件包含 Amazon S3 中存取控制的其他最佳實務。

```
Granting short-term access to your content
```

當只需要在特定的有限時間內存取時， 會產生暫時 URLs以授予您內容的短期存取權。Amazon S3 支援產生預先簽章URLs，這可讓物件擁有者授予 Amazon S3 中物件的時間限制存取，而無需更新儲存

貯體政策。如此一來，被授予存取的最終使用者或應用程式就不需要擁有帳戶或 IAM 許可，而是使用預先簽章的 URL 來存取內容。

這是在各種遊戲使用案例中常用的最佳實務，例如授予授權玩家存取他們有權下載的內容，以及暫時存取有限的遊戲內容。預先簽章URLs 也可以用來提供將內容上傳至 S3 儲存貯體的暫時許可。例如，您可以考慮使用預先簽章的 URL 為玩家提供上傳用戶端日誌的存取權，以協助支援團隊疑難排解玩家支援案例。

使用內容交付網路來提供內容的存取權

雖然您的應用程式、遊戲開發人員、藝術家和其他人員可能需要直接存取 S3 儲存貯體中的內容以進行開發和管理，但請使用內容交付網路來提供透過網際網路公開提供給玩家或其他使用者的內容存取權。這種方法透過快取經常存取的內容來改善下載效能並降低成本。Amazon CloudFront 可以透過快取並將內容交付到更接近玩家的位置，同時減少遊戲下載原始伺服器的負載，例如 Amazon S3。

建議您不要直接從 S3 儲存貯體提供公有內容，而是使用 CloudFront 來保持此內容的私密性並公開提供。CloudFront 可設定為要求玩家使用簽章的 URLs或簽章的 Cookie 來存取您的私有內容（例如僅限付費玩家的新遊戲下載）。然後，您可以開發應用程式，以建立已簽署 URLs並將其分發給已驗證的使用者，或傳送設定已驗證使用者已簽章 Cookie 的 set-Cookie 標頭。當您建立已簽章URLs 或已簽章的 Cookie 以控制對檔案的存取時，您可以指定結束日期和時間，之後 URL 和 Cookie 將不再有效。

或者，您也可以指定可用於存取內容之電腦的 IP 地址或地址範圍，如果您想要限制對特定遊戲開發工作室合作夥伴或承包商網路的存取，這會很有用。當您想要提供對多個受限檔案的存取權，或者您不想變更目前的 URLs Cookie。當您想要限制對個別檔案的存取，或您的使用者使用的用戶端不支援 Cookie 時，請使用已簽署URLs。已簽署的網址優先於已簽署的 Cookie。

實作步驟

- 使用 IAM 角色和儲存貯體政策，為內部應用程式、團隊或跨帳戶案例授予 S3 儲存貯體的適當存取權。
- 產生預先簽章URLs，以授予 S3 物件的短期存取權，適用於可下載的內容或暫時上傳，例如用戶端日誌。
- 搭配已簽章URLs 或 Cookie 使用 Amazon CloudFront，以更安全地為已驗證的使用者提供私有內容

GAMESEC04-BP02 限制對授權內容交付網路 (CDNs原始存取

封鎖使用者規避您的內容交付網路，以直接從原始伺服器存取內容，例如 Amazon S3 儲存貯體。請務必將原始伺服器的存取限制為您授權CDNs，如此可降低因不必要地從原始伺服器提供內容而產生的資料傳輸成本。它還可以透過相同的進入點來流動對原始伺服器的公開存取，以改善您的安全狀態，您可

以在其中部署邊緣安全控制，例如第 7 AWS WAF 層篩選、注入和檢查安全相關的 HTTP 請求參數，以及分散式拒絕服務 (DDoS) 保護。

未建立此最佳實務時的曝險等級：高

實作指引

若要為 Amazon S3 原始伺服器實作這些控制項，您可以使用 Amazon CloudFront 原始存取身分 (OAI)，以驗證對 S3 物件的請求是否來自您的 CloudFront 分佈。 AWS WAF 與您的 CloudFront 分佈建立關聯以提供 layer-7 篩選。不過，如果您要從其他 CDNs 提供內容，您可以設定 CDN 將一或多個自訂 HTTP 標頭插入原始伺服器請求，以供 檢查 AWS WAF ，以驗證傳入流量是否來自授權的 CDN 供應商。

當您的原始伺服器託管在 Application Load Balancer (ALB) 後方時，此方法也有助於防止使用者規避您的 CDN 供應商。ALBs 可與 建立關聯 AWS WAF 以進行 layer-7 保護。您可以設定 AWS WAF 插入由 ALB 檢查的自訂 HTTP 標頭，以處理和檢查傳入負載平衡器的流量 AWS WAF。

客戶範例

AnyCompany Games 實作原始存取限制，以協助保護其遊戲資產、可下載內容和修補檔案，避免未經授權的直接存取，讓玩家略過安全檢查或取得高級內容，而無需進行適當的身分驗證。這種方法允許他們透過集中點監控內容存取模式，讓他們直接識別可能表示存在協同攻擊或未經授權的內容重新分發的可疑下載行為。

實作步驟

- 使用 Amazon CloudFront 原始存取身分 (OAI) 來限制對 S3 物件的直接存取
- AWS WAF 與 CloudFront 或 ALB 建立關聯，以提供 layer-7 篩選，並協助防範 DDoS 攻擊和惡意請求。
- 在 Cloudfront 中設定自訂 HTTP 標頭，以確認傳入流量來自授權來源。

GAMESEC04-BP03 實作地理限制以限制未經授權的存取

當玩家請求您的內容時，Amazon CloudFront 會從最近的節點提供請求的內容，無論玩家位於何處。不過，在某些情況下，您可能需要限制世界各地的使用者如何存取您的內容。例如，您可能有一個滾動遊戲部署策略，以country-by-country為基礎分階段發佈內容，或者您可能必須遵守國家/地區特定的存取控制。

未建立此最佳實務時的曝險等級：高

實作指引

您可以使用地理限制，也稱為地理封鎖，封鎖特定地理位置的玩家存取您透過 CloudFront 分佈分佈的內容。此功能可讓您限制存取與分佈相關聯的檔案，並在國家/地區層級限制存取。或者，您可以使用第三方地理位置服務來限制存取與分佈相關聯的檔案子集，或限制比國家/地區層級更精細的存取。

透過使用 CloudFront 地理限制，您可以只允許玩家在核准國家/地區允許清單上的其中一個國家/地區存取您的內容。如果玩家位於被禁止國家/地區拒絕名單中的其中一個國家/地區，您也可以封鎖其存取您的內容。如果從封鎖的地理位置收到請求，CloudFront 會將 403 禁止的 HTTP 狀態碼傳回給玩家。請務必注意，這適用於非敏感內容，不應用作 PII 或敏感遊戲成品的獨立保護。

實作步驟

- 使用 CloudFront 地理限制，根據國家/地區層級允許或拒絕清單來允許或拒絕內容存取。
- 傳回來自封鎖地理位置之請求的 403 禁止 HTTP 狀態碼。
- 避免完全依賴地理限制來保護敏感內容或 PII

GAMESEC04-BP04 使用數位版權管理 (DRM) 解決方案限制對內容的存取

考慮使用強大的加密工具來限制對遊戲內容的存取，例如數位版權管理 (DRM) 解決方案。這類解決方案可用來加密您的私有內容，並將解密金鑰分發給授權的玩家。

未建立此最佳實務時的曝險等級：中

實作指引

當您想要允許玩家儘早下載遊戲內容，但不希望他們能夠在預定時間之前存取或播放內容時，建議使用 DRM 解決方案。例如，在允許玩家預先排序遊戲並設定其遊戲用戶端以提早開始下載加密檔案的情況下，這種情況很常見。此策略會驗證遊戲已下載並準備好在遊戲正式發行後進行遊戲。遊戲發行後，玩家的遊戲用戶端可以從 DRM 後端解決方案請求解密金鑰，以便解密先前下載的檔案並開始玩遊戲。

DRM 系統也用於在授權玩家下載和安裝遊戲之後，封鎖未經授權的重新分發和操作遊戲。DRM 系統需要與原始伺服器整合，才能交換加密金鑰並授權玩家擷取解密金鑰。商業 DRM 提供者提供各種解決方案，具有不同裝置的功能和支援。

實作步驟

- 使用 DRM 解決方案加密私有遊戲內容，並將解密金鑰分發給授權的玩家。
- 啟用預先排序遊戲的預先下載加密檔案，在發行時以解密金鑰解鎖存取。
- 將 DRM 系統與原始伺服器整合，以管理加密金鑰並封鎖未經授權的重新分發或操作內容。

偵測

GAMESEC05：如何監控和分析遊戲中的玩家使用行為？

監控和分析玩家使用行為對於遊戲工作室至關重要，因為它可讓您偵測可能危及遊戲完整性和玩家安全的安全威脅、作弊和其他形式的濫用行為。透過追蹤異常的進展率、異常的遊戲內交易或可疑的通訊行為等模式，您可以在潛在作弊者、詐騙帳戶或協調威脅對玩家體驗造成重大影響之前對其進行識別。

最佳實務

- GAMESEC05-BP01 實作全面的資料收集策略來監控玩家行為
- GAMESEC05-BP02 收集、存放和分析玩家用量日誌，以偵測不適當的行為

GAMESEC05-BP01 實作全面的資料收集策略來監控玩家行為

為了維持正面的玩家體驗，請實作全面的資料收集和分析策略。擷取、儲存和分析相關資料可讓您深入了解玩家如何與遊戲的功能和彼此互動。這種資料驅動型方法可以引導決策、增強玩家參與度和保留率、最佳化獲利策略，最終改善整體玩家體驗。

未建立此最佳實務時的曝險等級：中

實作指引

實作資料收集系統來擷取並記錄相關的玩家動作，例如遊戲工作階段、進度、成就、購買、與遊戲元素的互動，以及社交活動。收集伺服器端資料，例如伺服器負載、網路流量和錯誤日誌，以監控技術效能並識別潛在問題。透過問卷、論壇、支援票證和社交媒體管道收集玩家意見回饋，以了解他們的體驗和偏好設定。

儲存您的遊戲資料時，請建立集中式資料倉儲或資料湖來存放和組織收集的資料，並實作管道以進行資料清理、轉換和彙總，以準備資料以進行有效的分析。

儲存資料後，請加以分析以取得洞見，例如玩家保留率和流失率、獲利策略，以及透過資料視覺化工具使用特徵。

實作步驟

- 擷取並記錄玩家動作、伺服器端指標和意見回饋，以監控互動和技術效能。

- 使用 Amazon Redshift 或 S3 資料湖等集中式資料倉儲來存放、清理、轉換和整理遊戲資料以供分析。
- 使用視覺化工具分析收集的資料，例如 Amazon Quicksight，以深入了解玩家保留、獲利和特徵用量。

GAMESEC05-BP02 收集、存放和分析玩家用量日誌，以偵測不適當的行為

檢測您的遊戲以收集日誌，以了解玩家如何使用遊戲的功能，以及他們如何與其他玩家互動。然後，您可以封鎖可能降低玩家體驗的未經授權活動。

未建立此最佳實務時的曝險等級：中

實作指引

使用 Amazon CloudWatch Logs 或 Amazon OpenSearch Service 等記錄解決方案，或透過 Datadog、Sumo Logic、New Relic、Honeycomb.io 或 Splunk 等合作夥伴的解決方案 AWS ，將結構化日誌事件傳送至遊戲分析管道。建構這些玩家用量日誌，以便它們可用於偵測何時需要調查玩家的特定動作。

擷取資料之後，請考慮實作工具來偵測不當的使用行為。例如，如果您的遊戲具有社交功能，例如遊戲內玩家傳訊、語音聊天或線上論壇，請將這些玩家參與的日誌儲存為可用於管制目的分析的格式。

設定遊戲的語音聊天功能，將錄音匯出至 Amazon S3，並使用 Amazon Transcribe 將音訊語音轉換為文字格式，以供儲存以供處理。或者，您可以透過直接整合遊戲後端語音聊天服務與 Transcribe API 來執行即時串流轉錄，以即時轉錄串流音訊。管制團隊可以手動檢閱內容，一旦內容採用標準格式，您也可以使用 AWS AI/ML 服務自動執行管制。Amazon Comprehend 可用來執行自然語言處理 (NLP)，以從非結構化文字中發現資訊，這些文字可以將對話分類並組織成相關主題，並識別不適當的行為，例如褻瀆行為。

實作步驟

- 收集、存放和分析玩家用量日誌。
- 使用 AWS 服務進行人工智慧和機器學習，更有效率地檢閱並深入了解您的玩家用量日誌。

基礎設施保護

如需適用於遊戲工作負載之安全性的基礎設施保護最佳實務，請參閱 Well-Architected Framework 白皮書。

GAMESEC06：如何監控和回應基礎設施威脅？

監控和回應基礎設施威脅對於遊戲工作室至關重要，因為其基礎設施代表支援數百萬並行玩家、處理真實交易，以及存放寶貴玩家資料和專屬遊戲內容的骨幹。遊戲工作室必須實作監控系統和事件回應程序，以保持玩家體驗和業務操作的完整性。

最佳實務

- GAMESEC06-BP01 使用工具來偵測和回應對基礎設施的威脅
- GAMESEC06-BP02 使用人工智慧和機器學習工具自動化基礎設施保護策略的各個層面
- GAMESEC06-BP03 使用系統層級日誌的洞見來持續改善您的基礎設施保護策略

GAMESEC06-BP01 使用工具來偵測和回應對基礎設施的威脅

若要持續監控 AWS 環境中的惡意活動和未經授權的行為，請考慮使用 Amazon GuardDuty。GuardDuty 透過監控您環境中的帳戶行為、網路活動和資料存取模式來識別威脅。它分析多個資料來源的事件，例如 CloudTrail 事件日誌、Amazon VPC 流程日誌和 DNS 日誌，以找出潛在威脅。透過與 Amazon CloudWatch Events 和 Lambda 整合，GuardDuty 警示可以自動轉送至相關的安全團隊以進行進一步分析。

未建立此最佳實務時的曝險等級：高

實作指引

AWS Security Hub CSPM 提供 中安全狀態的完整檢視 AWS ，並根據安全產業標準和最佳實務檢查您的環境。Security Hub CSPM 會從跨 AWS 帳戶、 服務和支援的第三方合作夥伴產品收集安全資料，並分析您的安全趨勢並識別最高優先順序的安全問題。Amazon GuardDuty 與 Security Hub CSPM 整合可讓您將問題清單從 GuardDuty 傳送至 Security Hub CSPM。然後，Security Hub CSPM 可以在分析您的安全狀態時包含這些調查結果。

惡意行為者通常會使用機器人來接管帳戶，並在遊戲中作弊。WAF Bot Control 可讓您掌握和控制常見和普遍的機器人流量，這些流量可能會耗用過多的資源、扭曲指標、導致停機時間，或執行其他不需要的活動。

勒索軟體是惡意程式碼，旨在未經授權存取系統和資料集，並加密該資料以封鎖合法玩家的存取。勒索軟體鎖定玩家離開系統並加密敏感資料後，網路犯罪者會先要求支付贖金，再提供解密金鑰來解鎖資料。組織可能因惡意事件而完全關閉，進而產生大量成本並降低業務生產力。請參閱保護您的 AWS 雲

端 環境免受勒索軟體的影響，了解您可以套用的最佳實務，以增強您在事件發生之前、期間和之後對抗勒索軟體的能力。

您的遊戲可能讓玩家能夠透過 Amazon Connect 等呼叫中心或使用 Amazon Lex 來聯絡玩家支援客服人員。Amazon Connect 支援監控即時和錄製的對話。若要分析玩家與玩家之間的互動，支援使用 Amazon Lex 建置的聊天機器人，您可以將這些互動的對話日誌儲存在 Amazon CloudWatch Logs 中，您可以將這些對話日誌匯出至 Amazon S3 並依先前所述進行分析。

最後，在基礎設施保護策略中執行滲透測試練習。無論您是在內部或透過 AWS 合作夥伴執行這些評估，請遵守AWS 客戶支援政策進行滲透測試。

實作步驟

- 使用 Amazon GuardDuty 監控帳戶行為、網路活動和資料存取模式是否有威脅，並與 Security Hub CSPM 整合以獲得統一的安全檢視。
- 實作 AWS WAF Bot Control，以協助偵測和緩解可能損害資源和玩家體驗的機器人流量。
- 定期執行滲透測試練習，遵守 AWS 客戶支援政策，以評估和強化您的安全狀態。

GAMESEC06-BP02 使用人工智慧和機器學習工具自動化基礎設施保護策略的各個層面

Amazon Lookout for Metrics 使用機器學習自動偵測和診斷業務和營運資料中的異常，並以更高的速度和準確性監控對業務最重要的指標。該服務也可直接診斷異常的根本原因，例如收入突然下降、登入、交易或保留。它不需要遊戲開發人員擁有 ML 經驗來設定，並且可以連線到熱門資料來源，包括 Amazon S3、Amazon CloudWatch、Amazon RDS、Amazon Redshift，以及許多 SaaS 應用程式。例如，您可以將 Amazon Lookout for Metrics 與 Game Analytics Pipeline 和其他資料來源整合，開始分析行為以偵測異常。

未建立此最佳實務時的曝險等級：高

實作指引

或者，您可以選擇使用 Amazon SageMaker AI 建置、訓練和託管自訂機器學習模型，以解決使用案例，例如內容管制、毒性偵測、詐騙偵測等。

客戶範例

AnyCompany Games 使用 Amazon Lookout for Metrics 來自動偵測伺服器效能、玩家登入嘗試或交易量中的異常模式，這些模式可能表示來自惡意人士的威脅。此外，他們已使用 Amazon SageMaker AI

來開發自訂機器學習模型，持續分析網路流量模式和玩家行為，以協助識別協調的威脅，例如嘗試利用其虛擬經濟的機器人網路。

這種自動化方法可讓安全團隊專注於調查和回應真正的威脅，而不是手動監控數千個指標，同時確保在對遊戲可用性或玩家安全造成重大影響之前偵測和解決新興威脅模式。

實作步驟

- 使用 Amazon Lookout for Metrics 協助自動偵測和診斷關鍵業務和營運資料的異常
- 將 Amazon Lookout for Metrics 與 Game Analytics Pipeline、Amazon S3 或 CloudWatch 等資料來源整合，以監控營收、登入和保留等指標。
- 使用 Amazon SageMaker AI 來建置、訓練和託管自訂機器學習模型，以用於詐騙偵測、詐騙預防和內容管制等進階使用案例。

GAMESEC06-BP03 使用系統層級日誌的洞見來持續改善您的基礎設施保護策略

從相關服務擷取和存放系統層級日誌，例如 S3 伺服器存取日誌、CloudFront 存取日誌和 ALB 存取日誌。這些日誌可以存放在您帳戶中的 S3 儲存貯體中，有助於將遊戲中的玩家使用資訊與系統層級資訊建立關聯，包括連線詳細資訊，例如 IP 地址、請求標頭，以及您在遊戲後端中可能設定的相關請求操作和篩選。您可以將這些日誌傳送到先前提到的相同日誌解決方案，而且您可以使用 SQL 查詢搭配 Amazon Athena 進行分析，而不需要將日誌移出 Amazon S3。

未建立此最佳實務時的曝險等級：中

實作指引

Access Analyzer for S3 是一項功能，可監控您的儲存貯體存取政策，確保政策僅提供 Amazon S3 資源的預期存取權。Access Analyzer for S3 會評估您的儲存貯體存取政策，並可讓您探索和快速修復具有潛在意外存取的儲存貯體。

實作步驟

- 使用 AWS 服務進行威脅偵測和事件回應，以自動化基礎設施保護策略的各個層面。
- 透過人工智慧和機器學習的系統層級日誌 AWS 和服務，深入了解您的基礎設施保護。

