aws

Entwicklerhandbuch

Amazon GameLift

Amazon GameLift: Entwicklerhandbuch

Table of Contents

Was ist Amazon GameLift?

Sie können Amazon verwenden, GameLift um dedizierte, kostengünstige Server in der Cloud für sitzungsbasierte Multiplayer-Spiele bereitzustellen, zu betreiben und zu skalieren. Amazon basiert auf einer AWS globalen Computerinfrastruktur und GameLift unterstützt Sie bei der Bereitstellung leistungsstarker, hochzuverlässiger Spieleserver und skaliert gleichzeitig Ihre Ressourcennutzung dynamisch, um der weltweiten Nachfrage der Spieler gerecht zu werden.

Einsatzmöglichkeiten von Amazon GameLift

Amazon GameLift unterstützt diese und weitere Anwendungsfälle:

- Verwenden Sie Ihre eigenen benutzerdefinierten Multiplayer-Spieleserver oder verwenden Sie ready-to-go Echtzeitserver, um Ihre Spiele zu hosten.
- Führen Sie kostengünstige Hosting-Ressourcen mithilfe von Amazon Elastic Compute Cloud (Amazon EC2) Spot-Instances aus.
- Skalieren Sie automatisch die Menge der Hosting-Ressourcen, die Ihr Spiel benötigt, je nach Nutzung.
- Mit Amazon FleetIQ verwalten Sie Ihre Amazon EC2-Rechenressourcen an einem Ort. GameLift
- Kombiniere Spieler in Multiplayer-Spielen mit Amazon GameLiftFlexMatch.
- Testen Sie Ihre Gameserver- und Client-Builds iterativ mit Amazon GameLiftAnywhere.
- Verwenden Sie Ihre eigene Hardware und verwalten Sie alles an einem Ort mit Amazon GameLiftAnywhere.

> **Tip**
>
> Informationen zum Ausprobieren des GameLift Amazon-Gameserver-Hostings finden Sie unterErste Schritte mit Amazon GameLift.

Beginnen Sie mit GameLift Amazon-Lösungen

GameLiftAmazon-Lösungen für Spieleentwickler

- GameLiftAmazon-Hosting für benutzerdefinierte Server

- GameLiftAmazon-Hosting mit Echtzeitservern
- Amazon GameLift FleetIQ für das Hosting auf Amazon EC2
- Amazon GameLift FlexMatch für Matchmaking
- GameLiftAnywhereAmazon-Hardwarehosting

GameLiftAmazon-Hosting für benutzerdefinierte Server

Amazon GameLift ersetzt die Arbeit, die erforderlich ist, um Ihre eigenen benutzerdefinierten Spieleserver zu hosten. Dank der automatischen Skalierungsfunktionen können Sie vermeiden, dass Sie für mehr Ressourcen bezahlen, als Sie benötigen. Die automatische Skalierung hilft auch dabei, sicherzustellen, dass neue Spieler immer Spiele zur Verfügung haben, an denen sie teilnehmen können, und das mit minimaler Wartezeit.

Weitere Informationen zu GameLift Amazon-Hosting finden Sie unterSo GameLift funktioniert Amazon.

Schlüsselfunktionen

- Verwenden Sie die GameLift Verwaltungsfunktionen von Amazon, einschließlich automatischer Skalierung, Warteschlangen an mehreren Standorten und Platzierung von Spielsitzungen.
- Stellen Sie Spieleserver bereit, die unter den Betriebssystemen Amazon Linux oder Windows Server laufen.
- Verwalten Sie Spielsitzungen und Spielersitzungen.
- Richten Sie eine benutzerdefinierte Statusverfolgung für Serverprozesse ein, um Probleme zu erkennen und Prozesse mit schlechter Leistung zu beheben.
- Verwalte deine Spielressourcen mithilfe von AWS CloudFormation Vorlagen für AmazonGameLift.

GameLiftAmazon-Hosting mit Echtzeitservern

Verwenden Sie Echtzeitserver, um Spiele zu starten, für die keine maßgeschneiderten Gameserver erforderlich sind. Diese leichte Serverlösung bietet Spieleserver, die Sie an Ihr Spiel anpassen können.

Weitere Informationen zum GameLift Amazon-Hosting mit Realtime Servern finden Sie unterIntegration von Spielen mit Amazon GameLift Realtime Servern.

Schlüsselfunktionen

- Verwenden Sie die GameLift Verwaltungsfunktionen von Amazon, einschließlich automatischer Skalierung, Warteschlangen an mehreren Standorten und Platzierung von Spielsitzungen.
- Verwenden Sie GameLift Amazon-Hosting-Ressourcen und wählen Sie die Art der AWS Computerhardware für Ihre Flotten.
- Nutze einen vollständigen Netzwerk-Stack für die Interaktion zwischen Spielclient und Server.
- Nutzen Sie die Kernfunktionalität des Spielservers mit anpassbarer Serverlogik.
- Nehmen Sie Live-Updates für Echtzeitkonfigurationen und Serverlogik vor.

Amazon GameLift FleetIQ für das Hosting auf Amazon EC2

Verwenden Sie Amazon GameLift FleetIQ, um direkt mit Ihren Hosting-Ressourcen in Amazon EC2 und Amazon EC2 Auto Scaling zu arbeiten. Dies bietet den Vorteil der GameLift Amazon-Optimierungen für kostengünstiges, robustes Game-Hosting. Diese Lösung richtet sich an Spieleentwickler, die mehr Flexibilität benötigen als die vollständig verwalteten GameLift Amazon-Lösungen.

Informationen darüber, wie Amazon GameLift FleetIQ mit Amazon EC2 und EC2 Auto Scaling für Game-Hosting zusammenarbeitet, finden Sie im Amazon GameLift FleetIQ Developer Guide.

Schlüsselfunktionen

- Holen Sie sich mithilfe des FleetIQ-Algorithmus ein optimiertes Spot-Instance-Balancing.
- Nutze die Routing-Funktionen für Spieler, um die Ressourcen deines Gameservers effizient zu verwalten und für ein besseres Spielerlebnis zu sorgen, wenn du an Spielen teilnimmst.
- Skalieren Sie die Hosting-Kapazität automatisch auf der Grundlage der Spielernutzung.
- Verwalten Sie Amazon EC2-Instances direkt in Ihren eigenen. AWS-Konto
- Verwenden Sie eines der unterstützten ausführbaren Gameserver-Formate, einschließlich Windows, Linux, Container und Kubernetes.

Amazon GameLift FlexMatch für Matchmaking

Verwenden Sie diese FlexMatch Option, um benutzerdefinierte Regelsätze zu erstellen, um Multiplayer-Matches für Ihr Spiel zu definieren. FlexMatchverwendet Regelsätze, um kompatible Spieler für jedes Spiel zu vergleichen und den Spielern das ideale Multiplayer-Erlebnis zu bieten.

Weitere Informationen zu FlexMatch finden Sie unter Was ist Amazon GameLiftFlexMatch?

Schlüsselfunktionen

- Bringen Sie die Geschwindigkeit der Match-Erstellung und die Match-Qualität in Einklang.
- Kombiniere Spieler oder Teams anhand definierter Eigenschaften.
- Definieren Sie Regeln, um Spieler basierend auf der Latenz in Matches einzuteilen.

GameLiftAnywhereAmazon-Hardwarehosting

Verwenden Sie Amazon GameLiftAnywhere, um Hardware überall in Ihrer Umgebung in Ihr GameLift Amazon-Game-Hosting zu integrieren. Sie können Anywhere Flotten und EC2-Flotten in Matchmaker- und Spielsitzungswarteschlangen integrieren, um das Matchmaking und die Platzierung von Spielen auf Ihrer Hardware zu verwalten.

Weitere Hinweise zum Testen mit Anywhere finden Sie unterTesten Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten. Weitere Informationen zum Einrichten einer Anywhere Flotte finden Sie unterGameLiftAmazon-Flotten einrichten.

Schlüsselfunktionen

- Führe schnelle, iterative Tests deiner Gameserver- und Client-Builds durch.
- Verwenden Sie die bereitgestellten GameLift Amazon-Tools, um Spiele auf Ihrer eigenen Hardware bereitzustellen.
- Verwenden Sie überall Hardware, die Ihren Playern am nächsten ist.

Zugriff auf Amazon GameLift

Verwenden Sie diese Tools, um mit Amazon zu arbeitenGameLift.

Amazon GameLift SDKs

> Die Amazon GameLift SDKs enthalten die Bibliotheken, die für die Kommunikation mit Amazon GameLift über Ihre Spieleclients, Spieleserver und Spieledienste erforderlich sind. Weitere Informationen finden Sie unter Entwicklungsunterstützung mit Amazon GameLift.

Amazon GameLift Realtime-Client-SDK

Das Realtime Client SDK ermöglicht es einem Spielclient, sich mit dem Realtime-Server zu verbinden, an Spielsitzungen teilzunehmen und mit anderen Spielern synchron zu bleiben. Laden Sie das SDK herunter und erfahren Sie mehr über API-Aufrufe mit der Realtime Server-Client-API (C#).

GameLiftAmazon-Konsole

Verwenden Sie das AWS Management Consolefür Amazon GameLift, um Ihre Spielbereitstellungen zu verwalten, Ressourcen zu konfigurieren und die Nutzungs- und Leistungskennzahlen der Spieler zu verfolgen. Die GameLift Amazon-Konsole bietet eine GUI-Alternative zur programmatischen Verwaltung von Ressourcen mit der AWS Command Line Interface (AWS CLI).

AWS CLI

Verwenden Sie dieses Befehlszeilentool, um das AWS SDK, einschließlich der GameLift Amazon-API, aufzurufen. Informationen zur AWS CLI Verwendung von finden Sie unter Erste Schritte mit dem AWS CLI im AWS Command Line InterfaceBenutzerhandbuch.

Preise für Amazon GameLift

Amazon GameLift berechnet für Instances nach Nutzungsdauer und für Bandbreite nach übertragener Datenmenge. Eine vollständige Liste der Gebühren und Preise für Amazon GameLift finden Sie unter GameLiftAmazon-Preise.