資料保護

開發和建構遊戲時，請考慮您的 Studio 正在收集哪些類型的資料，以及您決定如何保護它。在這個安全層面中要探索的主題包括：

- 您選擇識別和分類資料的方式
- 如何保護靜態資料
- 如何保護傳輸中的資料

Games Lens 沒有特定的資料保護最佳實務。如需安全資料保護的最佳實務，請參閱 Well-Architected Framework 白皮書。

事件回應

GAMESEC07：您如何定義和強制執行政策，以回應玩家不當和濫用行為？

玩家不當和濫用行為可能會大幅影響玩家的體驗。錯誤的玩家行為可能會讓合法玩家退場，導致玩家保留率降低、遊戲內購買收入降低，以及可能損害遊戲評價和未來銷售的負面評論。

定義可在玩家之間提升正面動作的政策，並決定如何強制執行這些政策。

最佳實務

- GAMESEC07-BP01 實作事件回應計劃，以處理不良行為者和濫用行為
- GAMESEC07-BP02 禁止與不法份子相關聯的帳戶

GAMESEC07-BP01 實作事件回應計劃，以處理不良行為者和濫用行為

建立行動計劃，以回應遊戲中的惡意行為者和濫用行為。

未建立此最佳實務時的曝險等級：中

實作指引

考慮一些因素，例如何時暫停或永久禁止玩家，以及停用暫停玩家的登入資料多久。

客戶範例

AnyCompany Games 會建立分層事件回應系統，其中不適當的聊天訊息等次要違規會導致 24 小時帳戶自動停用，而詐騙或騷擾等更嚴重的違規則會觸發立即的 7 天停用，並經人工主持人強制審核。

此外，AnyCompany Games 會建立升級程序，其中重複違規者會逐漸面臨較長的停權。他們會建立申訴程序，允許錯誤標記的玩家對自動化動作提出爭論，同時透過身分驗證要求來維護安全性。

GAMESEC07-BP02 禁止與不法份子相關聯的帳戶

如果未緩解，遊戲中的濫用行為可能會繼續對其他人的遊戲體驗產生負面影響，並應盡快緩解。實作程序，對確認違反您的服務條款的不法份子實施禁止或其他形式的限制。

未建立此最佳實務時的曝險等級：高

實作指引

一般而言，用於判斷強加這類限制之情況的規則和評估程序，將由玩家社群團隊或組織中的信任和安全團隊等人員決定。標記惡意執行者之後，請執行預先確定的工作流程，以對已識別的玩家採取行動。

例如， AWS Step Functions 和 AWS Lambda函數可用來執行接受一批玩家帳戶做為輸入的自動化工作流程。工作流程接著會更新名為 Bans 的 Amazon DynamoDB 資料表中的項目，其中包括玩家帳戶、禁止原因和持續時間的詳細資訊。

根據您遊戲和帳戶管理系統的設計，以及您遇到的濫用行為類型，請維護與帳戶管理系統分開的禁止記錄系統。您可能不想從帳戶管理系統關閉玩家的帳戶，而是選擇關閉他們玩遊戲的能力。這在玩家的帳戶登入資料用於存取具有不同服務條款或政策的多個遊戲的情況下非常有用。

實作步驟

- 定義並強制執行政策，以回應惡意人士的濫用行為。
- 使用 AWS 服務來自動化您對不法份子的回應。

Resources

- AWS 安全事件回應技術指南
- AWS Machine Learning部落格：使用 Amazon Rekognition 人臉活體偵測真實和即時使用者並阻止惡意人士
- AWS 遊戲解決方案：社群運作狀態

應用程式安全

GAMESEC08：如何保護您的 CI/CD 管道？

遊戲開發 CI/CD 管道通常包含高可用性的來源控制伺服器和儲存體、用於執行建置的運算資源，以及用於執行自動化測試的軟體，以及來自開發機器的適當網路連線。保護您的 CI/CD 管道對於保護敏感資訊、維護程式碼完整性和維護受信任版本至關重要。內嵌控管和護欄可讓開發人員保持敏捷性，同時維持良好的安全實務。

由於遊戲通常會處理付款處理、儲存個人資訊，以及維護值得實際金錢的虛擬經濟體，因此開發過程中的安全違規可能會導致重大財務損失、法規處罰和玩家信任損失。

透過整合保護措施，組織可以維持對軟體交付程序的可見性和控制，實現快速的事件回應並促進安全編碼實務的文化。

最佳實務

- GAMESEC08-BP01 在 CI/CD 管道的每個階段套用安全性

GAMESEC08-BP01 在 CI/CD 管道的每個階段套用安全性

存取控制、職責分離和稽核線索等護欄可提供保護，防止未經授權的存取或惡意活動。

未建立此最佳實務時的曝險等級：中

實作指引

您的人員、程序和技術也應保護管道的安全。最接近程式碼的人員必須建立安全編碼實務，並確保遵循這些實務。持續反覆執行您的程序，以驗證整個管道的安全性層級是否一致。最後，實作技術來驗證未略過最佳實務和程序。

客戶範例

AnyCompany Games 實作以角色為基礎的存取控制，其中只有資深開發人員可以核准其反作弊系統程式碼的變更，同時需要安全團隊成員對處理玩家付款資料的元件進行強制性程式碼檢閱。

其 CI/CD 管道會自動執行威脅模型驗證檢查，確保玩家交易市場等新功能已針對先前識別的攻擊媒介進行測試，例如項目重複入侵或詐騙交易嘗試。

實作步驟

- 根據最低權限原則提供使用者許可。
- 使用 AWS CloudTrail 來稽核在管道中使用的服務之間進行的 API 呼叫。
- 使用預先遞交掛鉤來驗證程式碼是否遵循一般實務和公司政策。

自動化安全性

GAMESEC09：如何自動化 CI/CD 管道內的安全？

在您的 CI/CD 管道中整合安全措施，以在整個開發生命週期中維持健全的安全狀態。此程序提供許多與擁有管道安全性相同的優點。擁有安全的 CI/CD 管道可降低可能導致延遲遊戲開發時間表的安全事件可能性。

在 CI/CD 管道中提供安全性涉及在開發週期的每個階段實作安全最佳實務和工具。在管道中擁有安全也可讓您同時縮短安全審查的時間。

在 CI/CD 管道中自動化安全性對於驗證安全控制是否在每次程式碼變更時持續實作和測試至關重要。實作適當的工具和自動化可以提供安全可靠的遊戲。

最佳實務

- GAMESEC09-BP01 整合工具和自動化，以減少安全審查的平均時間

GAMESEC09-BP01 整合工具和自動化，以減少安全審查的平均時間

為了識別安全漏洞，組織可以使用各種不同的工具和服務，例如靜態應用程式安全測試 (SAST) 和動態應用程式安全測試 (DAST)。SAST 是檢閱原始程式碼並判斷安全性漏洞的一種方式。DAST 是一種測試程式碼的黑框方式，可在不查看原始程式碼的情況下測試應用程式。

未建立此最佳實務時的曝險等級：中

實作指引

組織可以使用的另一個工具是軟體合成分析 (SCA)，可評估第三方或開放原始碼相依性的安全性。如需更手動的方法，可以在整個管道中實作安全程式碼檢閱。

客戶範例

AnyCompany Games 使用 SAST 工具，在開發過程中自動標記潛在的安全漏洞。他們也會使用 DAST 工具來模擬對執行遊戲組建的威脅，以驗證安全控制是否如預期般運作。此外，AnyCompany Games 將相依性掃描工具整合到其開發程序中，以自動識別第三方程式庫和遊戲引擎中的已知漏洞。

實作步驟

- 使用 Amazon CodeGuru 做為 SAST 工具。
- 使用開放原始碼工具，例如 OWASP 相依性檢查、SonarQube 或 OWASPZap。

Resources

- 開發人員的安全

威脅建模

GAMESEC10：如何將威脅建模整合到組織的應用程式開發生命週期？

威脅建模是識別應用程式潛在威脅並排定其優先順序的程序，以及判斷可用於緩解這些威脅的解決方案。隨著遊戲發展為處理敏感使用者資料和真實交易的複雜連線系統，這種做法變得越來越重要。

將威脅建模整合為支援遊戲安全性的持續練習，不僅在初始設計階段，還會隨著遊戲持續成長和發展。

最佳實務

- GAMESEC10-BP01 決定何時及如何在整個應用程式開發生命週期中完成威脅建模練習

GAMESEC10-BP01 決定何時及如何在整個應用程式開發生命週期中完成威脅建模練習

沒有一種最佳的方法可以處理威脅建模。執行此作業的時間和方式詳細資訊會根據遊戲工作室的獨特需求而有所不同。例如，根據工作室的大小，您可能讓團隊成員參與威脅建模程序的一或多個層面。

未建立此最佳實務時的曝險等級：中

實作指引

AWS 安全部落格提供在設計威脅建模策略時需要注意的考量事項概觀，例如：

- 哪些團隊成員和角色應該參與威脅建模
- 如何判斷要使用的適當工作流程工具
- 如何判斷威脅建模各方面的擁有權
- 如何識別和評估要在工作負載設計中使用的安全控制

客戶範例

AnyCompany Games 首先編目寶貴的資產，例如玩家資料、遊戲程式碼和演算法、遊戲內貨幣、使用者產生的內容，以及未發行的內容或專屬引擎等智慧財產權。他們會考慮不同類型的潛在惡意行為者，例如尋求不公平優勢的作弊者、嘗試竊取個人或財務資料的惡意行為者，以及嘗試中斷遊戲的惡意使用者。

在整個開發過程中，AnyCompany Games 會使用威脅模型來引導安全編碼實務，並影響測試策略以專注於高風險領域。在遊戲啟動之前，他們會執行全面的威脅建模審查，以評估預期的玩家負載和未經授權的存取嘗試準備程度，並準備事件回應程序。

實作步驟

- 在 CI/CD 管道的每個階段實作護欄。
- 使用自動化和工具來提高應用程式安全性審查的效率。
- 使用威脅建模作為改善應用程式安全性的程序。

Resources

- AWS 安全部落格：如何處理威脅建模
- NIST：以資料為中心的系統威脅建模指南
- 為建置器正確建立威脅模型 – AWS 技能建置器虛擬自定進度訓練
- 建置器的威脅建模 – AWS Workshop

Resources

請參閱下列資源，進一步了解我們有關安全性的最佳實務。

相關文件：

- 常見 Amazon Cognito 案例

- 使用已簽章URLs
- 使用頻道流程，從 Amazon Chime SDK 訊息中的訊息中移除褻瀆和敏感內容
- Amazon GameLift 的安全性
- 使用 Amazon CloudFront 保護內容交付
- 安全回應指南
- AWS DDoS 彈性的最佳實務
- 保護您的 AWS 雲端 環境免受勒索軟體攻擊

相關的合作夥伴解決方案：

- Datadog
- Sumo Logic
- Splunk
- Honeycomb.io
- New Relic
- AWS Marketplace - DRM 解決方案

相關訓練資料：

- Amazon Cognito 入門
- 安全自定進度訓練

可靠性

可靠性支柱包括系統從基礎設施或服務中斷中復原的能力、動態取得運算資源以滿足需求，以及緩解組態錯誤或暫時性網路問題等中斷。

重點領域

- 設計原則
- 基礎
- 工作負載架構
- 變更管理
- 故障管理
- Resources

設計原則

除了 AWS Well-Architected Framework 白皮書中的設計原則之外，以下是可提高遊戲工作負載雲端可靠性的設計原則：

- 為達到業務預測所需的尖峰玩家並行和系統可擴展性目標建立基準：在啟動遊戲之前以及在即時遊戲操作期間，針對尖峰時預期的並行玩家數量制定預估，以建立系統可擴展性的目標，以符合這些預測。這有助於建立遊戲可靠性的基準。透過驗證擴展系統是否正常管理作用中玩家工作階段，定義擴展政策以自動因應需求變更，而不會影響可用性。
- 測量您的可靠性和對玩家體驗的影響：定義代表遊戲運作狀態的關鍵效能指標 (KPIs)。監控基礎設施和遊戲功能變更對可靠性的影響。

基礎

若要實現可靠性，系統必須具有妥善規劃的基礎和監控，並具有處理需求或需求變更的機制。系統應設計為偵測故障並自動自行修復。

沒有 Games Lens 特有的基礎最佳實務。如需適用於遊戲工作負載之可靠性的基礎最佳實務，請參閱 Well-Architected Framework 白皮書。

工作負載架構

GAMEREL01：您的遊戲架構是否利用雲端的彈性？

AWS 基礎設施是以區域和可用區域為基礎建置。 AWS 區域 提供多個實體隔離和隔離的可用區域，這些區域使用低延遲、高輸送量和高度備援聯網進行連接。這些建構可用於建構以可靠性目標為重心的工作負載。

最佳實務

- GAMEREL01-BP01 將遊戲基礎設施分散到多個可用區域和區域，以改善彈性

GAMEREL01-BP01 將遊戲基礎設施分散到多個可用區域和區域，以改善彈性

為了將本地化基礎設施損害對玩家的影響降至最低，您應該將基礎設施部署平均分散到足夠的獨立位置，以承受非預期的損害，同時仍有足夠的容量來滿足您的玩家需求。

未建立此最佳實務時的曝險等級：高

實作指引

部署遊戲基礎設施時，建議您將容量均勻分散到區域中的多個可用區域，以便能夠承受對一或多個可用區域的中斷，而不會中斷玩家體驗。Web 應用程式等遊戲後端服務應跨多個可用區域進行負載平衡，或使用 AWS Lambda 和 Amazon API Gateway 等受管服務建置，該服務根據設計提供區域高可用性。同樣地，維護快取、資料庫、訊息佇列和儲存解決方案等狀態的元件應設計為跨多個可用區域提供持久的資料持久持久性，這是由 Amazon S3、DynamoDB 和 Amazon SQS 等服務的設計所提供，並且可以在其他服務中設定。

設計遊戲伺服器託管架構以實現彈性時，請在 內的可用區域間統一部署遊戲伺服器機群 AWS 區域 ，以最大化您對區域中可用運算容量的存取，並減少可用區域受損的影響範圍。例如，您可以將 Amazon EC2 Auto Scaling 設定為使用可用區域。如果 EC2 執行個體運作狀態不佳，EC2 Auto Scaling 可以取代執行個體，並在一或多個可用區域無法使用時，將執行個體啟動至其他可用區域。

對於關鍵基礎設施，例如身分驗證、佈建跨多個可用區域執行的最少可行執行個體數量，以及如果其中一個可用區域受損，請使用自動擴展來處理負載增加或容錯能力。

將您的遊戲基礎設施部署到多個區域，以最大化可用性。跨區域災難復原功能，例如 Aurora 全域資料庫和備援基礎設施，只要將簡單的 DNS 變更部署到次要區域，就可以在主要區域受損時提供服務持續性。雖然我們鼓勵您的遊戲後端服務實現高可用性，但此建議對您的遊戲伺服器特別重要。

例如，在多玩家遊戲中，遊戲伺服器的基礎設施容量可能會超過其他 服務的容量需求，因為遊戲伺服器用於為玩家託管遊戲工作階段。許多遊戲選擇將玩家分成邏輯遊戲區域（例如美國西部和東部）。為了簡化玩家體驗，並直接使用全球基礎設施來託管遊戲，請考慮將玩家面向的遊戲區域名稱與實際託管遊戲伺服器的底層雲端供應商區域或資料中心位置，以及本機區域或您自己的資料中心等其他基礎設施取消耦合，這些資料中心會託管支援該玩家遊戲區域的遊戲伺服器執行個體。

設計配對服務時，請跨區域部署具有個別軟體部署的多區域架構。將您的配對服務部署與託管遊戲伺服器執行個體的機群分離，以便您可以將玩家路由到區域中的遊戲伺服器，無論配對服務的哪個區域部署處理配對請求。

在您的配對實作中設計邏輯，以偏好符合您延遲和其他規則的遊戲伺服器區域，如果您的機群容量不足或有其他區域基礎設施中斷，則能夠回溯到將玩家路由到其他區域。

實作步驟

- 將遊戲基礎設施均勻分散到多個可用區域，以提供高可用性和彈性。
- 使用 Amazon S3 AWS Lambda、DynamoDB 和 SQS 等受管元件部署遊戲後端服務和具狀態元件，或設定自訂解決方案的負載平衡和耐久性。
- 使用 Aurora 全域資料庫等災難復原解決方案，以及與基礎實體位置分離的邏輯玩家面向區域，為關鍵遊戲服務和伺服器實作多區域部署。

Resources

- 靜態穩定性
- Amazon GameLift 遊戲工作階段佇列的最佳實務
- Amazon GameLift 多區域機群
- Aurora 全球災難復原資料庫

變更管理

GAMEREL02：如何擴展具狀態遊戲以適應需求的變化？

隨著您的玩家需求隨著時間的推移而波動，您的遊戲基礎設施應該能夠適應擴展以處理這些不斷變化的需求。雖然很難事先預測遊戲的熱門程度，但請設計架構方法，允許新增和移除基礎設施容量，以適應玩家群體中的波動。

最佳實務

- GAMEREL02-BP01 實作包含作用中玩家遊戲工作階段狀態的擴展策略
- GAMEREL02-BP02 支援在遊戲中使用多種 EC2 執行個體類型

GAMEREL02-BP01 實作包含作用中玩家遊戲工作階段狀態的擴展策略

實作解決方案，以納入主動連線玩家工作階段的狀態本質，並正常處理擴展活動，而不會中斷遊戲體驗。

未建立此最佳實務時的曝險等級：高

實作指引

在雲端開發遊戲的優點之一，就是可視需要自動擴展伺服器基礎設施以滿足需求的彈性。雖然無狀態或非同步遊戲和後端服務可以使用 Amazon EC2 Auto Scaling 政策、EKS Autoscaling 或類似技術來動態擴展，但遊戲開發人員通常需要更自訂方法來擴展有狀態或同步遊戲，以協助封鎖對作用中玩家工作階段的中斷。

實作步驟

- 對於具狀態遊戲， 會產生自訂指標，可用來監控玩家工作階段的狀態和可用的遊戲伺服器容量，這些指標可以做為自訂指標報告給 Amazon CloudWatch。使用 CloudWatch Synthetics 等應用程式監控來練習功能，檢查遊戲是否有簡單的運作狀態監控可能無法偵測到的功能受損。
- 使用自訂指標，實作遊戲伺服器擴展軟體，例如，作為使用 AWS Lambda 函數的無伺服器應用程式 AWS Fargate，或使用 AWS SDK 進行 API 呼叫來更新託管遊戲伺服器組建之 EC2 Auto Scaling 群組的最小、最大和所需容量設定，以管理專用遊戲伺服器執行個體的機群。
- 使用 Amazon GameLift 託管您的遊戲伺服器，並使用out-of-the-box遊戲伺服器自動擴展功能來為您管理此擴展程序。

Amazon GameLift 的自動擴展功能知道作用中的玩家工作階段，並可設定為封鎖正在託管玩家的遊戲伺服器執行個體終止或縮減。如需詳細資訊，請參閱使用 Amazon CloudWatch 監控 Amazon GameLift 伺服器。

GAMEREL02-BP02 支援在遊戲中使用多種 EC2 執行個體類型

使用 EC2 執行個體託管遊戲時，或者如果您使用 中 EC2 執行個體託管的容器 AWS 帳戶，請在託管策略中使用多個執行個體類型。透過使用多種執行個體類型，您可以增加遊戲擴展時可以使用的運算選項數量，以新增更多伺服器以支援玩家成長，這可在您偏好的執行個體類型無法使用時提高可靠性。

未建立此最佳實務時的曝險等級：高

實作指引

使用 Spot 執行個體託管遊戲時，請使用多種執行個體類型，因為 Spot 執行個體的可用性會根據客戶需求而波動。

在多個執行個體類型上測試您的遊戲，以符合您的成本和效能需求，並判斷執行個體類型的優先順序排名。Amazon EC2 Auto Scaling 支援使用多個執行個體類型和大小，以及將權重指派給組態中的每個執行個體類型，讓您可以實作運算選項的優先順序排名。