Informationen zur Berechnung der Kosten für das Hosten Ihrer Spiele oder das Matchmaking bei Amazon finden Sie unter GameLiftGenerierung von GameLift Amazon-Preisschätzungen, wo beschrieben wird, wie Sie die AWS Pricing Calculatorverwenden.

So GameLift funktioniert Amazon

Dieses Thema behandelt die Kernkomponenten für das Hosting von Spielen und beschreibt, wie Amazon Ihre Multiplayer-Spieleserver den Spielern zur Verfügung GameLift stellt.

Bist du bereit, dein Spiel für das Hosting bei Amazon vorzubereitenGameLift? AuscheckenRoadmap für GameLift verwaltetes Hosting von Amazon.

Zentrale Komponenten

GameLiftUm Amazon für das Hosten Ihres Spiels einzurichten, müssen Sie mit den folgenden Komponenten arbeiten. Das Diagramm in Spielarchitektur mit verwaltetem Amazon GameLift visualisiert die Beziehungen zwischen diesen Komponenten.

- Ein Gameserver ist die Serversoftware Ihres Spiels, die auf einer Flotte läuft. Sie laden Ihren Gameserver-Build oder Ihr Skript auf Amazon hoch GameLift und teilen es Amazon mitGameLift. Wenn Sie Amazon GameLift Anywhere oder Amazon GameLift FleetIQ verwenden, laden Sie Ihren Gameserver-Build direkt auf die Rechenressource hoch.
- Eine Spielsitzung ist ein laufendes Spiel mit Spielern. Sie definieren die grundlegenden Merkmale einer Spielsitzung, z. B. die Lebensdauer und die Anzahl der Spieler. Die Spieler stellen dann eine Verbindung zum Spielserver her, um an einer Spielsitzung teilzunehmen.
- Ein Spiele-Client ist die Software Ihres Spiels, die auf dem gerät eines Spielers ausgeführt wird. Ein Spielclient stellt über Backend-Dienste eine Verbindung zu einem Spieleserver her, um an einer Spielesitzung teilzunehmen, basierend auf den Verbindungsinformationen, die er von Amazon GameLift erhält.
- Backend-Services sind zusätzliche, benutzerdefinierte Dienste, die Aufgaben im Zusammenhang mit Amazon GameLift erledigen. Es hat sich bewährt, dass Ihre Backend-Services die gesamte Kommunikation des Spielclients mit Amazon GameLift übernehmen sollten.

Hosten von Spieleservern

Mit Amazon GameLift kannst du deine Gameserver auf drei verschiedene Arten hosten: Managed AmazonGameLift, Amazon GameLift FleetIQ und Amazon. GameLift Anywhere Weitere Informationen zu Amazon GameLift FleetIQ finden Sie unter Was ist Amazon GameLift FleetIQ?

Sie können eine Flotte entwerfen, die den Anforderungen Ihres Spiels entspricht. Weitere Informationen zum Entwerfen einer Flotte finden Sie unterLeitfaden zur GameLift Amazon-Flottenplanung.

Verwaltetes Amazon GameLift

Mit verwaltetem Amazon GameLift können Sie Ihre Spieleserver auf GameLift virtuellen Rechenressourcen von Amazon, sogenannten Instances, hosten. Richten Sie Ihre Hosting-Ressourcen ein, indem Sie eine Flotte von Instanzen erstellen und diese für den Betrieb Ihrer Spieleserver einsetzen.

Amazonas GameLift Anywhere

Mit Amazon GameLift Anywhere können Sie Ihre Spieleserver auf Computern hosten, die Sie verwalten. Richten Sie Ihre Hosting-Ressourcen ein, indem Sie eine Anywhere Flotte erstellen, die auf Ihre Rechenleistung verweist.

Flotten-Aliasse

Ein Alias ist eine Bezeichnung, die Sie zwischen Flotten übertragen können, sodass Sie auf bequeme Weise einen generischen Flottenstandort haben können. Du kannst einen Alias verwenden, um Spielclients von einer Flotte zu einer anderen zu wechseln, ohne deinen Spielclient zu ändern. Sie können auch einen Terminalalias erstellen, den Sie auf Inhalte verweisen.

Spielesitzungen durchführen

Nachdem Sie Ihren Gameserver-Build für eine Flotte bereitgestellt haben und Amazon Gameserverprozesse auf jeder Instance GameLift gestartet hat, kann die Flotte Spielesitzungen hosten. Amazon GameLift startet neue Spielesitzungen, wenn Ihr Spiele-Client-Dienst eine Platzierungsanfrage an den Backend-Service oder an Amazon GameLift sendet.