使用 Amazon GameLift 受管託管託管遊戲時，請決定遊戲所需的執行個體類型，以及如何對其執行遊戲伺服器程序（使用執行時間組態）。選擇機群的資源時，請考慮幾個因素，包括遊戲作業系統、執行個體類型（運算硬體），以及是否使用隨需執行個體、Spot 執行個體或兩者。Amazon GameLift 的託管成本主要取決於您使用的執行個體類型。如需詳細資訊，請參閱選擇受管機群的運算資源。

實作步驟

- 在 EC2 或容器上託管遊戲時，使用多種 EC2 執行個體類型來改善可靠性和擴展選項。
- 使用優先順序執行個體類型和權重設定 Amazon EC2 Auto Scaling 或 GameLift 機群，以最佳化成本和效能。
- 在各種執行個體類型上測試您的遊戲，以確認效能符合需求，並相應地調整您的託管策略。

故障管理

GAMEREL03：如何在基礎設施中斷期間持續遊戲狀態？

隨著您的遊戲基礎設施隨著時間經歷各種操作事件，遊戲的架構應設計成在基礎設施事件期間維持玩家體驗的持續性並保留遊戲狀態。若要處理這些事件，請實作監控、正常關機和狀態持續性機制，以驗證玩家的順暢遊戲體驗。

最佳實務

- GAMEREL03-BP01 監控遊戲伺服器中斷，並使用資料來改善託管架構以達成可靠性目標
- GAMEREL03-BP02 實作遊戲功能的鬆散耦合，在對玩家體驗影響最小的情況下處理失敗
- GAMEREL03-BP03 監控一段時間內的基礎設施事件，以測量對玩家行為的影響

GAMEREL03-BP01 監控遊戲伺服器中斷，並使用資料來改善託管架構以達成可靠性目標

監控遊戲伺服器指標，以及故障或效能降低的影響，例如負載下延遲增加，隨著時間的推移對玩家行為的影響，以便您可以調整遊戲伺服器託管策略，以滿足遊戲的可靠性需求。要降級的遊戲伺服器基礎設施如果會影響玩家，或當伺服器上沒有託管的作用中玩家工作階段時，應該立即從服務中移除。

未建立此最佳實務時的曝險等級：高

實作指引

對於將遊戲託管為 REST APIs的案例，系統可靠性可以像傳統 Web 應用程式架構一樣進行管理，其中流量可以在多個伺服器之間以分散式方式進行負載平衡，以降低伺服器故障的風險。

對於即時同步遊戲，遊戲工作階段通常託管在虛擬機器或遊戲伺服器執行個體上執行的遊戲伺服器程序上，因為遊戲狀態需要以高效能的方式維護，並複寫到連線的遊戲用戶端。此實作表示玩家的體驗與託管其遊戲工作階段的遊戲伺服器程序的效能和可靠性緊密結合。這種類型的架構使得管理遊戲伺服器的可靠性比傳統方法更複雜。

若要減輕遊戲伺服器故障的影響，請將您的遊戲設定為持續執行玩家遊戲狀態的非同步更新至高可用性快取或資料庫，例如 Amazon ElastiCache (Redis OSS) 或 Amazon MemoryDB。如果發生伺服器故障，可從外部資料存放區擷取玩家上次儲存的遊戲狀態，並在新的遊戲伺服器執行個體上還原其工作階段。

不過，這種方法會增加額外的成本和複雜性來管理此外部狀態，並可能不適用於快速步調或競爭性遊戲，其中狀態變更如此頻繁，而且在如此顯著的規模下發生，導致即使是高效能的記憶體內快取資料存放區，也會導致複寫延遲，這對於從中還原工作階段來說太重要，無法派上用場。對於這種性質的遊戲，最佳方法是接受伺服器遺失，並將玩家送回遊戲大廳以尋找另一個工作階段，或者您可以自動將其重新導向至另一個遊戲工作階段。

盡可能多地擷取造成伺服器中斷的有用日誌資料，以便稍後調查問題。Amazon GameLift 提供偵錯機群問題的指引，並提供遠端存取 Amazon GameLift 機群執行個體的能力。

實作步驟

- 監控遊戲伺服器指標是否有效能降級，並視需要移除或取代降級的伺服器，以維護可靠性。
- 在可行的情況下，使用 Amazon ElastiCache 或 MemoryDB 進行非同步遊戲狀態更新，以在伺服器故障後啟用工作階段復原。
- 擷取伺服器中斷的詳細日誌資料以進行調查和偵錯，利用 Amazon GameLift 等工具進行機群監控和遠端存取。

GAMEREL03-BP02 實作遊戲功能的鬆散耦合，在對玩家體驗影響最小的情況下處理失敗

解耦元件是指設計伺服器元件的概念，以便它們可以盡可能獨立運作。遊戲的某些方面很難解耦，因為資料必須盡可能保持最新狀態，才能為玩家提供良好的遊戲體驗。不過，許多元件和遊戲任務都可以解耦。例如，排行榜和統計資料服務對遊戲體驗來說並不重要，而且對這些服務的讀取和寫入可以從遊戲非同步執行。

未建立此最佳實務時的曝險等級：高

實作指引

為遊戲中的功能實作可自動停用的正常降級，或在偵測到問題時由管理員停用，以及設定依賴該功能的上游服務，以正常處理失敗。例如，如果特定玩家資料未正確載入遊戲用戶端，您應該考慮此資料是否對遊戲體驗至關重要。如果沒有，請設定遊戲用戶端以正常方式處理此失敗，而不會中斷玩家的體驗，選擇稍後當玩家重新瀏覽畫面時重試擷取此資料。

使用逾時、重試和退避等邏輯來處理錯誤和失敗。逾時可讓系統長時間不合理地停止停擺。重試可提供高可用性的暫時性和隨機錯誤。

定義可鬆散耦合至關鍵元件的非關鍵元件。鬆耦合可讓系統更具彈性，因為一個元件中的故障不會層疊到其他元件。當遊戲功能不需要與遊戲伺服器或後端的狀態連線時，您應該實作無狀態通訊協定，以動態擴展並從暫時性故障中復原。開發您的非關鍵元件，其中它可以使用 HTTP/JSON API 鬆散地與無狀態通訊協定耦合。從遊戲用戶端實作非同步和非封鎖的網路呼叫，將執行緩慢的遊戲功能或其他相依服務對玩家的影響降至最低。

若要透過鬆散耦合進一步改善彈性，請在可非同步處理的元件之間使用訊息服務，例如佇列、串流或主題型系統。此模型適用於不需要立即回應或確認已註冊請求的互動。此解決方案涉及一個產生事件的元件，以及另一個耗用它們的元件。這兩個元件不會透過直接point-to-point互動進行整合，而是透過耐用的儲存體或佇列層等中繼體進行整合。這也有助於在處理失敗時保留訊息，以改善系統的可靠性。

研究並選擇適當的簡訊機制，因為各種簡訊服務具有不同的特性，例如訂購和交付機制。設計等冪操作，讓所選的訊息系統至少傳遞訊息一次。例如，假設您的遊戲需要追蹤玩家播放時間、統計資料或其他相關資料的典型遊戲使用案例，這可能會在玩家並行峰值時導致高寫入輸送量使用案例。

若要實作可靠的架構，請考慮使用案例是否需要玩家感知的read-after-write一致性。一般而言，這類案例適用於非同步處理，並且可以透過實作寫入佇列模式來實現，其中請求會擷取到可擴展且耐用的訊息佇列，例如 Amazon SQS，並且可以使用消費者服務，例如 Lambda 函數，分批插入您的後端資料庫。這種方法比多個分散式元件之間的同步通訊更可靠，包括玩家的遊戲用戶端、您的後端 Web 和應用程式伺服器，以及您的內部資料庫系統。它還降低成本，因為後端資料庫不需要擴展以滿足峰值寫入輸送量，因為來自寫入佇列的消費者處理可以根據需要用來減慢此擷取速率。

實作步驟

- 將非關鍵元件，例如排行榜和統計資料服務與關鍵遊戲功能分離，以允許非同步操作並增強彈性。
- 使用逾時、重試和退避的邏輯對非關鍵功能實作正常降級，並確認遊戲用戶端在不中斷玩家體驗的情況下處理失敗。
- 使用 Amazon SQS 等傳訊系統，在元件之間進行非同步通訊，從而實現高輸送量使用案例的可擴展性、耐用性和可靠處理。

Resources

- 使用微服務架構建置高度可擴展且可靠的工作負載
- 使用無 AWS 伺服器服務整合微服務
- 了解微服務的非同步傳訊
- 上的可擴展遊戲開發模式簡介 AWS
- 實作寬容降級

GAMEREL03-BP03 監控一段時間內的基礎設施事件，以測量對玩家行為的影響

監控您的遊戲伺服器程序、遊戲伺服器執行個體指標和遊戲體驗指標，以判斷問題的根本原因。除了監控 CPU 和記憶體之外，您也可以設定監控與 EC2 執行個體網路限制相關的網路指標，以提醒您超出頻寬、packets-per-second數或其他網路層級的問題，這些問題可能表示您的伺服器資源佈建不足。

未建立此最佳實務時的曝險等級：高

實作指引

使用 CloudWatch Synthetics 檢查玩家體驗的關鍵路徑應用程式功能，例如無法登入或其他影響問題的服務。對於使用 Amazon GameLift 託管的遊戲伺服器，請考慮監控以下指標：

- GameServerInterruptions 和 InstanceInterruptions，可協助了解 Spot 執行個體可用性的限制如何影響使用 Spot 部署的遊戲伺服器。
- ServerProcessAbnormalTerminations，可用於偵測遊戲伺服器程序中的異常終止。

建議您維護遊戲伺服器可靠性的歷史指標資料。將此歷史資料用於報告目的，並將其與其他資料集聯結，以發現潛在的趨勢，並評估一段時間內可能由於遊戲伺服器問題而對玩家行為的影響。

Amazon CloudWatch 不會無限期保留指標，而且指標的儲存解析度會隨著時間而增加，因此請考慮將這些指標匯出至經濟實惠的長期儲存，例如 Amazon S3。您可以設定 CloudWatch Metric Streams 自動將指標從 CloudWatch 區域交付至您自己的 S3 儲存貯體，這些儲存貯體長期存放在 S3 Intelligent-Tiering 等儲存層中，最終使用 Amazon Glacier 封存。透過將指標放在 Amazon S3 中，即可隨時與資料湖中的其他資料集聯結，以便使用 Amazon Athena 進行互動式查詢。

實作步驟

- 使用 Amazon CloudWatch 和 CloudWatch Synthetics 進行關鍵路徑功能檢查，監控遊戲伺服器、執行個體和網路指標，包括頻寬和packet-per-second數限制。
- 追蹤 GameServerInterruptions 和 ServerProcessAbnormalTerminations 等 GameLift 特定指標，以評估 Spot 執行個體可用性的影響，並偵測異常的伺服器終止。 GameServerInterruptions ServerProcessAbnormalTerminations
- 將 CloudWatch 指標匯出至 Amazon S3 以進行長期儲存、使用 S3 Intelligent-Tiering 或 Glacier 等符合成本效益的層，並使用 Amazon Athena 等工具分析趨勢。

Resources

- Amazon EC2 執行個體層級網路效能指標發現新的洞見
- CloudWatch 指標串流 – 即時將 AWS 指標傳送至合作夥伴和您的應用程式

Resources

請參閱下列資源，進一步了解我們有關可靠性的最佳實務。

相關文件：

- 在 上練習持續整合和持續交付 AWS
- 自動擴展非同步任務佇列
- 設計您的 WorkloadService 架構
- 具有抖動的逾時、重試和退避
- Well-Architected Framework - 可靠性支柱
- 可靠擴展性的架構
- Amazon Builder 的程式庫
- 多玩家遊戲的大規模即時傳訊
- 上的可擴展遊戲開發模式簡介 AWS
- 在 上執行容器化微服務 AWS
- 雲端中的 Web 應用程式託管
- 建置可擴展且安全的多 VPC 網路基礎設施

相關影片：

- re：Invent 2020：Ubisoft：在 上建置多平台多玩家遊戲 AWS
- re：Invent 2018：Supercell- Scaling Mobile Games
- re：Invent 2019：CAPCOM 如何使用容器、資料和 ML 建置有趣的遊戲
- re：Invent 2018：在 Riot Games 全球化玩家帳戶，同時維持可用性
- re：Invent 2020：GameLoft - 零停機時間資料湖遷移深入探討

相關訓練：

- 使用適用於遊戲伺服器的 Amazon GameLiftFleetIQ
- 使用 Amazon EC2 託管遊戲伺服器

效能效率

效能效率支柱著重於有效率地使用運算資源以滿足需求，並隨著需求變更和技術的演進而維持該效率。

採取資料驅動型方法來選擇高效能架構。收集架構的完整資料，從高階設計到資源類型的選擇和組態。以週期為基礎考慮您的架構選擇，以利用不斷發展的服務和解決方案集。指標有助於了解偏離預期效能的情況，以便您可以採取行動。資料驅動型方法可協助對您的架構進行權衡，以改善效能、降低成本或改善開發人員體驗。

重點區域

- 設計原則
- 架構選擇
- 區域選擇
- 反覆開發
- 運算與硬體
- 運算選擇
- 資料管理
- 聯網與內容交付
- 程序和文化
- Resources

設計原則

除了 AWS Well-Architected Framework 白皮書中的設計原則之外，下列設計原則可以為您的遊戲實現效能效率：

- 從end-to-end測量遊戲效能：測量效能非常重要，因為從玩家的角度來看。這表示您應該測量遊戲用戶端、遊戲基礎設施以及將玩家連接到基礎設施的網際網路連線的效能。這將有助於了解您在架構中可以在何處進行效能改善。
- 最佳化您的架構以改善反映實際玩家體驗的指標：隨著您隨著時間調整和發展架構，請考慮這些改進和變更將如何影響玩家體驗。遊戲工作負載應該能夠承受並盡量減少失敗的影響，以封鎖對遊戲的廣泛干擾。不嚴重依賴彼此的遊戲功能和系統應解耦，以減少故障的爆量半徑，並隔離影響問題的玩家。

- 使用可簡化遊戲操作並提高開發速度的技術：優先採用可改善開發人員效率的技術。在開發的生產前階段，營運開銷可能會讓您分心，無法改善遊戲體驗。透過利用 AWS 或 AWS 合作夥伴的受管服務，可以減少工程，讓遊戲開發人員專注於核心遊戲迴圈和玩家體驗。架構和效能需求可能會在整個遊戲開發生命週期中變更和發展，而且每個階段都應考慮技術權衡。
- 設計基礎設施以滿足尖峰玩家並行並根據需要動態擴展：基礎設施應設計為擴展以滿足玩家需求。玩家工作階段並行和登入次數等指標可用於在系統過載之前先行擴展。被動系統使用率指標，例如 CPU 和記憶體耗用量，可用於在系統超載之後進行擴展。透過動態擴展您的基礎設施，您可以降低操作遊戲的成本。

架構選擇

GAMEPERF01：如何為您的遊戲伺服器選取適當的託管選項？

為您的遊戲伺服器選擇適當的託管選項是遊戲伺服器效能的基礎。決定使用 EC2 執行個體、容器解決方案或全受管服務，是架構生產時要做出的第一個決定之一。每個託管選項在效能調校、擴展、操作和整合方面都有不同的功能和考量。

最佳實務

- GAMEPERF01-BP01 評估遊戲伺服器資源需求和可擴展性需求
- GAMEPERF01-BP02 考慮擴展遊戲伺服器的操作開銷
- GAMEPERF01-BP03 評估與其他 AWS 服務、開發環境、目標 CPU 架構和功能的整合

GAMEPERF01-BP01 評估遊戲伺服器資源需求和可擴展性需求

根據您的可擴展性評估伺服器需求，以確認您選取的託管選項同時符合您的需求，並提供最佳效能。

未建立此最佳實務時的曝險等級：高

實作指引

為您的遊戲伺服器選取適當的託管選項時，請考慮下列因素：

遊戲伺服器資源需求

評估遊戲伺服器程序的 CPU、記憶體、網路和儲存需求，以判斷您的遊戲使用量。請勿忽略聯網；每個影格都需要 CPU 週期才能接收玩家動作、更新遊戲狀態，並將其傳回給玩家。卸載封包處理可以釋

放核心遊戲函數的 CPU。網路是順暢和回應式遊戲的基礎，因此在程序初期進行測試會定義遊戲的基準效能設定檔。

第一個人射擊遊戲的每秒動作可能很高，CPU 需要快速移出網路，這可能有利於運算最佳化的 C 系列執行個體，而輪換型策略遊戲，可能會花費更多 CPU 週期處理，因此在將遊戲送回玩家之前，可能需要從 R 系列執行個體增加記憶體，才能在本機儲存和更新伺服器上的遊戲狀態。使用資料驅動型方法，例如使用率飽和和錯誤 (USE) 方法，來做出明智的架構選擇。

可擴展性和彈性

評估每個託管選項擴展以滿足玩家需求的速度和流暢程度，而不會犧牲效能。考慮遊戲工作負載所需的自動化和彈性層級，以在尖峰時間維持順暢的遊戲體驗。透過在相同執行個體上新增其他遊戲伺服器程序，遊戲伺服器可能會快速擴展，其中遊戲後端可能會根據不斷增加的作用中使用者計數和正在播放的遊戲而擴展速度變慢。您的機群應隨需求擴展，以將成本降至最低，同時縮短玩家進入遊戲的等待時間。檢閱 Amazon EC2 Spot Instance Advisor，深入了解遊戲伺服器機群的成本效益可用容量。

實作步驟

- 評估 CPU、記憶體、網路和儲存體的遊戲伺服器資源需求，以選擇適合的執行個體類型，並考慮遊戲特定的效能需求，例如 FPS 遊戲的高網路輸送量或輪換策略遊戲的記憶體最佳化。
- 透過使用 USE 方法等架構分析效能資料，比較容器、執行個體、裸機和受管服務等不同的託管選項。使用這些洞見，為您的系統架構做出更好的決策。
- 為可擴展性和彈性設計機群，利用 EC2 Spot Instance Advisor 等工具來最佳化成本，同時促進快速擴展以滿足尖峰時段的玩家需求。

GAMEPERF01-BP02 考慮擴展遊戲伺服器的操作開銷

考慮與每個託管選項相關聯的管理和操作開銷。

未建立此最佳實務時的曝險等級：高

實作指引

營運開銷

EC2 或容器上的自我託管解決方案可以提供更多控制，但也需要更多管理。ECS 或 EKS 等容器協調器可以縮短容器化伺服器的啟動時間，同時增加聯網複雜性和維護協同運作開銷。

例如，EKS 受管節點群組可以自動化遊戲伺服器的佈建和生命週期管理，但在終止節點時不遵守 Pod 中斷預算，如果您的遊戲需要超過 15 分鐘的終止期間才能安全地完成遊戲，您可能需要建立生命週期關聯或考慮使用自訂控制器自行管理的節點來封鎖遊戲中斷。

Amazon Game Lift 等受管服務可能會處理大部分的營運開銷，但減少對低層級聯網和安全組態特殊需求的可見性和控制。選擇遊戲伺服器解決方案是您在調校遊戲伺服器效能和擴展行為時所承擔的自訂、控制和責任層級之間的權衡。

實作步驟

- 評估託管選項的操作開銷、平衡 EC2、ECS 或 EKS 等自我託管解決方案與 Amazon Game Lift 等受管服務之間的控制和管理工作。
- 使用 EKS 受管節點群組進行自動化，但如果您的遊戲伺服器需要比預設值更長的終止期間，則實作生命週期掛鉤或自訂控制器。
- 選取遊戲伺服器解決方案時，權衡自訂、可見性和營運責任之間的權衡。

GAMEPERF01-BP03 評估與其他 AWS 服務、開發環境、目標 CPU 架構和功能的整合

評估每個託管選項與您的遊戲依賴的其他服務 AWS 整合的程度，例如資料庫、分析或內容交付服務。

未建立此最佳實務時的曝險等級：高

實作指引

與其他 AWS 服務整合

服務之間的無縫整合可提供營運優勢，例如改善效能監控，以及遊戲元件、遊戲伺服器、遊戲後端服務和可觀測性解決方案之間的高效率安全資料交付。

例如，協調即時遊戲的流量轉移可能很複雜。Amazon Route 53 將協助將您的 DNS 記錄保持在最新狀態，以簡化協調流量轉移。 AWS Global Accelerator 流量撥號可讓您將一定百分比的流量傳送至另一個區域，並在維護期間保持遊戲執行。

開發環境和工具

請考慮每個架構選項支援的開發工具、架構和環境。驗證您選擇的選項是否符合您的遊戲開發解決方案和程式設計語言，因為這可能會影響您的團隊最佳化和維護遊戲伺服器效能的能力。在行動、主控台和 PC 之間交付遊戲將增加工具和測試的複雜性。跨系統支援對於多遊戲工作室特別重要，其中集中式服務可以標準化跨標題的開發最佳實務。

目標 CPU 架構和功能

考慮遊戲引擎和遊戲伺服器程序的效能設定檔，以及可用的 ARM 支援層級。評估您是否可以受益於 ARM 型 Graviton 或 x86 型 AMD64 處理器的改善價格效能。您需要使用 AES-NI 加密、AVX 或 Turbo Boost 等 Intel 功能嗎？ 檢閱專用主機類型，以識別單一和多通訊端執行個體系列。使用多插槽執行個體系列時，請考慮在遊戲伺服器程序中使用 NUMA 鎖定和 L3 快取共用。使用 C-state 和 P-state 組態，透過調校頻率時鐘和降低休眠層級來取得遊戲的最佳效能。

實作步驟

- 選取與 AWS ACM 等服務無縫整合的託管選項 AWS Secrets Manager，以協助簡化效能監控、安全資料交付，並減少手動操作任務。
- 驗證託管選項與開發環境、架構和程式設計語言之間的相容性，以有效最佳化和維護伺服器效能。
- 評估 CPU 架構需求、將 Graviton 用於價格效能或 x86 用於 AES-NI、AVX 和 Turbo Boost 等特定功能，並使用 NUMA 鎖定和 C-state/P-state 調校來最佳化伺服器效能。

區域選擇

GAMEPERF02：如何決定託管遊戲基礎設施的地理區域？

選擇遊戲基礎設施的理想位置可以改善玩家和後端的網路效能。考慮您的玩家群連接的位置，以及您的社群或伺服器的建置方式，對於地理區域的長期成長和永續性至關重要。部署解耦的遊戲伺服器基礎設施和後端服務，可以透過使用多個區域、本機區域和前哨來託管您的遊戲，讓您的整體營運效率受益並提高彈性。

最佳實務

- GAMEPERF02-BP01 選取靠近玩家的主區域
- GAMEPERF02-BP02 設計一種方法，可支援將延遲敏感的遊戲基礎設施放在玩家附近，以改善效能

GAMEPERF02-BP01 選取靠近玩家的主區域

對於初始遊戲啟動，您應該根據與業務利益相關者的討論來決定部署基礎設施的位置，例如發佈團隊，以判斷遊戲預計提供給玩家的位置，以及他們專注於上市前行銷和廣告工作的位置。

未建立此最佳實務時的曝險等級：高

實作指引

您的業務利益相關者也應該有機制來刺激需求，以便更好地了解玩家的接收和可行性。例如，這些團隊將具有機制，例如遊戲預先訂購、行銷活動和行銷活動、讓玩家在啟動前註冊興趣的公有電子郵件清單，以及其他建立相關訊號的方法，以判斷遊戲在啟動時可能擁有最多玩家的位置。遊戲也可能使用區域推出策略，其中包含遊戲測試和軟啟動階段，以判斷區域玩家的需求。

選取靠近玩家群和開發人員的主要區域，並擁有託管遊戲所需的 AWS 服務和功能。主 RSegion 將是遊戲後端服務將執行的位置，也可能執行遊戲伺服器。根據支援的服務、與節點的連線、與容錯移轉區域的鄰近程度，以及可用區域的數目來評估主要區域。如果您使用的是 Local Zone，請考慮父區域有時位於不同的地理區域。例如：聖地牙哥，智利本地區域 us-east-1-scl-1a 將 N. Virginia us-east-1 作為其父區域，即使其地理位置更接近聖保羅 sa-east-1。

實作步驟

- 根據預先啟動活動的玩家需求訊號來識別部署區域，例如預先訂購、行銷活動和興趣註冊。
- 選擇主要玩家基礎和開發人員附近的主要區域，確保其支援所需的 AWS 服務、節點和容錯移轉區域。
- 仔細評估 Local Zones，考慮到父區域可能與 Local Zone 的位置不同。

GAMEPERF02-BP02 設計一種方法，可支援將延遲敏感的遊戲基礎設施放在玩家附近，以改善效能

遊戲伺服器等延遲敏感基礎設施的個別配置，可將長時間網路路由的影響降至最低。可重複的部署可讓您輕鬆維護多個位置，這些位置對於您的玩家效能更佳。Ping 是在遊戲 UI 中呈現的常見指標，而低 ping 可以是差異化功能。

未建立此最佳實務時的曝險等級：高

實作指引

第一次啟動遊戲時，您可能還沒有關於玩家基礎的足夠資訊，無法充分了解部署最接近最有興趣玩遊戲之玩家的基礎設施的位置。這是常見的挑戰，您應該透過設計可讓您快速調整託管置放策略的架構來部署需要更靠近玩家的伺服器，為此案例做好準備。遊戲開發人員通常會定期評估其遊戲基礎設施部署，做為週期性啟動後分析，以反覆方法逐步投資於隨著時間的改進。

最佳實務是使用infrastructure-as-code範本，例如 AWS CloudFormation 或 Hashicorp 的 Terraform，用於啟動關鍵遊戲服務所需的基礎設施組態，例如 VPCs、子網路組態和相依性，以便您可以參考這些範本，在需要時快速自訂範本，並將其部署到需要額外基礎設施以支援玩家的位置。

您也應該確保了解目前的部署策略如何發展，以允許未來的擴展。IaC 範本是可重複的，但不能取代網路規劃。IPAM 會管理您的 VPCs。子網路大小、可用區域選擇，以及 IP 清查和跨帳戶可用區域對齊。網路很重要，需要考慮，並在變更時對玩家造成干擾。跨多個地理位置部署的遊戲伺服器將連接到您的遊戲後端，這在單一或多個主區域中較常見，這可能需要額外的組態來支援私有連線。隨著時間的推移，應該持續評估這些考量，以便您可以隨著遊戲需求的變化或玩家需求變更，對遊戲託管策略進行變更。

判斷遊戲要使用多少遊戲託管位置時，請考慮下列因素：

- 玩家體驗改善的品質：您可以新增其他遊戲託管位置，藉此提升玩家體驗的程度？ 您可以藉由執行此作業達到什麼增量效能提升？ 您將如何衡量此效能改善？
- 要優先考慮的玩家群體：如果您新增其他遊戲託管位置，可以改善多少玩家的體驗？ 您將優先考慮哪些玩家群體或地理位置？
- 變更的下游影響：如果您變更遊戲託管策略，這將如何影響玩家的配對等待時間？ 玩家集區中的配對大小、技能平衡或玩家數量是否可以適應遊戲託管位置策略變更？ 支援更多位置可能會將玩家集區分段，並增加成本和複雜性。

當您決定新增或移除遊戲託管位置時，應評估這些考量。例如，您可以選擇優先改善遊戲體驗表現最差地理位置的玩家體驗，或是表達最聲音公開意見回饋的玩家體驗。您也可以選擇將玩家獲利納入您的優先順序，例如，專注於改善地理位置中玩家的體驗，這些玩家為您的遊戲產生重大收入來源，或者如果您引進效能提升，則有可能產生增量收入。

除了在 中託管基礎設施之外 AWS 區域，您還可以使用 Local Zones，這是 的延伸，來託管遊戲伺服器和其他延遲敏感的應用程式 AWS 區域，例如更接近玩家的語音聊天伺服器。您也可以選擇在 Local Zones 中執行遊戲開發基礎設施，以改善遊戲開發團隊的體驗。例如，您可以使用 Local Zones 來解決使用案例，例如託管更靠近遊戲開發人員的自我管理來源控制伺服器的複本，以及使用 Amazon EC2 執行個體、EBS 磁碟區和部署到開發工作室附近一或多個 Local Zones 中的 Amazon FSx 檔案系統，為使用者提供遊戲開發虛擬工作站和內容儲存體，而無需您託管內部部署基礎設施。

在相同地理區域無法使用區域或本地區域時，Outpost 是不錯的選擇。 AWS 應考量從資料中心到 的連線能力，讓遊戲伺服器能夠後端系統可靠性。 AWS Outposts 和 Outpost 伺服器專為 AWS 在資料中心使用相同的服務和 APIs，無論您在何處執行遊戲，都能協助建立一致的部署模型。多個機架可以合併為邏輯 Outpost，而且基礎設施可以跨其共用 AWS 帳戶。硬體生命週期由 管理 AWS ，且前置時間最短可達 3 個月。

如果您使用容器建置遊戲，並希望能夠靈活地使用可在您自己的內部部署基礎設施上部署的開放原始碼軟體來採用混合部署架構，您可以使用 ECS Anywhere 或 EKS Anywhere 作為 AWS Outposts 或 Local Zones 的替代方案。如果您使用 Amazon GameLift 託管；Amazon GameLift Anywhere 可用於

在本機硬體上執行伺服器建置，以加速開發程序，讓您能夠使用本機區域或將自己的金屬註冊為機群的一部分。

實作步驟

- 使用infrastructure-as-code工具，例如 AWS CloudFormation 或 Terraform 進行可重複的部署，可根據玩家需求快速自訂和擴展遊戲託管位置。
- 在新增或移除遊戲託管位置時，評估玩家體驗改進、玩家人口優先順序和下游影響，例如配對時間。
- 使用 AWS 本機區域、Outpost 或混合選項，例如 ECS Anywhere、EKS Anywhere 或 GameLift Anywhere，以最佳化對延遲敏感的基礎設施，並支援各種部署需求。

反覆開發

GAMEPERF03：如何使用 Amazon GameLift 提高反覆開發效能效率？

Amazon GameLift end-to-end工作流程，以在本機測試環境中開發和測試遊戲的效能。

最佳實務

- GAMEPERF03-BP01 使用 Amazon GameLift Anywhere 和 GameLift 測試工具組
- GAMEPERF03-BP02 測試遊戲伺服器的效能和可擴展性
- GAMEPERF03-BP03 最佳化 GameLift 容器的資源使用率

GAMEPERF03-BP01 使用 Amazon GameLift Anywhere 和 GameLift 測試工具組

為了透過反覆開發程序提高效能效率，請利用 Amazon GameLift Anywhere 與 Amazon GameLift 測試工具組來建立全面的測試環境。

未建立此最佳實務時的曝險等級：高

實作指引

這種方法允許快速迭代、有效率的資料收集，以及詳細的效能分析。關鍵步驟包括：

建立測試環境

使用 Amazon GameLift Anywhere 設定本機或雲端測試環境。此設定可免除將每個遊戲伺服器建置迭代上傳至受管機群的需求，進而縮短啟用時間。

整合 Amazon GameLift 測試工具組

將 Amazon GameLift 測試工具組納入您的開發工作流程。工具組提供指令碼、工具和程式庫，以視覺化 Amazon GameLift 基礎設施、啟動虛擬玩家，並使用 FlexMatch 模擬器反覆查看 FlexMatch 規則集。它簡化了 Amazon GameLift 資源的整合和管理，可讓您自動化常見任務並收集效能分析所需的資料。

快速建置和測試週期

使用新建置快速更新測試機群、啟動測試機群並開始測試。這有助於快速的build-test-repeat週期，讓開發人員能夠驗證遊戲玩家體驗的各個層面，包括多玩家互動。

全面測試

測試您的遊戲伺服器與 Amazon GameLift 伺服器 SDK、後端服務互動、配對組態和其他 GameLift 託管功能的整合。利用 GameLift 測試工具組來自動化測試並收集詳細的效能指標，確保遊戲元件可無縫搭配運作。

分析效能資料

使用 GameLift 測試工具組收集的資料，分析效能瓶頸並最佳化您的遊戲伺服器。此工具組可協助追蹤關鍵指標、識別問題，並做出資料驅動型決策，以改善效能效率。

透過將 Amazon GameLift Anywhere 和 GameLift 測試工具組整合到您的反覆開發程序中，您可以透過快速測試、全面整合檢查和詳細的效能分析，大幅提升效能效率。

實作步驟

- 使用 Amazon GameLift Anywhere 建立測試環境，減少遊戲伺服器建置的啟用時間，並啟用快速反覆運算。
- 整合 Amazon GameLift 測試工具組，以在開發期間自動化測試任務、模擬玩家和驗證 FlexMatch 組態。
- 使用 GameLift 測試工具組收集和分析效能資料，以識別瓶頸、最佳化遊戲伺服器並提高效能效率。

GAMEPERF03-BP02 測試遊戲伺服器的效能和可擴展性

若要測試遊戲伺服器的效能和可擴展性，請使用 Amazon GameLift 功能和 GameLift 測試工具組實作強大的測試架構。

未建立此最佳實務時的曝險等級：高

實作指引

關鍵實務包括：

反覆測試

使用 Amazon GameLift Anywhere 機群來建立雲端託管環境，您可以在其中反覆建置和測試遊戲元件。此環境應反映實際託管條件，實現逼真的效能和可擴展性測試。

遊戲伺服器整合測試

測試遊戲伺服器與 Amazon GameLift 伺服器 SDK 的整合，包括啟動新的遊戲工作階段，以及使用 AWS CLI 或 GameLift 測試工具組追蹤遊戲工作階段事件。這會驗證遊戲伺服器在 GameLift 環境中是否正常運作。

使用 GameLift 測試工具組來自動化測試並收集詳細的效能指標。此工具組可讓您視覺化 GameLift 基礎設施、啟動虛擬播放器進行負載測試，以及使用 FlexMatch 模擬器在 FlexMatch 規則集上迭代。它特別適用於擴展 ECS Fargate 任務，透過建立多個並行遊戲工作階段來對伺服器基礎設施進行壓力測試來模擬玩家工作階段。

可擴展性測試

試用遊戲工作階段佇列設計、多位置機群、Spot 和隨需機群，以及多種執行個體類型。測試遊戲工作階段置放選項、延遲政策和機群優先順序設定。設定容量擴展以滿足玩家需求，並驗證系統是否可以在不同條件下處理預期的負載。

實作步驟

- 使用 Amazon GameLift Anywhere 設定逼真的測試環境，以進行反覆效能和可擴展性測試。
- 測試遊戲伺服器與 GameLift 伺服器 SDK 的整合，在 GameLift 環境中促進正確的工作階段管理和事件追蹤。
- 使用 GameLift 測試工具組執行可擴展性測試、模擬玩家負載、測試工作階段佇列，以及驗證機群擴展、延遲政策和優先順序設定。

GAMEPERF03-BP03 最佳化 GameLift 容器的資源使用率

若要最佳化 GameLift 容器的資源使用率，請有效設計容器機群並設定精確的資源限制。

未建立此最佳實務時的曝險等級：高

實作指引

關鍵指導方針包括：

- 容器群組設計：將您的軟體組織成容器群組。主要容器應綁定您的遊戲伺服器應用程式和 Amazon GameLift 代理程式。針對其他軟體使用附屬容器來管理相依性，並設定記憶體和 CPU 用量的容器特定限制。
- 設定資源限制：針對每個容器群組，判斷所需的記憶體和 CPU 資源。設定個別容器的選用限制，以確認其具有預留資源，但如果有其他資源可用，也可能超過這些限制。這有助於防止資源爭用和潛在的容器故障。
- 協助程式容器群組：考慮使用協助程式容器群組進行背景或監控程序，這些程序不需要使用主要容器群組進行擴展。這可驗證有效處理基本背景任務，而不會影響主要遊戲伺服器程序。

實作步驟

- 使用遊戲伺服器和 GameLift 代理程式的主要容器來設計容器群組，並使用特定記憶體和 CPU 限制來管理相依性。
- 為每個容器群組設定資源限制，以保留必要的資源，同時允許受控資源使用以避免爭用。
- 使用協助程式容器群組進行背景或監控任務，確保它們可有效率地運作，而不會影響主要遊戲伺服器程序。

運算與硬體

GAMEPERF04：如何阻止遊戲工作階段影響在相同遊戲伺服器執行個體上執行的玩家？

在遊戲伺服器執行後 AWS，您需要監控其效能，以提供高品質的玩家體驗，無論資源使用率、基礎運算或飽和度為何。

最佳實務

- GAMEPERF04-BP01 監控遊戲伺服器程序以偵測問題
- GAMEPERF04-BP02 使用模擬且真實的遊戲案例測試遊戲伺服器的效能

GAMEPERF04-BP01 監控遊戲伺服器程序以偵測問題

您可以為每個執行個體執行多個遊戲伺服器程序，以有效率地利用遊戲伺服器執行個體上的資源。若是如此，請設計您的架構，讓託管遊戲工作階段的個別遊戲伺服器程序不會對同一執行個體上託管的其他遊戲工作階段造成負面影響。使用指標來了解遊戲置放和遊戲模式類型如何影響遊戲伺服器執行個體的效能。結合低負載 (lobby、shop 或單一玩家教學課程） 和高負載 （排名、多玩家或高技能遊戲） 程序的混合，以避免熱點遊戲伺服器執行個體。

未建立此最佳實務時的曝險等級：高

實作指引

透過收集 ping 時間和抖動、影格下降、API 回應時間、錯誤和遊戲迴圈成功完成的遙測，透過用戶端和伺服器端指標監控玩家體驗。將這些事件的時間戳記與玩家支援問題和伺服器日誌建立關聯，以識別效能瓶頸。Dtrace、ftrace、uperf 和 eBPF 等工具可用於深入調查和分析系統效能。

實作對遊戲伺服器執行個體可用有限資源的監控，以便在個別遊戲伺服器程序違反預先確定的資源預算閾值時產生提醒。超過閾值時，您可能想要設定遊戲伺服器軟體傾印相關系統和遊戲伺服器登出耐用的儲存體，例如中央記錄解決方案，以便您的遊戲伺服器工程師可以調查此行為。此外，您的遊戲伺服器執行個體應設定為報告在執行個體上執行的每個遊戲伺服器程序的指標，以便除了遊戲伺服器執行個體的整體指標之外，您還可以監控這些個別遊戲伺服器程序。

例如，GameLift 提供監控遊戲工作階段的指標，這些指標可與使用 Amazon CloudWatch Agent 收集的自訂遊戲特定指標和日誌增強，您可以在遊戲伺服器執行個體上設定這些指標和日誌。您可以在 CloudWatch 中檢視指標，或匯出至與單一登入整合的其他工具，例如 Amazon Managed Grafana，讓無法存取 管理主控台的使用者直接存取指標。請參閱下列使用 Amazon GameLift 管理日誌和指標的最佳實務，這也支援檢視個別遊戲工作階段日誌。

實作步驟

- 每個執行個體執行多個遊戲伺服器程序，並混合低負載和高負載遊戲模式，以避免熱點並驗證平衡的資源使用率。
- 監控用戶端和伺服器端指標，例如 ping、抖動、影格下降和 API 回應時間，並將這些指標與玩家回報的伺服器日誌和問題建立關聯，以識別瓶頸。
- 設定每個遊戲伺服器程序的資源監控、產生閾值違規警示，以及使用 CloudWatch 和 Amazon Managed Grafana 等工具將日誌存放在耐用的儲存體中進行分析。