Platzierung der Spielsitzungen und der FleetIQ-Algorithmus

Warteschlangen verwenden den FleetIQ-Algorithmus, um einen verfügbaren Spielserver für eine neue Spielsitzung auszuwählen. Die Hauptkomponente für die Platzierung von Spielesitzungen ist die Amazon-Warteschlange für GameLift Spielesitzungen. Sie weisen einer Warteschlange für Spielsitzungen eine Liste von Flotten zu, die bestimmt, wo in der Warteschlange Spielsitzungen platziert werden können. Weitere Informationen zu Warteschlangen für Spielesessions und wie du sie für dein Spiel gestaltest, findest du unter. Entwerfen Sie eine Warteschlange für Spielsitzungen

Spielerverbindungen zu Spielen

Im Rahmen der Platzierung der Spielsitzung fordert die Warteschlange oder Spielsitzung den ausgewählten Spielserver auf, eine neue Spielsitzung zu starten. Der Spielserver reagiert auf die Aufforderung und meldet Amazon, GameLift wenn er bereit ist, Spielerverbindungen zu akzeptieren. Amazon GameLift übermittelt dann Verbindungsinformationen an den Backend-Service oder den Game-Client-Dienst. Ihre Spielclients verwenden diese Informationen, um sich direkt mit der Spielsitzung zu verbinden und mit dem Spielen zu beginnen.

Skalieren der Flottenkapazität

Wenn eine Flotte aktiv und bereit ist, Spielsitzungen zu veranstalten, kannst du deine Flottenkapazität an die Nachfrage der Spieler anpassen. Wir empfehlen, ein Gleichgewicht zwischen allen neuen Spielern zu finden, die schnell ein Spiel finden und zu viel für ungenutzte Ressourcen ausgeben.

Amazon GameLift bietet ein hocheffektives Tool zur automatischen Skalierung, oder Sie können die Flottenkapazität manuell festlegen. Weitere Informationen finden Sie unter Skalierung der GameLift Amazon-Hosting-Kapazität.

Auto Scaling

Amazon GameLift bietet zwei Methoden der automatischen Skalierung:

- Zielgerichtete automatische Skalierung
- Automatische Skalierung mit regelbasierten Richtlinien

Zusätzliche Skalierungsfunktionen

- Schutz vor Spielsitzungen — Verhindern Sie, dass Amazon GameLift Spielsitzungen beendet, in denen aktive Spieler anwesend sind, während einer heruntergezogenen Veranstaltung.
- Skalierungslimits — Steuern Sie die gesamte Instance-Nutzung, indem Sie Mindest- und Höchstgrenzen für die Anzahl der Instances in einer Flotte festlegen.
- Automatische Skalierung aussetzen — Unterbrechen Sie die automatische Skalierung auf Flottenstandortebene, ohne Ihre Auto-Scaling-Richtlinien zu ändern oder zu löschen.
- Skalierungskennzahlen — Verfolgen Sie den Verlauf der Kapazität und der Skalierungsereignisse einer Flotte.

Amazon überwachen GameLift

Wenn Ihre Flotten in Betrieb sind, GameLift sammelt Amazon eine Vielzahl von Informationen, die Ihnen helfen, die Leistung Ihrer bereitgestellten Spieleserver zu überwachen. Sie können diese Informationen verwenden, um Ihre Nutzung von Ressourcen zu optimieren, Probleme zu beheben und Einblicke in die Aktivitäten der Spieler in Ihren Spielen zu erhalten. Amazon GameLift sammelt Folgendes:

- Flotte, Standort, Spielsitzung und Spielersitzungsdetails

- Nutzungsmetriken
- Integrität der Serverprozesse
- Protokolle der Spielsitzungen

Weitere Informationen zur Überwachung in Amazon GameLift finden Sie unterÜberwachung von Amazon GameLift.

Verwendung anderer AWS Ressourcen

Ihre Spielserver und Anwendungen können mit anderen AWS Ressourcen kommunizieren. Sie können beispielsweise eine Reihe von Webdiensten für die Spielerauthentifizierung oder soziale Netzwerke verwenden. Damit Ihre Spielserver auf AWS Ressourcen zugreifen können, die Sie AWS-Konto verwalten, müssen Sie Amazon ausdrücklich GameLift den Zugriff auf Ihre AWS Ressourcen gestatten.

Amazon GameLift bietet eine Reihe von Optionen für die Verwaltung dieser Art von Zugriff. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

So verbinden sich Spieler mit Spielen

Eine Spielesitzung ist eine Instanz, in der Ihr Spiel auf Amazon läuftGameLift. Um dein Spiel zu spielen, kann ein Spieler entweder eine bestehende Spielsitzung finden und dieser beitreten oder eine neue Spielsitzung erstellen und dieser beitreten. Ein Spieler nimmt teil, indem er eine Spielersitzung für die Spielsitzung erstellt. Wenn die Spielsitzung für Spieler geöffnet ist, GameLift reserviert Amazon einen Slot für den Spieler und stellt Verbindungsinformationen zur Verfügung. Der Spieler kann dann eine Verbindung mit der Spielsitzung herstellen und den reservierten Platz in Anspruch nehmen.

Detaillierte Informationen zum Erstellen und Verwalten von Spielsitzungen und Spielersitzungen mit benutzerdefinierten Spielservern finden Sie unterFüge Amazon GameLift zu deinem Spielclient hinzu. Informationen zum Verbinden von Spielern mit Echtzeitservern finden Sie unterIntegration eines Game-Clients für Realtime Server.