GAMEPERF04-BP02 使用模擬且真實的遊戲案例測試遊戲伺服器的效能

執行效能測試並評估各種遊戲案例，以判斷遊戲伺服器程序是否適當地處理固定資源的使用率，例如 EC2 執行個體記憶體、CPU 和網路頻寬。

未建立此最佳實務時的曝險等級：高

實作指引

使用可反映玩家常見遊戲路徑和行為的機器人建立模擬遊戲測試，可以判斷您的遊戲伺服器如何處理不同使用案例。例如，您可以實作解決方案，例如 上的分散式負載測試 AWS，您可以自訂以執行遊戲用戶端模擬或遊戲用戶端建置來產生遊戲案例。執行內部遊戲測試，並使用 QA 團隊來對遊戲的各種功能進行壓力測試，以便您可以建立對遊戲設計為最佳執行的信心。 AWS Device Farm可用於在多種裝置類型上執行 iOS、Android 和瀏覽器遊戲的行動和 Web 測試。

實作步驟

- 使用模擬常見玩家行為的機器人執行效能測試，以在不同案例中評估遊戲伺服器資源使用率。
- 使用 上的分散式負載測試等解決方案 AWS 來自訂和模擬壓力測試的遊戲案例。
- 執行內部播放測試，並使用 等工具 AWS Device Farm 在各種裝置上進行行動和瀏覽器遊戲測試。

運算選擇

GAMEPERF05：如何為您的遊戲選取適當的運算解決方案？

運算效能因執行個體大小和系列而異。使用來自不同容量集區的多個運算選項是有益的。制定機群合成策略，以偏好效能，但包含足夠的多樣性，以避免容量不足錯誤。

最佳實務

- GAMEPERF05-BP01 跨多種運算類型的遊戲效能基準
- GAMEPERF05-BP02 non-latency-sensitive的運算任務移至非同步工作流程

GAMEPERF05-BP01 跨多種運算類型的遊戲效能基準

對於遊戲伺服器工作負載，沒有一種方法可以識別託管遊戲伺服器的最佳運算解決方案。對遊戲伺服器進行基準測試的常見策略是從運算最佳化 EC2 'c' 執行個體開始，因為此執行個體系列可為運算密集的

工作負載提供高效能。或者，如果您的遊戲需要大量記憶體來實作特定功能，則記憶體最佳化執行個體可能最適合。

未建立此最佳實務時的曝險等級：高

實作指引

如果您的工作負載使用大量的網路資源，請考慮實作網路最佳化的執行個體，通常在執行個體名稱中使用 'n' 表示，避免爆量執行個體類型 't'，因為在額度用盡之後，效能將會降低。遊戲對延遲和捨棄的封包很敏感，因此建議使用 EC2 增強型聯網來協助改善遊戲伺服器的網路效能。增強型聯網使用單一根 I/O 虛擬化 (SR-IOV)，在支援的執行個體類型 上提供高效能聯網功能。SR-IOV 是一種相較於傳統虛擬網路介面可提高 I/O 效能及降低 CPU 使用率的裝置虛擬化方式。增強聯網提供更高的頻寬、更高的每秒封包數 (PPS) 效能，以及一致較低的執行個體間延遲。使用 Elastic Network Adapter 的增強型聯網適用於最新的 EC2 執行個體類型，並且重要的是定期更新，以從較新的執行個體和 AWS Nitro Hypervisor 的改進中受益於效能增強。

如果您的遊戲在多個 EC2 執行個體類型之間執行類似，則應考慮使用多個執行個體類型來託管您的遊戲伺服器。隨著時間的推移監控效能，並在託管足夠的生產遊戲工作階段後執行進一步的最佳化，以識別效能趨勢。請記住，隨著您在遊戲中新增需要不同資源配置的新功能，您的運算需求可能會變更。您可以設定 EC2 Auto Scaling 群組使用多個執行個體類型，也可以使用個別的 Auto Scaling 群組來託管執行個別執行個體類型的遊戲伺服器執行個體，這可以更輕鬆地管理指標的相互關聯和彙總。

評估遊戲在 Intel 型執行個體、AMD 型執行個體和 ARM 型 Graviton 執行個體等不同類型的處理器上的執行方式。Unreal Engine 5.1.1 或更新版本可以為 Graviton 編譯遊戲伺服器，並提高遊戲的價格效能。在每個系列中執行各種大小的掃描和飽和測試，以確定使用率和效能一致的甜甜地。

您也應該對使用容器和 Lambda 函數託管的遊戲效能有何影響進行基準測試。對於不需要長時間遊戲伺服器程序的使用案例，例如非同步遊戲和遊戲後端服務，請考慮搭配 Lambda 使用無伺服器架構，這可以簡化遊戲操作團隊的管理和操作，並可讓您更快速地將遊戲全域部署到許多 AWS 區域。如需無伺服器最佳實務，請參閱無伺服器應用程式鏡頭 - Well-Architected Framework。

實作步驟

- 針對 CPU 密集型工作負載的運算最佳化 'c' 執行個體、記憶體密集型任務的記憶體最佳化執行個體，以及高網路輸送量的網路最佳化 'n' 執行個體，對遊戲伺服器進行基準測試。
- 在支援的執行個體上使用增強型聯網搭配彈性網路轉接器 (ENA)，以改善網路效能、降低延遲並提高封包處理速率。
- 評估和測試多個執行個體類型、處理器 (Intel、AMD、Graviton) 和容器或 Lambda 託管選項，隨著遊戲功能的發展調整運算解決方案。

如需詳細資訊，請參閱為您的全域遊戲伺服器選擇正確的運算策略。

GAMEPERF05-BP02 non-latency-sensitive的運算任務移至非同步工作流程

當您最佳化遊戲的效能時，請務必記住，只有用戶端和遊戲後端之間的部分互動必須以同步方式執行。您應該從玩家體驗的角度考慮每個功能，並判斷某些互動是否需要同步通訊、封鎖和資源密集，或者這些功能是否可以以非同步方式實作。當您實作網路呼叫時，請使用非同步、非封鎖方法。此外，您的遊戲後端也應設定為透過將任務卸載至佇列，並盡可能優先回應用戶端，以有效率的方式執行工作。

未建立此最佳實務時的曝險等級：高

實作指引

例如，在玩家工作階段結束時更新排行榜可以非同步實作，以便用戶端不需要等待排行榜更新完成。相反地，請在遊戲用戶端上以非同步方式實作此項目，並考慮設計您的後端服務，將這些類型的操作推送至佇列，例如 Amazon SQS。使用此架構，將後端設定為接受請求、在 SQS 中排入佇列，以協助長期儲存訊息以進行非同步處理，並立即回覆用戶端。當排行榜更新完成時，後端可以傳送更新至遊戲用戶端，以便更新玩家的排行榜檢視。

或者，玩家可以直接造訪遊戲的排行榜畫面來擷取最新資料，這會向您的後端發出 Web 請求，以從快取擷取最新資料。

實作步驟

- 判斷用戶端與後端互動是否需要同步通訊；盡可能實作非同步、非封鎖方法，以最佳化資源用量。
- 使用 Amazon SQS 卸載非關鍵任務，例如排行榜更新。
- 允許用戶端以非同步方式擷取更新的資料，例如隨需或透過背景更新擷取最新的排行榜資料。

Resources

- 了解微服務的非同步傳訊
- Lambda - 使用服務整合和非同步處理

資料管理

GAMEPERF06：如何有效率地管理和分析遊戲伺服器日誌，並存放不同類型的遊戲資料以獲得最佳效能？

遊戲可以有玩家資料、遊戲日誌和伺服器日誌，應盡可能彼此分離。集中日誌擷取和生命週期管理可讓您的遊戲團隊受益，方法是深入了解遊戲和伺服器上發生的情況。

最佳實務

- GAMEPERF06-BP01 集中日誌收集和儲存
- GAMEPERF06-BP02 根據存取模式分類和存放遊戲資料
- GAMEPERF06-BP03 啟用有效的日誌格式和批次處理
- GAMEPERF06-BP04 實作日誌輪換和保留政策
- GAMEPERF06-BP05 使用監控和視覺化工具

GAMEPERF06-BP01 集中日誌收集和儲存

實作集中式日誌收集和儲存解決方案，從遊戲伺服器執行個體和 GameLift 收集日誌。

未建立此最佳實務時的曝險等級：高

實作指引

使用 Amazon CloudWatch Logs 之類的服務，從遊戲伺服器和 GameLift 執行個體收集、監控和儲存日誌資料。CloudWatch Logs 提供可擴展且全受管的日誌管理解決方案，有助於有效率地儲存和擷取日誌資料，而不會影響遊戲伺服器效能。如果您執行 CloudWatch Logs 代理程式，請考慮各種安裝類型和組態選項，例如批次大小、緩衝持續時間，以將對遊戲伺服器的影響降至最低。考慮遊戲伺服器執行個體暫時性，並盡可能減少對當地語系化記錄的相依性。建立集中式政策以實作記錄最佳實務。

實作步驟

- 使用 Amazon CloudWatch Logs 從遊戲伺服器執行個體和 GameLift 收集、監控和存放日誌資料，促進集中且可擴展的日誌管理。

GAMEPERF06-BP02 根據存取模式分類和存放遊戲資料

根據遊戲資料的存取模式和儲存需求，將遊戲資料分類為不同的類型。

未建立此最佳實務時的曝險等級：高

實作指引

常見類別包括玩家資料、遊戲儲存、持久性世界儲存和分析資料。

實作步驟

為每個資料類型使用適當的儲存解決方案，以最佳化效能和成本效益：

- 玩家資料：使用快速且可擴展的 NoSQL 資料庫 Amazon DynamoDB 來存放玩家設定檔、偏好設定和進度資料。DynamoDB 的低延遲存取和自動擴展功能可有效擷取和更新玩家資料。
- 遊戲儲存：使用 Amazon S3 存放遊戲儲存和檢查點。S3 為儲存大量遊戲儲存資料提供了高耐用性和可擴展性。請考慮使用 S3 Transfer Acceleration 或 Amazon CloudFront，以更快的速度上傳和下載遊戲儲存。
- 持久性世界儲存：對於具有持久性世界狀態或共用遊戲資料的遊戲，請考慮使用 Amazon DynamoDB、Amazon ElastiCache 或 Amazon MemoryDB。ElastiCache 和 MemoryDB 提供記憶體內鍵值存放區，而 DynamoDB 是 SSD 支援的 NoSQL 資料庫。這些服務可快速存取儲存的資料，減少遊戲伺服器程序儲存遊戲狀態所需的時間，進而改善整體程序效能。
- 分析資料：使用 Amazon Managed Streaming for Apache Kafka 或 Kinesis Data Streams 從您的遊戲資料生產者擷取資料串流。Amazon Managed Service for Apache Flink 可用於即時轉換和分析，並傳送至 Amazon Data Firehose 以處理和交付至後端資料湖、倉儲和分析服務。上的 Game Analytics Pipeline 指引 AWS說明服務如何協同運作，以提供近乎即時的批次分析。

GAMEPERF06-BP03 啟用有效的日誌格式和批次處理

設定遊戲伺服器程序，以結構化和可剖析的格式產生日誌，例如 JSON。

未建立此最佳實務時的曝險等級：高

實作指引

實作日誌批次處理技術，將日誌資料從遊戲伺服器傳輸到集中式日誌儲存的頻率降至最低。批次處理日誌可減少網路額外負荷，並改善遊戲伺服器效能。使用詳細或偵錯層級日誌作為例外狀況，而不是預設值，因為它們可能會產生效能和成本損失，應盡可能避免。

GAMEPERF06-BP04 實作日誌輪換和保留政策

建立日誌輪換和保留政策，以管理日誌資料的成長並最佳化儲存使用率。

未建立此最佳實務時的曝險等級：低

實作指引

設定您的遊戲伺服器，根據大小或時間間隔自動輪換日誌。在 Amazon CloudWatch Logs 中定義日誌保留政策，以自動封存或刪除作用中分析或疑難排解不再需要的舊日誌資料。

GAMEPERF06-BP05 使用監控和視覺化工具

使用監控和視覺化工具來深入了解遊戲伺服器效能，並識別最佳化機會。

未建立此最佳實務時的曝險等級：高

實作指引

使用 Amazon CloudWatch 監控關鍵指標並設定主動通知警示。利用 Amazon Managed Service for Prometheus 和 Amazon Managed Grafana 等工具，從您的遊戲伺服器和基礎設施收集、查詢和視覺化指標。建立資訊豐富的儀表板來追蹤效能、找出瓶頸，以及進行資料驅動最佳化。

聯網與內容交付

GAMEPERF07：如何設計配對服務以最佳化效能？

玩家技能、網際網路服務提供者 (ISP) 品質和玩家人口分佈，是效能調校的維度。遊戲工作階段可以放置在具有策略位置的伺服器上，以水平遊戲欄位並託管公平遊戲。

最佳實務

- GAMEPERF07-BP01 定義遊戲的網路延遲閾值
- GAMEPERF07-BP02 為每個遊戲模式和遊戲託管區域執行單獨的配對服務
- GAMEPERF07-BP03 定期監控配對效能
- GAMEPERF07-BP04 定期監控網路效能
- GAMEPERF07-BP05 使用網路加速技術來改善網際網路的效能

GAMEPERF07-BP01 定義遊戲的網路延遲閾值

開發多玩家遊戲時，請確認您的遊戲基礎設施不會為玩家帶來不必要的延遲。如果您的遊戲對網路延遲敏感，則應在配對邏輯中設定延遲閾值，以優先將玩家放置在託管在附近遊戲伺服器位置 AWS 區域或符合您理想玩家體驗目標的遊戲伺服器工作階段上。

未建立此最佳實務時的曝險等級：高

實作指引

在許多延遲敏感的遊戲中，通常會檢測遊戲用戶端以 ping 每個遊戲的基礎設施位置，以收集效能資料，例如網路延遲、抖動和封包遺失，並將此資料報告給指標收集後端，以便進行分析。將玩家配對至遊戲工作階段時，您可以將遊戲設定為將遊戲用戶端感知的網路延遲納入遊戲伺服器基礎設施，做為配對服務邏輯中使用的其中一個輸入。

GAMEPERF07-BP02 為每個遊戲模式和遊戲託管區域執行單獨的配對服務

如果您的遊戲提供多種遊戲模式供玩家選擇，您應該為每個遊戲提供不同的配對系統，以便您可以根據其唯一需求獨立調整每個遊戲模式的效能，並減少資源爭用。每個遊戲模式對於可接受的延遲、配對大小，以及其他自訂遊戲特定的配對邏輯，可能會有獨特的需求。他們也可能吸引不同類型的玩家。將每個遊戲模式的配對服務作為單獨的軟體部署執行，以便您可以獨立測試和操作遊戲模式。

未建立此最佳實務時的曝險等級：高

實作指引

例如，您可以為每個遊戲模式執行這些作為單獨的 Lambda 函數，也可以將其作為單獨的容器型服務部署操作。

將配對服務部署到遊戲伺服器位置附近的多個區域。玩家流量需要許多路由，因此配對服務必須跨多個 ISPs up-to-date延遲描述檔，以改善低延遲遊戲工作階段放置的效率。GameLift FlexMatch 提供為配對建構器選取區域的其他指導，並包含整合配對建構器與多區域遊戲工作階段佇列的能力。

GAMEPERF07-BP03 定期監控配對效能

最佳化玩家遊戲效能最明顯的方式之一，就是縮短他們進入遊戲工作階段之前必須等待的時間。長時間等待可能會導致玩家失去興趣並導致流失，因此在設計配對解決方案時請務必考慮這一點。

未建立此最佳實務時的曝險等級：高

實作指引

當您為遊戲設計配對組態時，請建立規則來判斷套用條件以形成配對。您應該考慮這些規則對系統效能的影響，特別是玩家的等待時間。部署配對實作的變更之前，例如新增配對條件或篩選條件，請事先測試此變更，或考慮將此變更逐步發佈給小型範例玩家，做為 Canary 或 A/B 測試，以收集效能指標，然後再將此變更引入整個玩家群體。

設定您的配對服務來產生詳細日誌，以了解套用至每個配對請求的條件或規則。這有助於審核，並視需要調整配對實作。

例如，Amazon GameLift FlexMatch 提供全受管配對服務，可做為獨立服務與您自己的遊戲伺服器託管，或與 Amazon GameLift 上託管的遊戲伺服器搭配使用。FlexMatch 可以產生 Amazon EventBridge 的事件通知，請參閱設定 FlexMatch 事件通知。使用 Amazon Simple Notification Service (Amazon SNS) 以 JSON 格式接收配對資料，可讓您自動處理和儲存此資訊以供分析，以改善配對效能。

設定指標以追蹤配對服務為玩家尋找適合的遊戲工作階段所需的時間。定期檢閱配對持續時間指標，並將這些時間與玩家行為和社群情緒建立關聯。使用此資料來開發適當的配對逾時閾值，這些閾值可以包含在配對規則組態中。

例如，Amazon GameLift FlexMatch 支援定義配對請求逾時，以及建立配對規則，以允許一段時間內的需求放寬。此功能可讓您建立配對，以直接調整以建立配對，並在難以找到配對時將玩家放入遊戲工作階段。

GAMEPERF07-BP04 定期監控網路效能

對於競爭性遊戲，請務必擁有一致的玩家體驗。

未建立此最佳實務時的曝險等級：高

實作指引

對較大的玩家群來說，可靠 50 毫秒的遊戲比一對玩家具有 10 毫秒 ping，另一對玩家具有 70 毫秒 ping，更公平、更有趣。ISP 路由變更可能會影響部分玩家人口，您的配對系統將需要調整。Amazon CloudWatch Network Monitoring 可協助判斷問題是與您的遊戲還是玩家網際網路供應商有關。

實作步驟

- 使用 Amazon Cloudwatch 網路監控來追蹤網路效能並識別路由問題。
- 使用 VPC 流量日誌來識別異常流量模式或捨棄的封包，這可能表示網路擁塞、ISP 問題或會影響玩家延遲的錯誤設定。

GAMEPERF07-BP05 使用網路加速技術來改善網際網路的效能

除了將對延遲敏感的遊戲基礎設施實際放置在更接近玩家的位置之外，您還可以透過最佳化遊戲的網路效能來改善玩家體驗。 AWS 使用 BGP 通訊協定來影響網際網路路由，以使用從網際網路服務供應商通往邊界網路的最快路徑。如果您操作自己的網路，並需要更多對路由行為和 BGP 公告的控制和可觀測性，您可以使用私有對等互連或 ， Direct Connect 將流量從網際網路路由到執行中的遊戲 AWS。