Amazon GameLift bietet verschiedene Funktionen im Zusammenhang mit Spiel- und Spielersitzungen.

Hoste Spielsitzungen auf den besten verfügbaren Ressourcen an mehreren Standorten

Wählen Sie aus mehreren Optionen, wenn Sie konfigurieren, wie Amazon Ressourcen für das Hosten neuer Spielesitzungen GameLift auswählt. Wenn du Flotten an mehreren Orten einsetzt, kannst du Warteschlangen für Spielsitzungen erstellen, die neue Spielsitzungen auf jeder Flotte platzieren, unabhängig vom Standort.

Kontrolliere den Spielerzugriff auf Spielsitzungen

Konfiguriere Spielsitzungen, um Beitrittsanfragen von neuen Spielern zuzulassen oder abzulehnen, unabhängig von der Anzahl der verbundenen Spieler.

Verwende benutzerdefinierte Spiel- und Spielerdaten

Füge benutzerdefinierte Daten zu Spielsitzungsobjekten und Spielersitzungsobjekten hinzu. Amazon GameLift leitet Spielsitzungsdaten an einen Spieleserver weiter, wenn eine neue Spielsitzung gestartet wird. Amazon GameLift leitet Spielersitzungsdaten an den Spielserver weiter, wenn ein Spieler eine Verbindung zur Spielsitzung herstellt.

Filtere und sortiere verfügbare Spielsitzungen

Verwenden Sie die Sitzungssuche und -sortierung, um das bestmögliche Spiel für einen potenziellen Spieler zu finden, oder lassen Sie den Spieler aus einer Liste verfügbarer Spielsitzungen wählen. Verwenden Sie die Sitzungssuche und -sortierung, um Spielsitzungen anhand der Sitzungsmerkmale zu finden. Sie können außerdem entsprechend Ihren eigenen, benutzerdefinierten Spieldaten suchen und sortieren.

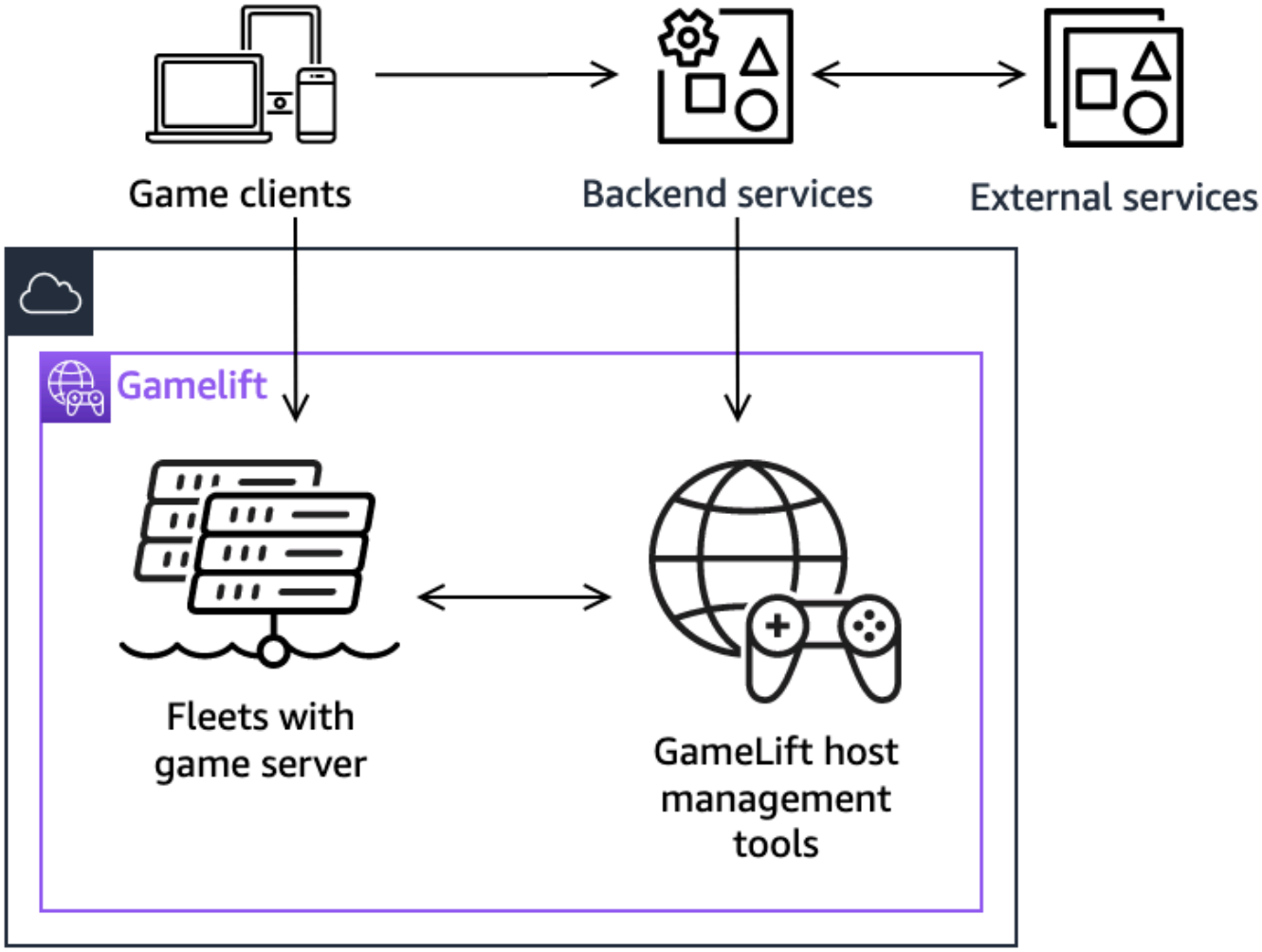
Verfolge die Nutzungsdaten von Spielen und Spielern