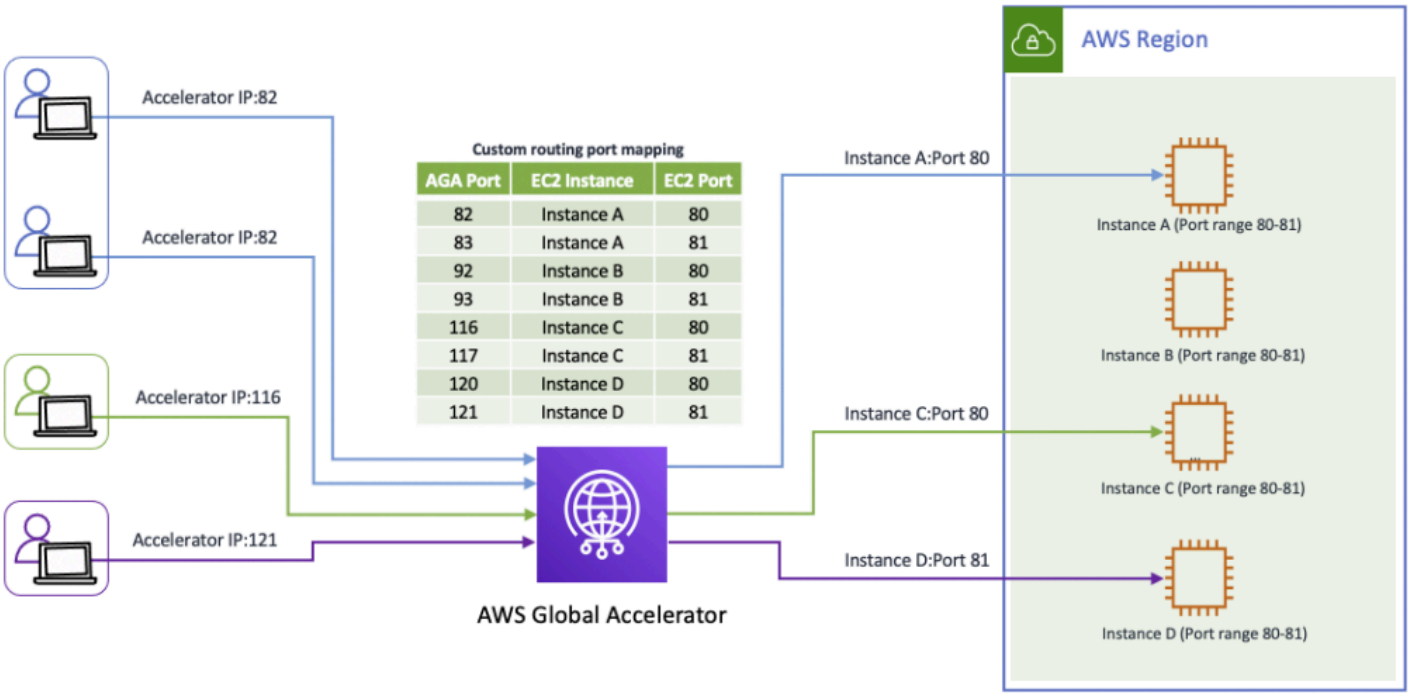
未建立此最佳實務時的曝險等級：高

實作指引

請考慮下列參考架構，以支援改善的網際網路效能和回應能力。

使用 Global Accelerator 增強遊戲的網路效能

對於全受管的網路路由解決方案，AWS Global Accelerator 使用 AWS 全域網路改善應用程式的網路效能，可用於加速遊戲流量、語音聊天和即時簡訊流量，以及其他延遲敏感的應用程式，同時提供快速容錯移轉至遊戲伺服器。Global Accelerator 自訂路由加速器可以與您的配對服務整合，以便使用靜態廣播 IP 地址和連接埠，將多個玩家的確定性路由傳送至相同的遊戲工作階段。



您的遊戲開發團隊可能分佈於世界各地，需要高效能存取共用內容或資產。若要改善存放在 Amazon S3 儲存貯體中共用內容的效能，您可以使用 S3 跨區域複寫跨區域設定資料的雙向複寫，以便使用者可以從更接近它們的儲存貯體存取資料。若要簡化此存取模式，請使用 S3 多區域存取點，透過全球加速器加速對 S3 的請求。

如需詳細資訊，請參閱利用 AWS Global Accelerator 和 Amazon GameLift FleetIQ 改善玩家體驗。

實作步驟

- 使用 AWS Global Accelerator 協助改善遊戲流量、語音聊天和即時訊息的網路效能，同時促進快速容錯移轉至遊戲伺服器。

- 設定 Global Accelerator 自訂路由加速器以與您的配對服務整合，使用靜態任意投射 IPs 讓玩家決定性路由至遊戲工作階段。
- 啟用 S3 跨區域複寫，為分散式遊戲開發團隊跨區域複寫共用內容。
- 使用 S3 多區域存取點，為全球分散式使用者加速透過 AWS 全球網路存取 S3 資料。

程序和文化

GAMEPERF08：如何使遊戲的效能列符合玩家和開發人員的期望？

了解您的玩家和開發人員是提升效能效率最重要的層面之一。提供低營運開銷的高效能遊戲，是向玩家和開發人員展示您關心其體驗的最佳方式之一，並且可以讓您的遊戲和工作室脫穎而出。

最佳實務

- GAMEPERF08-BP01 在您的程序中通知並包含玩家
- GAMEPERF08-BP02 將解決方案選擇與工程團隊的技能和專業知識保持一致

GAMEPERF08-BP01 在您的程序中通知並包含玩家

提供在遊戲指標中顯示的選項，例如延遲、每秒影格數和捨棄封包。透過玩家面對狀態頁面等通訊來解決基礎設施問題和維護停機時間。使用玩家通訊慶祝新的遊戲位置，包括開發部落格，並設定預期玩家體驗改善的期望。

未建立此最佳實務時的曝險等級：高

實作指引

包含玩家

提供簡單的診斷提交程序，收集相關檔案並將其連接至來自遊戲用戶端的玩家支援票證。啟用支援論壇，讓玩家可以互相協助，並成為改善遊戲體驗的一部分

考慮權衡與玩家期望的比較

玩家可能無法注意到移動後端系統以實現成本效益，但移動遊戲伺服器可能會變更 ping 時間。對具有擴展和減少遊戲託管位置推理的玩家一致且公平。

玩家社群和地理將具有自己的特性，這可能會影響對遊戲的期望。例如，韓國有一些全球最快的網際網路，對遊戲體驗的預期是單一位數延遲，可推動高度競爭的遊戲體驗。行動裝置上的臨時遊戲體驗會建立不同的效能設定檔，並使用與主控台和 PC 工作階段播放相比的模式。

登入和大廳是體驗的一部分，即使伺服器離線進行維護，也應該感到回應。在大廳進行突如其來的夜間規劃或閒暇是玩家體驗的一部分，在選擇重點區域以提高效能時，請務必考慮這一點。玩家可能會讓您的遊戲用戶端保持開啟數月，有時可能只是偶爾登入以讀取修補程式備註。作為工程程序和文化的一部分，即時操作遊戲需要記住整個玩家體驗。

實作步驟

- 提供遊戲內指標，例如延遲、FPS 和封包遺失，並透過狀態頁面和面向玩家的更新來傳達基礎設施問題和維護排程。
- 在遊戲用戶端中實作診斷傾印和提交功能，並建立支援論壇，以促進社群驅動的疑難排解和改善。
- 將效能最佳化調整為玩家社群期望，例如競爭區域的低延遲，或休閒和長工作階段玩家的回應式登入/lobby 體驗。
- 設計即時操作工作流程，以考量從主動遊戲到閒置用戶端行為的整個玩家體驗，促進無縫參與。

GAMEPERF08-BP02 將解決方案選擇與工程團隊的技能和專業知識保持一致

在選擇託管選項時，評估您的團隊在管理和最佳化遊戲伺服器效能方面的技能和專業知識。EC2 和容器等自我託管解決方案需要更多基礎設施管理、效能調校和擴展的知識。如果您的團隊缺乏這些技能，GameLift 等受管服務可能更適合，因為它可消除許多複雜性，並讓您的團隊專注於遊戲特定的最佳化。

未建立此最佳實務時的曝險等級：高

實作指引

透過評估這些因素並跨不同的託管選項執行效能測試，您可以選取最符合遊戲特定需求的解決方案，同時最佳化效能效率。

Resources

進一步了解與效能效率相關的最佳實務。

相關文件：

- AWS 架構中心
- 效能效率支柱 – AWS Well-Architected Framework
- 使用儲存服務比較您的內部部署 AWS 儲存模式
- EC2 執行個體的執行個體存放區暫存區塊儲存
- 使用 CloudWatch 代理程式收集指標、日誌和追蹤
- CloudWatch 代理程式
- 如何在 EC2 執行個體上開啟和設定增強型聯網？
- 透過利用 AWS Global Accelerator 和 Amazon GameLift FleetIQ 改善玩家體驗
- Riot Games Technology 部落格：Valorant 的可擴展性和負載測試
- 使用混合 AWS 解決方案進行超大規模線上遊戲
- 使用率飽和和錯誤 (USE) 方法
- Amazon EC2 Spot 執行個體
- Amazon ElastiCache 的大規模效能
- 使用 Redis 的資料庫快取策略
- Amazon Virtual Private Cloud 連線選項
- 最佳實務設計模式：最佳化 Amazon S3 效能

相關基準：

- Amazon EBS 磁碟區基準
- Amazon EC@ 執行個體網路頻寬

相關工具：

- Unreal Engine：測試和最佳化您的內容
- Unity Profiler
- 開啟 3D 引擎 (O3DE) 品質系統
- 監控 Amazon GameLift 伺服器
- Amazon GameLift 測試工具組

相關影片：

- AWS re：Invent 2019：【REPEAT 2】 Amazon EC2 基礎 (CMP211-R2)
- AWS re:Invent 2019：為新一代 Amazon EC2 提供支援：深入探索 Nitro 系統 (CMP303-R2)
- Amazon GameLift FleetIQ – AWS 線上技術講座入門
- 使用 AWS 改善遊戲的暴動遊戲 Zach Blitz
- AWS re：Invent 2023 - AWS Graviton： AWS 工作負載的最佳價格效能 (CMP334)

相關訓練：

- 上的遊戲伺服器託管 AWS
- 使用適用於遊戲伺服器的 Amazon GameLiftFleetIQ
- AWS 遊戲入門 – 第 I 部分
- 上的遊戲伺服器託管 AWS
- AWS re：Invent 2023 – AWS Graviton： AWS 工作負載的最佳價格效能 (CMP334)

成本最佳化

成本最佳化支柱包括在整個生命週期內精簡和改進系統的持續程序。此程序的範圍從您第一個概念驗證的初始設計到生產工作負載的持續操作。透過採用本文概述的實務，您可以建置和操作成本感知系統，以最低的價格達到所需的業務成果。實作這些成本最佳化實務可讓您的企業將雲端投資的價值最大化。

遊戲是獨特的創意專案，必須爭奪玩家的注意力和播放時間。在啟動之前，遊戲開發人員通常無法清楚了解其遊戲的熱門程度或持久程度。根據遊戲的獲利策略、業務優先順序和生命週期階段，開發人員在評估成本最佳化決策時需要做出權衡。

例如，在高預期新遊戲的上市前階段，重點通常是speed-to-market、功能開發和效能。優先順序是驗證基礎設施可以擴展以滿足尖峰玩家需求。相反地，如果遊戲不成功或開發速度變慢，則焦點可能會轉移到盡可能降低成本，以繼續為現有玩家操作遊戲。

許多遊戲開發人員也會同時操作多個遊戲，這需要額外的考量。基礎設施、軟體和員工等資源可能會在多個即時遊戲之間共用，允許一個遊戲的損失被另一個遊戲的利潤抵銷。在此案例中，專注於成本最佳化可以改善整個遊戲產品組合中的財務。

鑑於遊戲的獨特商業模式、規模和不可預測性，下列關鍵問題可以引導成本最佳化決策：

- 如何測量每個玩家、系統和遊戲功能的基礎設施成本？
- 對於我遊戲目前的生命週期階段，成本最佳化和玩家體驗之間的適當平衡是什麼？
- 如何使用適當的 AWS 資源定價模式來最大化投資報酬率？

套用這些最佳實務並提出正確的問題，可協助遊戲開發人員建置和操作成本感知系統，以實現業務成果，同時將成本降至最低。

重點區域

- 設計原則
- 實作雲端財務管理
- 了解支出和用量
- 具有經濟效益的資源
- 資料傳輸費用
- 管理需求和供應資源
- 隨著時間進行最佳化
- Resources

設計原則

除了來自 Well-Architected Framework 成本最佳化支柱的設計原則之外，下列設計原則也會最佳化在雲端中執行遊戲工作負載的成本。

- 測量每個玩家、系統和遊戲功能的基礎設施成本：了解和追蹤遊戲系統中特定玩家體驗和功能所需的基礎設施成本。這可以識別架構中可能需要成本最佳化的區域。
- 評估成本最佳化與玩家體驗的權衡：評估遊戲處於哪個階段，以確定正確的焦點：玩家體驗或成本最佳化。一般而言，一旦遊戲達到關鍵質量且玩家群體穩定，就可以專注於最佳化營運成本。關鍵在於平衡提供絕佳的玩家體驗，以最具成本效益的方式執行遊戲基礎設施。實作這些設計原則可將遊戲投資的回報最大化。

實作雲端財務管理

Games Lens 沒有特定的雲端財務管理最佳實務。如需雲端財務管理的指引，請參閱 Well-Architected Framework Cost Optimization Pillar。

了解支出和用量

GAMECOST01：如何衡量遊戲環境的成本？

了解每個玩家、遊戲功能和環境的成本，以便隨著玩家數量隨著時間的推移而變更和功能新增和改善，管理和預測您的花費。請考慮下列最佳實務，以管理不同遊戲環境的成本。

最佳實務

- GAMECOST01-BP01 實作每個玩家、遊戲功能和環境的成本歸因
- GAMECOST01-BP02 探索最佳化的機會

GAMECOST01-BP01 實作每個玩家、遊戲功能和環境的成本歸因

遊戲伺服器的成本歸因通常比遊戲後端服務更容易執行，因為遊戲伺服器通常經過最佳化，能夠在每個執行個體託管特定數量的並行玩家，這些玩家可以分配到執行執行個體的成本。

未建立此最佳實務時的曝險等級：高

實作指引

對於遊戲後端服務，建議將遊戲的元件解耦成不同的功能，這些功能可以作為單獨的邏輯或實體資源進行管理，使其直接分析成本。

例如，雖然實作單一單體應用程式來託管遊戲後端服務看起來很簡單，但此模式可讓您在新增更多功能時，難以衍生每個玩家和遊戲功能的總成本，因為資源的運算、聯網和儲存成本會跨功能共用。考慮使用 Amazon API Gateway 和 AWS Lambda 或 AWS Fargate 等服務、Amazon SQS 和 Amazon SNS 用於傳訊、Amazon S3 用於物件儲存，以及 Amazon DynamoDB 用於資料庫儲存等服務，為您的遊戲後端服務採用無伺服器架構。這些服務只是一些產品範例，提供以用量為基礎的定價，主要由請求量驅動，因此成本可以精細地視覺化。Lambda 函數、Fargate 服務、DynamoDB 資料表和 S3 儲存貯體等個別資源可以與成本分配標籤相關聯，因此您可以使用遊戲功能名稱來歸納這些服務的成本，讓您直接了解每個服務的成本。

也建議您個別管理每個遊戲開發環境，以便您可以歸納不同環境的成本。一般而言，遊戲開發人員會管理開發、測試、預備和生產環境的個別環境，如此遊戲產業鏡頭的操作支柱所述。每個環境通常都有不同的可擴展性、效能和用量需求，並且可能由不同的團隊管理。若要控制成本，請組織這些環境，以便您可以正確監控和歸納每個環境的成本。

如需詳細資訊，請參閱下列文件：

- 建置可擴展的無伺服器多玩家遊戲
- 具有 WebSockets 型後端的獨立遊戲工作階段伺服器
- 具有無伺服器後端的獨立遊戲工作階段伺服器

實作步驟

- 使用 Amazon API Gateway 等無伺服器或容器化架構 AWS Fargate ，將遊戲後端服務解耦為不同的功能 AWS Lambda，並啟用每個功能的精細成本歸因。
- 將成本分配標籤套用至個別資源 （例如 Lambda 函數、DynamoDB 資料表和 S3 儲存貯體），將成本與特定遊戲功能建立關聯，以便進行更好的成本分析。
- 管理個別環境以進行開發、測試、預備和生產，獨立組織和監控其成本，以符合可擴展性和用量需求。

GAMECOST01-BP02 探索最佳化的機會

遊戲開發人員和發佈者可以使用 AWS FinOps 實務來協助最佳化雲端成本，並更清楚地了解其雲端支出。透過這樣做，遊戲生產者可以將維護玩家基礎設施所需的平均成本與遊戲交付的財務結果保持一致。

未建立此最佳實務時的曝險等級：低

實作指引

AWS 為 Cloud Financial Management 提供隨時可用的解決方案指引，以管理和最佳化雲端服務的費用。此功能包括精細的可見性和成本和用量分析，以支援對支出儀表板、最佳化、支出限制、退款以及異常偵測和回應等主題的決策。Cloud Financial Management 的解決方案指引包含預算和預測功能，為您的工作負載提供已定義的成本最佳化架構，讓您可以選擇正確的定價模型，以及與團隊相關的屬性資源成本。這可啟用整個環境和資源的追蹤、通知和成本最佳化技術。您可以集中管理費用資訊，並視需要為關鍵利益相關者提供存取權，以實現目標可見性和支援決策。

另一個關鍵 FinOps 工具是 Cost Optimization Hub，它提供集中檢視整個 和 的成本最佳化建議 AWS 帳戶 和機會 AWS 區域，讓您可以充分利用 AWS 支出。您可以使用 Cost Optimization Hub 來識別、篩選和彙總整個 AWS 帳戶 和 AWS 的成本最佳化建議 AWS 區域。它會針對資源權利調整、閒置資源刪除、Savings Plans 和預留執行個體提出建議。使用單一儀表板，您不必前往多個 AWS 產品來識別成本最佳化機會。

如果您的遊戲團隊使用共用 AWS 帳戶 的 myApplications in AWS 管理主控台 Home， 可用來檢視個別工作負載的應用程式資源成本。此精細檢視可讓您識別遊戲基礎設施中的特定成本趨勢，讓您做出有關資源配置和最佳化的明智決策。

此外，使用 AWS Data Exports 定期檢閱您的帳單和成本管理資料，發現隱藏的成本節省機會。此詳細報告提供雲端支出的完整明細，可讓您識別過度支出、未充分利用的資源，以及利用更具成本效益的服務或定價模型的機會。

透過採用 FinOps 原則並利用 提供的工具 AWS，遊戲開發人員和發佈者可以最有效地利用其雲端資源，最終提高其底線並釋出資金以進一步開發遊戲和創新。

實作步驟

- 使用 AWS 雲端 財務管理工具來精細且詳細的可見性、支出儀表板、異常偵測和成本歸因，以有效地最佳化和追蹤雲端費用。
- 使用 Cost Optimization Hub 集中 和 區域之間的權利化、Savings Plans AWS 帳戶 和預留執行個體建議。

- 在 上使用資料匯出和 MyApplication 定期檢閱 AWS 帳單資料 AWS ，以協助分析工作負載特定的成本、發現節省成本的機會，以及最佳化資源配置。

具有經濟效益的資源

GAMECOST02：如何為您的遊戲伺服器選擇正確的運算解決方案？

與其他類型的工作負載相比，遊戲工作負載最獨特的層面之一是遊戲伺服器。遊戲伺服器對於玩家體驗至關重要，因為玩家會從遊戲用戶端連接到遊戲伺服器，以玩遊戲工作階段。

遊戲伺服器也是操作多玩家遊戲成本的最大驅動因素之一。因此，最佳化遊戲伺服器的運算基礎設施以降低成本的方式非常重要。

最佳實務

- GAMECOST02-BP01 最佳化跨網際網路的資料傳輸成本
- GAMECOST02-BP02 最佳化在每個遊戲伺服器執行個體上託管的遊戲工作階段數量，以最佳化成本
- GAMECOST02-BP03 選取適當的運算定價選項以降低成本

GAMECOST02-BP01 最佳化跨網際網路的資料傳輸成本

雖然 AWS 主要收取從 AWS 資源到網際網路的傳出（傳出）資料傳輸費用，但遊戲公司可能會面臨與透過 AWS Direct Connect 或 AWS Gateway 負載平衡器進行資料傳輸相關的高成本，這可能會同時收取傳入（傳入）和傳出資料的費用。實作解決方案，降低將資料從遊戲 AWS 後端傳輸至玩家的整體成本，專注於將 AWS 資源的輸出費用降至最低，以及評估透過 AWS 連線服務管理輸入和輸出費用的選項。