Speichern Sie automatisch Protokolle für abgeschlossene Spielsitzungen. Du kannst Logspeicher einrichten, wenn du Amazon GameLift in deine Spieleserver integrierst. Weitere Informationen finden Sie unter Protokollieren von Servernachrichten in Amazon GameLift.

Verwenden Sie die GameLift Amazon-Konsole, um detaillierte Informationen zu Spielsitzungen einzusehen, einschließlich Sitzungsmetadaten, Einstellungen und Spielersitzungsdaten. Weitere Informationen finden Sie unter Daten zu Spiel- und Spielersitzungen anzeigen und Metriken.

Spielarchitektur mit verwaltetem Amazon GameLift

Das folgende Diagramm zeigt die wichtigsten Komponenten einer Spielarchitektur, die mithilfe der verwalteten GameLift Amazon-Lösung gehostet wird.



Zu den wichtigsten Komponenten dieser Architektur gehören:

Spiele-Clients

Um einem auf Amazon gehosteten Spiel beizutreten GameLift, muss dein Spielclient zuerst eine verfügbare Spielsitzung finden. Der Spielclient sucht nach bestehenden Spielsitzungen, fordert Matchmaking an oder startet eine neue Spielsitzung, indem er GameLift über einen Backend-Service mit Amazon kommuniziert. Der Backend-Service stellt Anfragen an Amazon GameLift und als Antwort erhält der Dienst Informationen zur Spielsitzung, die er an den Spielclient zurückleitet. Der Spielclient stellt dann eine Verbindung zum Spieleserver her. Weitere Informationen finden Sie unter Vorbereiten von Spielen für Amazon GameLift.

Backend-Dienste

Ein Backend-Service wickelt die Kommunikation zwischen Spieleclients und Amazon ab, GameLift indem er die Amazon GameLift Service-API-Operationen im AWS SDK aufruft. Sie können Backend-Services auch für andere spielspezifische Aufgaben wie Spielerauthentifizierung und -autorisierung, Inventar oder Währungskontrolle verwenden. Weitere Informationen finden Sie unter Gestalte deinen Spiele-Client-Dienst.

Externe Dienste

Ihr Spiel kann sich auf einen externen Dienst verlassen, z. B. für die Bestätigung einer Abonnement-Mitgliedschaft. Ein externer Dienst kann Informationen über einen Backend-Service und Amazon GameLift an Ihre Spieleserver weitergeben.

Spiel-Server

Sie laden Ihre Gameserver-Software auf Amazon hoch und Amazon stellt sie GameLift dann auf Hosting-Computern bereit GameLift, um Spielsitzungen zu hosten und Spielerverbindungen zu akzeptieren. Spieleserver kommunizieren mit Amazon, GameLift um Spielsitzungen zu starten, neu verbundene Spieler zu validieren und den Status von Spielsitzungen, Spielerverbindungen und verfügbaren Ressourcen zu melden.

Benutzerdefinierte Spieleserver kommunizieren GameLift über das Amazon GameLift Server SDK mit Amazon. Spieleclients stellen eine direkte Verbindung zu einem Spieleserver her, nachdem sie Verbindungsdetails von Amazon GameLift über einen Backend-Service erhalten haben. Weitere Informationen finden Sie unter Integrieren Sie Spiele mit benutzerdefinierten Spieleservern.

Echtzeitserver sind Spieleserver, auf denen Ihr benutzerdefiniertes Skript ausgeführt wird. Wenn Sie einem Spiel beitreten, stellt ein Spielclient mithilfe des Realtime Client SDK eine direkte Verbindung zu einem Realtime-Server her. Weitere Informationen finden Sie unter Integration von Spielen mit Amazon GameLift Realtime Servern.

Tools zur Hostverwaltung

Bei der Einrichtung und Verwaltung von Hosting-Ressourcen verwenden Spielebesitzer Tools zur Hosting-Verwaltung, um Spielserver-Builds oder -Skripte, Flotten, Spielersuche und Warteschlangen zu verwalten. Das GameLift Amazon-Toolset im AWS SDK und in der Konsole bietet Ihnen mehrere Möglichkeiten, Ihre Hosting-Ressourcen zu verwalten. Sie können zur Fehlerbehebung remote auf jeden einzelnen Spieleserver zugreifen.

Einrichtung

Holen Sie sich Hilfe bei der Einrichtung Ihrer Amazon AWS-Konto GameLift zum Hosten Ihrer Multiplayer-Spiele.

> **Tip**
>
> Informationen zum Ausprobieren des GameLift Amazon-Gameserver-Hostings finden Sie unterErste Schritte mit Amazon GameLift.

Themen

Richten Sie ein AWS-Konto

Um mit der Nutzung von Amazon zu beginnen GameLift, erstellen und richten Sie Ihre ein AWS-Konto. Die Erstellung eines ist kostenlos AWS-Konto. In diesem Abschnitt erfahren Sie, wie Sie Ihr Konto erstellen, Ihre Benutzer einrichten und Berechtigungen konfigurieren.

Themen

Melden Sie sich an für ein AWS-Konto

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

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

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

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

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

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

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

Melden Sie sich als Benutzer mit Administratorzugriff an

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

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Benutzerberechtigungen für Amazon verwalten GameLift

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

Sicherheitsmethoden in IAM) sollten Sie allen Benutzern Berechtigungen mit den geringsten Rechten zuweisen. Hinweise zur Syntax von Berechtigungen finden Sie unter. Beispiele für IAM-Genehmigungen für Amazon GameLift

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

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

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

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

Richten Sie den programmatischen Zugriff für Benutzer ein

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

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

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benut zeranmeldedaten im Benutzerhandbuch. AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		CLIAWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Wenn Sie Zugriffsschlüssel verwenden, finden Sie weitere Informationen unter Bewährte Methoden für die Verwaltung von AWS Zugriffsschlüsseln.

Richte den programmatischen Zugriff für dein Spiel ein

Die meisten Spiele verwenden Backend-Dienste, um über die AWS SDKs GameLift mit Amazon zu kommunizieren. Sie verwenden beispielsweise einen Backend-Service (der im Namen von Spieleclients agiert), um Spielsitzungen anzufordern, Spieler in Spiele einzubinden und andere Aufgaben zu erledigen. Diese Dienste benötigen programmatischen Zugriff und Sicherheitsanmeldedaten, um Aufrufe von Amazon GameLift Service APIs zu authentifizieren.

Für Amazon verwalten Sie diesen Zugriff GameLift, indem Sie einen Player-Benutzer in AWS Identity and Access Management (IAM) erstellen. Verwalten Sie die Benutzerberechtigungen von Spielern über eine der folgenden Optionen:

- Erstellen Sie eine IAM-Rolle mit Spielerbenutzerberechtigungen und ermöglichen Sie dem Player-Benutzer, die Rolle bei Bedarf zu übernehmen. Der Backend-Service muss Code enthalten, um diese Rolle zu übernehmen, bevor Anfragen an Amazon GameLift gestellt werden. Gemäß den bewährten Sicherheitsmethoden bieten Rollen begrenzten, temporären Zugriff. Sie können Rollen

für Workloads verwenden, die auf AWS Ressourcen (IAM-Rollen) oder außerhalb von AWS (IAM Roles Anywhere) ausgeführt werden.

- Erstellen Sie eine IAM-Benutzergruppe mit Spielerbenutzerberechtigungen und fügen Sie Ihren Player-Benutzer der Gruppe hinzu. Mit dieser Option erhalten Ihre Spielerbenutzer langfristige Anmeldeinformationen, die der Backend-Service speichern und für die Kommunikation mit Amazon GameLift verwenden muss.

Informationen zur Syntax der Berechtigungsrichtlinien finden Sie unterBeispiele für Benutzerberechtigungen für Spieler.

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

Beispiele für IAM-Genehmigungen für Amazon GameLift

Verwenden Sie die Syntax in diesen Beispielen, um AWS Identity and Access Management (IAM) -Berechtigungen für Benutzer festzulegen, die Zugriff auf GameLift Amazon-Ressourcen benötigen. Weitere Informationen zur Verwaltung von Benutzerberechtigungen finden Sie unterBenutzerberechtigungen für Amazon verwalten GameLift. Bei der Verwaltung von Berechtigungen für Benutzer außerhalb des IAM Identity Center hat es sich bewährt, Berechtigungen immer IAM-Rollen oder Benutzergruppen zuzuordnen, nicht einzelnen Benutzern.

Wenn Sie Amazon GameLift FleetIQ als eigenständige Lösung verwenden, finden Sie weitere Informationen unter Einrichten Ihres AWS-Konto für Amazon FleetIQ. GameLift

Beispiele für Administratorberechtigungen

Diese Beispiele geben einem Benutzer vollen Zugriff auf die Verwaltung der GameLift Amazon-Game-Hosting-Ressourcen.

Example Syntax für GameLift Amazon-Ressourcenberechtigungen

Das folgende Beispiel erweitert den Zugriff auf alle GameLift Amazon-Ressourcen.