未建立此最佳實務時的曝險等級：高

實作指引

使用 Amazon CloudFront 降低內容交付和公開 Web 應用程式的成本。

存放在雲端的遊戲內容和資產通常存放在 Amazon S3 中，並直接從 S3 或從 Amazon EC2 託管的 Web 伺服器交付至遊戲用戶端，該伺服器會從 Amazon S3 擷取內容並將其交付至用戶端。若要降低內容下載的資料傳輸成本，請考慮在雲端儲存之前使用 Amazon CloudFront 將內容交付給使用者。

使用 CloudFront 可以降低資料傳輸的成本，因為從 CloudFront points-of-presence交付內容的成本低於直接從 區域交付內容的成本，而且 CloudFront 不會針對 AWS Amazon EC2 和 Amazon S3 等原始伺服器收取原始伺服器擷取費用。如果您的內容為靜態且不常變更，您可以使用 CloudFront 將資料快取到更接近最終使用者的位置，進而進一步降低成本。

CloudFront 也改善了面向前端的公開 Web 應用程式和服務的成本效率，即使未使用快取，因為透過 AWS 網路路由流量可以降低伺服器和用戶端之間的資料傳輸成本。

Amazon CloudWatch 可用於監控您的 Amazon CloudFront 用量。對於使用多個內容交付網路 (CDNs) 的使用案例，Amazon CloudFront Origin Shield 可以提供額外的快取層，以合併並減少來自不同供應商的原始伺服器請求數量。

若要了解您的遊戲網路流量，您可以啟用 VPC 流程日誌和 Amazon CloudWatch 網路監視器，讓玩家或遊戲後端連線具有end-to-end可見性。該方法可以識別高資料傳輸成本的原因，並執行架構變更以最佳化資料傳輸支出。

實作步驟

- 在 Amazon S3 或 EC2 型內容原始伺服器前使用 Amazon CloudFront，透過利用 CloudFront points-of-presence的低成本交付並移除原始伺服器擷取費用，來降低資料傳輸成本。 Amazon S3 EC2-based
- 啟用 VPC 流程日誌和 Amazon CloudWatch 網路監視器，以分析網路流量並識別架構變更，以最佳化資料傳輸成本。
- 實作 CloudFront Origin Shield，以便在使用多個 CDNs 以提高成本效益時合併和減少原始伺服器請求。

如需內容交付的更多最佳實務，請參閱遊戲的內容交付白皮書。

GAMECOST02-BP02 最佳化在每個遊戲伺服器執行個體上託管的遊戲工作階段數量，以最佳化成本

最佳化每個伺服器執行個體託管的遊戲工作階段數量，以達到更佳的運算使用率並降低運算基礎設施成本。

未建立此最佳實務時的曝險等級：中

實作指引

為了最佳化成本，遊戲開發人員應最大化在相同實體或虛擬伺服器上託管的遊戲工作階段數量，也稱為遊戲伺服器的封裝密度。這是透過增加可同時託管在執行個體上的遊戲伺服器程序數目來實現的。

單一遊戲伺服器程序通常不應要求使用 EC2 執行個體上可用的整個資源。這是降低遊戲運算成本的最重要方式之一，需要使用可在個別連接埠上產生和管理 EC2 執行個體上多個伺服器程序的軟體。

例如，Amazon GameLift 對每個執行個體的遊戲伺服器程序數量上限有配額，您應該努力利用這些配額，以便降低託管成本。如需詳細資訊，請參閱 Amazon GameLift Servers 端點和配額，以取得每個執行個體最大遊戲伺服器程序目前配額的詳細資訊。

作為在 EC2 執行個體等虛擬機器上部署遊戲伺服器程序的替代方案，遊戲開發人員使用容器協同運作解決方案，將其遊戲伺服器作為容器型應用程式執行變得越來越熱門。遊戲開發人員可以使用 Amazon Elastic Container Service (Amazon ECS) 或在 Amazon EKS 上使用 Agones 和 Open Match 的遊戲伺服器託管指南。另一個選項是 Game Server Hosting on AWS Fargate，這是一個可與 ECS 和 EKS 搭配使用的無伺服器運算引擎，可讓您專注於遊戲，而無需管理基礎基礎設施。

容器解決方案提供任務排程功能，可根據您指定的資源需求和其他置放邏輯，在叢集中自動尋找可用的容器執行個體來託管遊戲伺服器容器。不過，請務必考慮如何以不中斷作用中玩家工作階段的方式管理擴展和玩家放置行為。

實作步驟

- 使用不同的連接埠和程序管理軟體，為每個 EC2 執行個體執行多個遊戲伺服器程序，以增加封裝密度。
- 使用 Amazon GameLift 或容器解決方案，例如 ECS、EKS 或 AWS Fargate ，有效率地管理遊戲伺服器程序並降低基礎設施成本。
- 持續監控資源使用率，以精簡封裝密度並維持成本效益，同時不影響玩家體驗。

GAMECOST02-BP03 選取適當的運算定價選項以降低成本

在各種執行個體類型和運算選項中執行遊戲伺服器軟體的效能測試，以判斷哪個選項最適合您的遊戲。

未建立此最佳實務時的曝險等級：中

實作指引

除了為您的工作負載有效利用正確的 EC2 執行個體類型之外，請考慮哪些可用的運算定價選項最適合您的成本最佳化目標。有多種定價選項可供使用，包括隨需執行個體、Spot 執行個體、預留執行個體和 Savings Plans。

Savings Plans (SPs) 透過做出用量承諾提供運算折扣，非常適合無法預測 1 年或 3 年期間的預期用量的情況。它們提供像是預留執行個體的折扣，以及跨區域、執行個體系列、作業系統、租用套用這些折扣的彈性。也可以套用到 AWS Fargate，其可以是臨時遊戲的遊戲伺服器託管選項 AWS Lambda ，或用作不需要遊戲伺服器的輪換型遊戲的絕佳選項。如需詳細資訊，請參閱建置可擴展的無伺服器多玩家遊戲。

Savings Plans 會在遊戲啟動期間推出，以節省遊戲伺服器工作負載的成本，這些工作負載在遊戲發佈給觀眾時導致 EC2 執行個體花費。當遊戲操作團隊更了解遊戲在生產環境中長時間執行之後的玩家流量時，也可以在推出後推出 Savings Plans。

由於 Savings Plans 提供區域彈性，因此特別適合最佳化遊戲伺服器在跨地理區域使用量無法預測的遊戲支出。

例如，如果您的每日玩家使用模式需要至少 20 個伺服器來支援您的玩家基礎，但定期需要最多 40 個伺服器，請考慮購買 Savings Plan 承諾來涵蓋 20 個伺服器的基準，因為該使用需求是可預測且一致的，並且將導致您購買之用量承諾的最大使用率。

將 Savings Plans 的使用率最大化，並使用其他購買選項加以增強，為無法預測的遊戲伺服器使用量尖峰提供更多彈性，例如隨需執行個體和 Spot 執行個體，以實現最佳節省。

Spot 執行個體非常適合執行遊戲伺服器，因為它們提供最大的運算折扣、不需要用量承諾，而且可為不可預測和尖峰工作負載類型提供彈性。不過，Spot 執行個體可能會中斷，因此最適合遊戲工作階段持續時間較短的遊戲伺服器工作負載，或容錯能力較高的情況。

如需在 Amazon EKS 上使用 Kubernetes 搭配 EC2 Spot 執行個體執行遊戲伺服器之指導的詳細資訊，請參閱如何使用 Aurora Serverless 搭配 EC2 Spot 執行大量多玩家遊戲。

使用 Amazon EC2 Spot 執行個體來判斷中斷機率最低的集區，相較於隨需費率，可提供最大的節省。

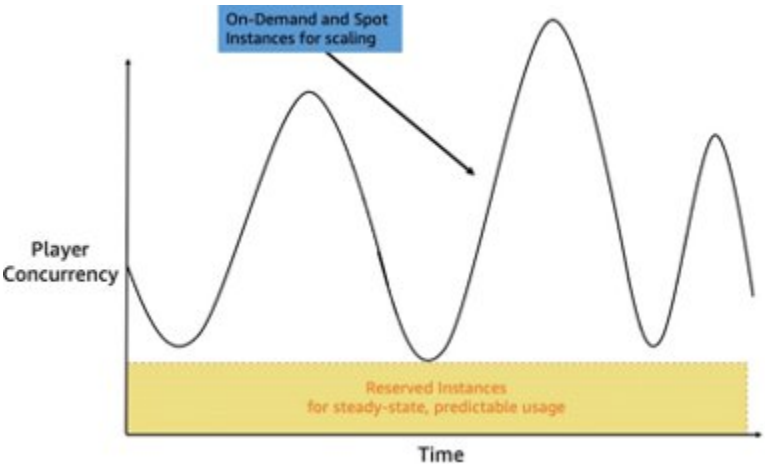
使用 Spot 時，也建議您在 中跨多個 EC2 執行個體類型和可用區域執行遊戲伺服器工作負載 AWS 區域 ，以多樣化使用容量並降低中斷風險。

考慮使用 Spot 執行個體搭配隨需執行個體，將潛在中斷對作用中遊戲工作階段的影響降至最低，並使用容量最佳化配置策略來進一步降低中斷的風險。

如需其他最佳實務，請參閱 Amazon EC2 Spot 的最佳實務。當 Spot 執行個體的中斷風險增加時，Auto Scaling 中的容量重新平衡可以用來主動監控和新增額外的容量。

Amazon GameLift FleetIQ 與 Spot 執行個體整合，以最佳化低成本 Spot 執行個體的使用，同時降低中斷的風險。如果您使用 GameLift 託管遊戲，請檢閱 GameLift 文件以選擇運算資源。如需詳細資訊，請參閱選擇受管機群的運算資源。

下圖提供範例，說明遊戲伺服器工作負載使用多個運算定價選項：



使用多個 EC2 定價選項託管遊戲伺服器

在圖表中，玩家並行會隨著時間而波動，這使得管理使用率和實現成本最佳化變得困難。為了解決這種波動，請考慮採用不同運算定價選項的混合，使用適用於 EC2 的 Savings Plans 來滿足最低用量需求，同時依賴 EC2 隨需執行個體和 EC2 Spot 執行個體來滿足玩家需求的需求。

實作步驟

- 將 Savings Plans 用於可預測的基準用量，將它們與 Spot 執行個體和隨需執行個體結合，以在用量尖峰期間實現彈性和成本最佳化。
- 將 Spot 執行個體用於具有短工作階段持續時間或更高中斷容錯性的遊戲伺服器，分散於執行個體類型和可用區域，以將風險降至最低。
- 實作 EC2 Spot Instances Advisor、容量重新平衡和 GameLift FleetIQ 等工具，以最佳化 Spot 執行個體用量並主動管理中斷。

資料傳輸費用

GAMECOST03：如何最佳化遊戲基礎設施的資料傳輸成本？

遊戲可以在玩家的遊戲用戶端裝置和遊戲基礎設施之間透過網際網路傳輸大量資料，以提供遊戲體驗，以及遊戲基礎設施的元件之間。

例如，當玩家下載遊戲內容更新到遊戲用戶端、將遊戲進度狀態儲存到雲端、與好友進行即時多玩家遊戲工作階段，以及遊戲基礎設施在區域和可用區域之間傳輸資料時，就會發生資料傳輸。請務必了解資料傳輸在遊戲工作負載中發生的位置，以最佳化您的架構選擇，進而降低此資料傳輸成本。

若要最佳化遊戲工作負載的資料傳輸成本，請考慮下列最佳實務：

最佳實務

- GAMECOST03-BP01 選擇適合使用者產生內容的儲存體類型，以降低成本
- GAMECOST03-BP02 最佳化遊戲後端的資料庫

GAMECOST03-BP01 選擇適合使用者產生內容的儲存體類型，以降低成本

在您的遊戲中產生和存放的每種資料類型都有獨特的特性，您在決定工作負載的正確儲存解決方案時應考慮這些特性。

未建立此最佳實務時的曝險等級：低

實作指引

使用 Amazon S3 物件生命週期管理，將物件資料存放在最具成本效益的儲存類別中。Amazon S3 提供多個儲存類別和物件生命週期管理，可讓您直接設定簡單且精細的政策，在儲存層之間自動轉換資料以降低成本。根據預設，與其僅將資料儲存在 S3 標準儲存體方案中，請考慮設定生命週期組態，以在層之間自動隨著時間轉換資料，或使用 S3 Intelligent-Tiering 儲存體方案進行未知或變更的存取模式。

或者，S3 Intelligent-Tiering 可以經濟實惠地自動在層之間轉換資料，建議將其作為預設儲存類別，因為它提供成本最佳化，而無需手動設定生命週期政策，現在是小型和短期物件的最佳選擇。如需詳細資訊，請參閱 Amazon S3 Intelligent-Tiering – 改善了短期和小型物件的成本最佳化。

Amazon S3 的常見使用案例包括遊戲資產儲存、靜態內容、遊戲日誌、資料湖儲存和備份。對於需要檔案系統的使用案例，例如在開發期間將共用檔案系統連接到工作站，請考慮使用 Amazon Elastic File System (Amazon EFS)，這會提供不同的儲存類別，並在新增和移除不需要管理基礎設施的檔案時自動擴展和縮減。

Amazon S3 One Zone-IA 是與遊戲內工作階段、配對或其他暫時性資訊相關的暫時性資料的理想儲存選項，可視需要重新建立。這類遊戲資料不需要跨多個可用區域 (AZs) 的備援。此低成本儲存類別非常適合用於分析或偵錯的玩家動作、遊戲事件和其他遙測資料記錄。

使用 S3 Express One Zone 進行此類遊戲資料的主要成本最佳化優勢是相較於標準 S3 儲存類別，可大幅節省成本，並降低高達 20% 的儲存成本。這對於具有大量資料的遊戲特別有利，這些大量資料不需要與關鍵任務應用程式資料相同的耐用性和可用性。透過利用 S3 One Zone，遊戲開發人員和發佈者可以最佳化其雲端儲存成本，而不會犧牲整體玩家體驗。

實作步驟

- 設定 Amazon S3 生命週期政策，以在儲存體方案之間轉換資料，或使用 S3 Intelligent-Tiering 做為預設選項，以在變更存取模式時自動進行成本最佳化。
- 使用 S3 單區域不常存取進行暫時性遊戲工作階段資料，例如遙測和配對記錄，將儲存成本降低高達 20%，同時保持足夠的可用性。
- 對於開發期間的共用檔案系統需求，請使用 Amazon EFS 以彈性容量和多個儲存類別簡化儲存管理。

GAMECOST03-BP02 最佳化遊戲後端的資料庫

遊戲非常依賴資料庫來存放各種關鍵資料，從玩家設定檔和庫存到遊戲內的微型交易和進度指標。資料庫在管理遊戲的社交方面也扮演重要角色，例如建立和維護玩家群組、隊伍和強制執行管制政策。隨著遊戲玩家基礎的成長，相關聯的資料庫成本將無可避免地增加，以滿足不斷增加的資料和用量需求。

未建立此最佳實務時的曝險等級：中

實作指引

對於在 Amazon Aurora 上執行的遊戲後端，可以採用多種成本最佳化策略。其中一個主要建議是根據使用模式自動擴展您的僅供讀取複本，動態擴展或縮減複本數量，以處理流量的波動。這表示您要支付真正需要的資源。另一個最佳化策略是將用於遊戲分析的僅供讀取複本取代為將資料庫快照匯出至 Amazon S3，因為 S3 儲存服務通常比佈建的 Aurora 資料庫執行個體更實惠。如需詳細資訊，請參閱將資料庫快照資料匯出至 Amazon RDS 的 Amazon S3。

探索為核心資料庫執行個體使用 Amazon Aurora 的預留資料庫執行個體，並轉換至 Aurora Serverless 組態，也可讓您更靈活地控制資源使用率，進而大幅節省長期成本。

同樣地，對於使用 Amazon DynamoDB 的遊戲後端，採用 DynamoDB 隨需容量模式可能是有效的選擇，特別是對於新的或無法預測的工作負載，因為它可讓您僅支付所耗用的資源，而不需要過度佈建。隨著您的遊戲流量模式隨著時間變得更穩定且可預測，您可以轉換到 DynamoDB 佈建容量模式，透過更好的容量規劃來節省成本。在 DynamoDB 資料表上啟用自動調整規模是另一個金鑰最佳化，可讓服務根據流量的波動動態調整佈建容量。在啟動之前，在開發環境中測試遊戲的資料結構，以尋找和移除

不必要的本機次要索引 (LSIs) 和全域次要索引 (GSIs)。這可以大幅節省遊戲資料儲存和操作的成本。從遊戲後端程式碼中移除效率低下的掃描操作，以偏好更精準的查詢、購買 Amazon DynamoDB 預留容量，以及利用 DynamoDB Streams 搭配 AWS Lambda 觸發程序來處理遊戲後端事件，可以進一步最佳化 DynamoDB 成本。如需詳細資訊，請參閱在 DynamoDB 中查詢和掃描資料的最佳實務。

透過為 Amazon Aurora 和 DynamoDB 實作這些成本最佳化策略，遊戲開發人員和發佈者可以大幅降低其遊戲後端資料庫的花費。

實作步驟

- 使用 Aurora 僅供讀取複本自動擴展和資料庫快照匯出至 Amazon S3，以經濟實惠的方式處理不斷變化的流量和分析需求。
- 最佳化 DynamoDB 成本，方法是從新工作負載的隨需容量開始，透過自動調整規模來轉換到佈建的容量，以取得可預測的流量，以及移除未使用的 LSIs 和 GSIs。
- 避免對目標查詢有利的無效率掃描操作、使用預留執行個體或預留容量，以及搭配 使用 DynamoDB Streams AWS Lambda 進行事件處理。

管理需求和供應資源

沒有管理需求和供應資源的 Games Lens 特定最佳實務。

如需管理需求和提供資源的詳細資訊，請參閱成本最佳化支柱 - AWS Well-Architected Framework。

隨著時間進行最佳化

隨著時間的推移，沒有針對 Games Lens 的最佳實務進行最佳化。

如需隨時間最佳化成本的詳細資訊，請參閱成本最佳化支柱 - AWS Well-Architected Framework。

Resources

請參閱下列資源，進一步了解成本最佳化的最佳實務：

相關文件：

- 如何降低 Amazon VPC 中 NAT 閘道的資料傳輸費用？
- Amazon GameLift FleetIQ 轉接器 Agones 簡介
- 如何在 Amazon VPC 中找到 NAT 閘道流量的主要參與者？

- 為您的全域遊戲伺服器選擇正確的運算策略
- AWS Well-Architected 實驗室 – 符合成本效益的資源
- Amazon VPC CNI 外掛程式會增加每個節點的 Pod 限制
- 成本最佳化的架構最佳實務
- 使用 Amazon SageMaker AI RL 和 Amazon EKS 減少玩家等待時間和正確調整運算配置大小
- AWS Compute Optimizer
- Electronic Arts 使用 Amazon S3 Intelligent-Tiering 和 Amazon Glacier 最佳化儲存成本和操作
- 將 Windows 工作負載遷移至 Linux，避免不友善的授權實務
- 常見架構的資料傳輸成本概觀
- AWS 和 Kubecost 協同合作，為 EKS 客戶提供成本監控
- 奠定基礎：設定您的環境進行成本最佳化
- Amazon EC2 Spot 執行個體概觀

永續性

永續性支柱提供設計原則、操作指引、最佳實務和改善計劃，以協助達成 AWS 工作負載的永續性目標。

您可以在永續性支柱 - AWS Well-Architected Framework 白皮書中找到有關實作的實作指引。

重點領域

- 設計原則
- 區域選擇
- 因應需求
- 軟體和架構
- 資料管理
- 硬體和服務
- Resources

設計原則

遊戲產業的永續性正在世界不同區域以不同的速率發展。藉助北美大片地區的永續能力和歐洲和英國的永續要求，架構師在不久的將來有數種方法來實現更綠色的工作負載。Well-Architected 永續性支柱可用於實現各種工作負載的這些措施。

鏡頭的本節說明可用於遊戲工作負載的幾個最佳實務。

- 為遊戲使用者資料選取適當的儲存類型。
- 請注意將刪除重複資料的資料生命週期政策，並從工作負載清除不需要的資料。
- 選擇正確調整運算資源的部署大小。
- 將無伺服器用於簡短和交易程序。

區域選擇

Games Lens 沒有特定的區域選擇最佳實務。如需詳細資訊，請參閱永續性支柱 - AWS Well-Architected Framework。

因應需求

沒有針對 Games Lens 的需求最佳實務的一致性。如需詳細資訊，請參閱永續性支柱 - AWS Well-Architected Framework。

軟體和架構

Games Lens 沒有特定的軟體和架構最佳實務。如需詳細資訊，請參閱永續性支柱 - AWS Well-Architected Framework。

資料管理

GAMESUS01：如何管理遊戲系統中的使用者和遊戲資料？

開發資料生命週期策略，透過僅保留關鍵的歷史資料來最佳化資料儲存和相關性。

最佳實務

- GAMESUS01-BP01 使用適合使用者內容、訂閱者資訊和遊戲內購買模式的儲存技術
- GAMESUS01-BP02 使用生命週期政策或 TTL 過期來刪除不必要的遊戲使用者資料、日誌檔案或已棄用資產

GAMESUS01-BP01 使用適合使用者內容、訂閱者資訊和遊戲內購買模式的儲存技術

您應該依類型、保留需求和存取頻率來分類資料。這可讓您為遊戲或後端服務產生的各種資料類型選取最最佳化的儲存解決方案。快速變更的資料應存放在金鑰值或記憶體內資料庫服務中。交易資料應存放在關聯式資料庫服務中。大型檔案、遊戲資產或使用者產生的內容應存放在物件儲存服務中。

未建立此最佳實務時的曝險等級：高

實作指引

遊戲會產生和使用各種資料類型，這些資料類型需要針對存取頻率、延遲和成本進行最佳化的儲存解決方案。存放的資料應使用標籤分類，以區分可以移除或需要長期存放的資料。

下列服務適用於各種遊戲使用案例：

Amazon Aurora（與 MySQL 和 PostgreSQL 相容）提供高可用性、低延遲和自動擴展，使其成為處理大量交易資料的理想選擇，例如玩家帳戶管理和身分驗證、遊戲內經濟體、排行榜和玩家排名、遊戲狀態持久性、事件和行銷活動管理，以及多區域和高可用性部署。

Amazon DynamoDB 是全受管的 NoSQL 資料庫，以其低延遲、高輸送量和無縫可擴展性著稱，因此非常適合處理即時玩家資料、工作階段管理、庫存、遊戲經濟、即時多玩家遊戲狀態、配對、事件記錄和全球受眾擴展。

Amazon DocumentDB（與 MongoDB 相容）提供可擴展、低延遲的文件導向資料庫服務，非常適合存放彈性的半結構化資料，例如庫存系統、玩家設定檔和自訂的、遊戲世界和程序產生的內容、社交和玩家互動、分析和行為追蹤，以及遊戲內中繼資料和組態。

Amazon ElastiCache 支援 Redis 或 Memcached 的記憶體內快取，可提供快速的資料存取和縮短回應時間，這對於即時多玩家遊戲至關重要，其中速度和效能對於順暢的使用者體驗至關重要。ElastiCache 用於即時排行榜、工作階段管理、快取遊戲中繼資料、遊戲內聊天和傳訊、配對、即時分析和遙測，以及高流量事件的擴展。

Amazon Simple Storage Service (S3) 可用來存放遊戲資產、影片、圖片、文字日誌檔案等物件。S3 是一種物件儲存服務，提供業界領先的可擴展性、資料可用性、安全性和效能。

如果它提供多個儲存類別，支援頻繁和不常的資料存取，以及經濟實惠的封存儲存。對於在整個開發過程中經常存取的資料，工作室應將物件存放在 S3 標準中，以實現低延遲和高輸送量效能。對於經常從熱到冷的資料，反之亦然，工作室應該調查 S3 Intelligent-Tiering。Intelligent-Tiering 會監控資料的存取模式，並自動將資料移至最具成本效益的存取層。

對於需要高輸送量、低延遲且適合位於單一可用區域的工作室，請使用 S3 Express One Zone。這會將資料複寫到單一可用區域，並且相較於 S3 標準，可以改善資料存取速度。對於歷史資料的深度封存需求，Amazon 也提供 Amazon Glacier。Amazon Glacier 儲存類別專為資料封存而打造，為您提供高效能、擷取彈性和低成本的雲端封存儲存。

Amazon Elastic Block Store 可用來存放遊戲伺服器的二進位檔、可執行檔，以及遊戲伺服器或資產儲存庫運作所需的組態。您應該快照並刪除未連接到 EC2 執行個體的未使用磁碟區。這可減輕您在降低不必要的服務和硬體使用量時產生的儲存費用。

實作步驟

- 依類型、保留需求和存取頻率分類遊戲資料，標記資料以區分短期和長期儲存需求。
- 將 Amazon Aurora 用於交易資料、將 DynamoDB 用於即時玩家資料、將 DocumentDB 用於半結構化資料，以及將 ElastiCache 用於低延遲快取的時間關鍵遊戲資訊。

- 將遊戲資產、日誌和使用者產生的內容儲存在 Amazon S3 中，根據存取模式和封存需求選擇適當的儲存類別（例如 Intelligent-Tiering、One Zone 和 Glacier），並使用 EBS 進行遊戲伺服器二進位檔和組態，並進行定期快照管理。

GAMESUS01-BP02 使用生命週期政策或 TTL 過期來刪除不必要的遊戲使用者資料、日誌檔案或已棄用資產

您可以使用標籤和資料類型來建立生命週期政策或 TTL 的 ，將資料移至封存儲存或從服務中完全移除。這可能包括暫時組態、過期的封存內容，以及不再需要的歷史日誌。大多數 服務都支援標記。

未建立此最佳實務時的曝險等級：高

實作指引

對於存放在 S3 中的資料，您可以使用生命週期政策將資料移至不常存取和封存的儲存層。您可於 S3 生命週期組態中，定義將物件從一個儲存類別轉換為另一個儲存類別的規則，並存於儲存體成本。當您不清楚物件的存取模式時，或如果您的存取模式會隨時間變更，您可以將物件轉換為 S3 Intelligent-Tiering 儲存類別，自動節省成本。

Amazon S3 支援瀑布模型以在儲存類別間轉換，如下圖所示。

您可以將轉換動作新增至 S3 生命週期組態，以指示 Amazon S3 在物件的生命週期結束時刪除物件。當物件根據其生命週期組態達到其生命週期結束時，Amazon S3 會根據儲存貯體所在的 S3 版本控制狀態採取過期動作：

- 非版本控制的儲存貯體：Amazon S3 會將物件排入佇列以進行移除，並以非同步方式將其移除，永久移除物件。
- 啟用版本控制的儲存貯體：如果目前的物件版本不是刪除標記，Amazon S3 會新增具有唯一版本 ID 的刪除標記。如此會讓目前的版本成為非目前的版本，而刪除標記成為目前版本。
- 暫停版本控制儲存貯體：Amazon S3 會建立刪除標記，並以 null 做為版本 ID。此刪除標記會將物件版本取代為版本階層中的 null 版本 ID，這會有效地刪除物件。
- 當您新增儲存貯體的生命週期組態時，組態規則會套用至現有物件以及稍後新增的物件。例如，如果您今天使用過期動作新增生命週期組態規則，導致具有特定字首的物件在建立後 30 天過期，Amazon S3 將排入佇列，以移除超過 30 天且具有指定字首的現有物件。

DynamoDB 的存留時間 (TTL) 功能是一種具成本效益的方法，可用於刪除不再需要的項目。TTL 可讓您為每個項目定義過期時間戳記，以標示該項目何時不再需要。DynamoDB 會在項目過期後的幾天內自動刪除它們，且不會耗用寫入輸送量。

- 若要使用 TTL，請先在資料表中啟用該功能，接著定義一個屬性以儲存 TTL 過期時間戳記。時間戳記必須以秒為單位，並採用 Unix epoch 時間格式進行儲存。每次建立或更新項目時，您可計算其過期時間，並將結果儲存在 TTL 屬性中。
- 系統通常會在過期後的幾天內刪除具有有效、過期 TTL 屬性的項目。您仍可更新尚未刪除的過期項目，例如變更或移除其 TTL 屬性。在更新過期項目時，建議使用條件運算式，以確保該項目未在此期間被刪除。使用篩選運算式，從掃描與查詢結果中排除過期項目。
- 被刪除的項目運作方式與一般刪除操作所刪除的項目相同。刪除後，項目會以服務刪除的形式進入 DynamoDB Streams，而不是使用者刪除，並從本機次要索引和全域次要索引中移除，就像其他刪除操作一樣。

使用 ElastiCache for Redis，您可以使用 TTLs 或快取金鑰上的過期來控制快取資料的新鮮度。在設定時間過後，金鑰會從快取中刪除，而且需要存取原始伺服器資料存放區，才能到達更新的資料。

- 兩個原則會決定要套用的適當 TTLs，以及要實作的快取模式類型。首先，請務必了解基礎資料的變更率。其次，請務必評估將過期資料傳回至應用程式的風險，而非其更新的複本。
- 透過經常變更的動態資料，您可能想要套用較低的 TTLs，以與主要資料庫相符的變更速率來使資料過期。這可降低傳回過時資料的風險，同時仍提供緩衝來卸載資料庫請求。
- 同樣重要的是，即使您只快取資料幾分鐘或幾秒鐘，相較於更長的持續時間，適當地將 TTLs到快取的金鑰也可能會導致效能提升，以及遊戲的整體更佳玩家體驗。

實作步驟

- 使用 Amazon S3 生命週期政策將物件轉換為不常存取或封存層，並設定過期動作以根據生命週期規則刪除不必要的物件。
- 在 DynamoDB 資料表中啟用存留時間 (TTL)，以自動刪除過期項目而不耗用寫入輸送量，以 Unix epoch 時間定義過期時間戳記。
- 根據資料變更率和過時資料的風險承受能力，為 ElastiCache 金鑰設定適當的 TTLs，促進快取資料更新並改善玩家體驗。

硬體和服務

GAMESUS02：如何在遊戲後端中管理運算資源的使用？

Studios 應該開發一種運算策略，使用混合不同的運算類型、受管服務和節省計劃來最佳化您的用量。您也應該最佳化遊戲伺服器和後端服務的封裝方式，以運算執行個體，以減少不需要的資源數量。

最佳實務

- GAMESUS02-BP01 針對適當的運算工作負載選取受管服務
- GAMESUS02-BP02 僅在需要時調整運算大小並部署 GPU 效能

GAMESUS02-BP01 針對適當的運算工作負載選取受管服務

建構您的遊戲後端服務，以針對事件驅動或高度可變的流量工作負載使用受管服務。由於多租用戶控制平面，受管服務會將基礎設施的管理轉移到 ， AWS 並將環境影響分散到多個使用者。

未建立此最佳實務時的曝險等級：高

實作指引

AWS AWS Lambda AWS Fargate （容器） 和 Amazon Gamelift （遊戲伺服器協調） 等 服務可以執行程式碼、容器或協調您的遊戲伺服器，而無需管理基礎基礎設施。這些服務會根據玩家需求自動擴展，而且您只需支付所使用資源的費用。由於基礎基礎設施是代表您管理的，因此您可以僅專注於遊戲和後端服務的需求。

可使用 AWS Lambda 來執行程式碼，無需佈建或管理伺服器。Lambda 會在高可用性的運算基礎設施上執行程式碼，並執行運算資源的管理，包括伺服器和作業系統維護、容量佈建和自動擴展，以及記錄。使用 Lambda，您需要在 Lambda 支援的其中一種語言執行時間中提供程式碼。Lambda 適用於處理遊戲事件、玩家身分驗證、遊戲內購買處理和配對請求。Lambda 會根據事件數量自動擴展，並可處理流量中非預期的峰值。

AWS Fargate 是一種無伺服器運算引擎，適用於同時與 Amazon Elastic Container Service (ECS) 和 Amazon Elastic Kubernetes Service (EKS) 搭配使用的容器。 AWS Fargate 透過減輕佈建和管理伺服器的需求，可讓您指定和支付每個應用程式的資源，並透過設計的應用程式隔離來提高安全性。Fargate 非常適合處理玩家設定檔、狀態管理和配對的後端服務。

Amazon GameLift 是一項受管服務，用於部署、操作和擴展工作階段型多玩家遊戲的專用遊戲伺服器。您可以在幾分鐘內在雲端部署您的第一個遊戲伺服器，在前期軟體開發中節省高達數千小時的工程時數，並降低通常會導致開發人員從設計中刪除多玩家功能的技術風險。

實作步驟

- AWS Lambda 用於事件驅動型工作負載，例如處理遊戲事件、玩家身分驗證、遊戲內購買和配對請求，並利用其自動擴展和無伺服器管理。

- AWS Fargate 使用 ECS 或 EKS 部署後端服務，例如玩家設定檔、狀態管理和配對、移除伺服器管理和改善應用程式隔離。
- 使用 Amazon GameLift 為工作階段型多玩家遊戲部署和擴展專用遊戲伺服器，從而減少開發時間和操作複雜性。

GAMESUS02-BP02 僅在需要時調整運算大小並部署 GPU 效能

建構您的遊戲伺服器和後端，以有效率地利用運算資源。過度佈建運算可能會導致不必要的成本，並將閒置或利用率不足的資源數量降至最低。GPU 執行個體應該用來支援特定的開發工作，例如 Unreal 中的 HLOD 重建，或者如果您的遊戲伺服器設計需要它們。這可大幅降低工作負載的環境影響和成本。

未建立此最佳實務時的曝險等級：高

實作指引

您應該最佳化遊戲伺服器和後端服務，以使用多個 EC2 執行個體類型和所需的最少執行個體數量。這會增加可在開發期間或遊戲啟動時滿足您的需求的可用執行個體數量。您也應該將執行個體類型與您部署的特定工作負載相符。Compute Optimized 執行個體支援各種使用案例，包括遊戲伺服器和後端服務，例如配對。記憶體最佳化執行個體旨在為處理記憶體中大型資料集的工作負載提供快速效能。依高效能需求使用 GPU 執行個體，但不適用於一般運算任務。如果可以，請建構您的服務或遊戲伺服器，以在具有 AWS Graviton 執行個體的 ARM 上執行。Graviton 是最高效能的節能執行個體類型 AWS。相較於 x86 執行個體類型，它們也提供更高的效能和成本。

使用 AWS Compute Optimizer來識別最佳 AWS 資源組態，例如 Amazon Elastic Compute Cloud (EC2) 執行個體類型、Amazon Elastic Block Store (EBS) 磁碟區組態、 上的 Amazon Elastic Container Service (ECS) 服務的任務大小 AWS Fargate、商業軟體授權、 AWS Lambda 函數記憶體大小，以及 Amazon Relational Database Service (RDS) 資料庫執行個體類別，使用機器學習來分析歷史使用率指標。Compute Optimizer 提供一組 APIs 和主控台體驗，透過為您的工作負載建議最佳 AWS 資源來降低成本並提高 AWS 工作負載效能。

實作步驟

- 使用遊戲伺服器的運算最佳化執行個體、大型資料集的記憶體最佳化執行個體，以及僅適用於 HLOD 重建或 GPU 相依遊戲伺服器等任務的 GPU 執行個體，將運算資源與特定工作負載配對。
- 與 x86 執行個體相比，盡可能部署 AWS Graviton 執行個體來最佳化運算使用率，以獲得能源效率、更好的效能和節省成本。
- 使用 AWS Compute Optimizer 來分析歷史使用率，並建議 EC2、 AWS ECS AWS Lambda和 Amazon RDS 工作負載最有效率的組態，以降低成本並改善效能。

Resources

請參閱下列資源，進一步了解我們有關永續性的最佳實務。

- UN：遊戲產業聚焦於對地球的威脅
- 中：遊戲開發中的環境永續性：為更綠色的未來負責任地玩耍

金鑰 AWS 服務

- Amazon Aurora
- Amazon DynamoDB
- Amazon DocumentDB (with MongoDB compatibility)
- Amazon ElastiCache
- Amazon S3
- Amazon S3 儲存類別
- Amazon S3 Intelligent-Tiering 儲存類別
- Amazon S3 Express One Zone 儲存類別
- Amazon Elastic Block Store
- AWS Lambda
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Amazon GameLift
- AWS Compute Optimizer

結論

遊戲旨在為全球玩家觀眾提供娛樂體驗，並具有通常無法預測且可變的使用特性。Games Industry Lens 描述了通常構成遊戲架構的常見案例類型，並提供一組問題和最佳實務，供您在雲端建置和操作遊戲時考慮。透過將此架構套用至您的遊戲架構，您可以在雲端中建置可靠、安全、有效率且符合成本效益的遊戲。

貢獻者

下列個人對本文件有所貢獻：

- Adam Hatfield，Amazon Web Services 資深解決方案架構師
- Brady Webb，Amazon Web Services 技術客戶經理
- Bruce Ross – Amazon Web Services Well-Architected Lens 領導者資深解決方案架構師
- Caleb Cecil，Amazon Web Services 副解決方案架構師
- Carlos Perez，Amazon Web Services 雲端最佳化成功 SA
- Chase Herrington，Amazon Web Services ESL 技術客戶經理。
- Chris Blackwell，Amazon Web Services 資深解決方案架構師
- Corey Ouderkirk，Amazon Web Services 資深技術客戶經理
- Derek Roomsvicencio、Prototyping SA、Amazon Web Services
- Erik Ynigo Becerril，Amazon Web Services 資深解決方案架構師
- Grzegorz Ochmanski，Amazon Web Services 資深解決方案架構師
- Hadrian Baron，Amazon Web Services 資深技術客戶經理
- Ian Armbruster，Amazon Web Services 資深客戶解決方案經理
- Jed Obray，Amazon Web Services 資深技術客戶經理
- Khurram Khokhar，Amazon Web Services 資深技術客戶經理
- Kyle Somers，Amazon Web Services 解決方案架構資深經理
- Madhuri Srinivasan，Amazon Web Services Well-Architected 資深技術作者
- Matthew Wygant，高級 TPM 指引， Well-Architected，Amazon Web Services
- Nataliya Godunok，Amazon Web Services 雲端最佳化成功 SA
- Nirav Doshi，Amazon Web Services 首席解決方案架構師
- Olivia Liddell，Amazon Web Services 解決方案架構師
- Randy James，Amazon Web Services 首席技術客戶經理
- Reou Ando，Amazon Web Services 遊戲解決方案架構師
- Richard Raseley，Amazon Web Services 資深技術客戶經理
- Sam Patzer，Amazon Web Services 資深解決方案架構師
- Scott Selinger，Amazon Web Services 資深解決方案架構師
- Sean Allen，Amazon Web Services 資深解決方案架構師

- Serge Poueme，Amazon Web Services 資深解決方案架構師
- Stewart Matzek，Amazon Web Services Well-Architected 資深技術作者
- Trenton Potgieter，Amazon Web Services 資深解決方案架構師 AI/ML/Analytics

文件修訂

若要收到此白皮書更新的通知，請訂閱 RSS 摘要。

變更	描述	日期
新的鏡頭版本	使用新的最佳實務指引更新整個鏡頭。	2025 年 12 月 9 日
初次出版	白皮書首次發佈。	2021 年 11 月 19 日

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。