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

```
  }
}
```

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

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

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

Example Syntax für GameLift Amazon-Ressourcen und **`PassRole`** -Berechtigungen

Das folgende Beispiel erweitert den Zugriff auf alle GameLift Amazon-Ressourcen und ermöglicht es einem Benutzer, eine IAM-Servicerolle an Amazon GameLift zu übergeben. Eine Servicerolle gibt Amazon GameLift eingeschränkte Möglichkeiten, in Ihrem Namen auf andere Ressourcen und Dienste zuzugreifen, wie unter beschriebenEinrichten einer IAM-Servicerolle für Amazon GameLift. Wenn Amazon beispielsweise auf eine `CreateBuild` Anfrage reagiert, GameLift benötigt es Zugriff auf Ihre Build-Dateien in einem Amazon S3-Bucket. Weitere Informationen zu der *`PassRole`* Aktion finden Sie im IAM-Benutzerhandbuch unter IAM: Eine IAM-Rolle an einen bestimmten AWS Dienst weitergeben.

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

```
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "gamelift.amazonaws.com"
        }
      }
    }
  ]
}
```

Beispiele für Benutzerberechtigungen für Spieler

Diese Beispiele ermöglichen es einem Backend-Service oder einer anderen Entität, API-Aufrufe an die GameLift Amazon-API zu tätigen. Sie decken die gängigen Szenarien für die Verwaltung von Spielsitzungen, Spielersitzungen und Matchmaking ab. Weitere Details finden Sie unter Richte den programmatischen Zugriff für dein Spiel ein.

Example Syntax für die Platzierungsberechtigungen für Spielsitzungen

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

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

```
}
```

Example Syntax für Matchmaking-Berechtigungen

Das folgende Beispiel erweitert den Zugriff auf die GameLift Amazon-APIs, die FlexMatch Matchmaking-Aktivitäten verwalten. FlexMatchbringt Spieler für neue oder bestehende Spielsitzungen zusammen und initiiert die Platzierung von Spielsitzungen für Spiele, die bei Amazon GameLift gehostet werden. Weitere Informationen zu FlexMatch finden Sie unter Was ist Amazon GameLiftFlexMatch?

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

Example Syntax für die manuelle Platzierung von Spielsitzungen

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

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForManualGameSessions",
    "Effect": "Allow",
    "Action": [
      "gamelift:CreateGameSession",
      "gamelift:DescribeGameSessions",
```

```
      "gamelift:SearchGameSessions",
      "gamelift:CreatePlayerSession",
      "gamelift:CreatePlayerSessions",
      "gamelift:DescribePlayerSessions"
    ],
    "Resource": "*"
  }
}
```

Einrichten einer IAM-Servicerolle für Amazon GameLift

Einige Amazon- GameLift Funktionen erfordern, dass Sie den eingeschränkten Zugriff auf AWS Ressourcen erweitern, die Sie besitzen. Erstellen Sie dazu eine AWS Identity and Access Management (IAM)-Rolle. Eine IAM-Rolle ist eine IAM-Identität, die Sie in Ihrem Konto mit bestimmten Berechtigungen erstellen können. Eine IAM-Rolle ist einem IAM-Benutzer ähnlich, weil es sich um eine AWS-Identität mit Berechtigungsrichtlinien handelt, die festlegen, welche Aktionen die Identität in AWS ausführen kann und welche nicht. Eine Rolle ist jedoch nicht einer einzigen Person zugeordnet, sondern kann von allen Personen angenommen werden, die diese Rolle benötigen. Einer Rolle sind außerdem keine standardmäßigen, langfristigen Anmeldeinformationen (Passwörter oder Zugriffsschlüssel) zugeordnet. Wenn Sie eine Rolle übernehmen, erhalten Sie stattdessen temporäre Anmeldeinformationen für Ihre Rollensitzung.

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

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

Erstellen der benutzerdefinierten IAM-Rolle

Schritt 1: Erstellen einer Berechtigungsrichtlinie.

So verwenden Sie den JSON-Richtlinieneditor zum Erstellen einer Richtlinie

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

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

3. Wählen Sie oben auf der Seite Create policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben oder fügen Sie ein JSON-Richtliniendokument ein. Weitere Informationen zur IAM-Richtliniensprache finden Sie in der IAM-JSON-Richtlinienreferenz.
6. Beheben Sie alle Sicherheitswarnungen, Fehler oder allgemeinen Warnungen, die während der Richtlinien-Validierung erzeugt wurden, und wählen Sie dann Weiter.

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

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

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

Schritt 2: Erstellen Sie eine Rolle, die Amazon übernehmen GameLift kann.

1. Klicken Sie im Navigationsbereich der IAM-Konsole auf Rollen, und wählen Sie dann Rolle erstellen.
2. Wählen Sie auf der Seite Vertrauenswürdige Entität auswählen die Option Benutzerdefinierte Vertrauensrichtlinie aus. Diese Auswahl öffnet den Editor Benutzerdefinierte Vertrauensrichtlinie.
3. Ersetzen Sie die Standard-JSON-Syntax durch Folgendes und wählen Sie dann Weiter, um fortzufahren.

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

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

Syntax der Berechtigungsrichtlinie

- Berechtigungen für Amazon GameLift , die Servicerolle zu übernehmen

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

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

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

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

Entwicklungsunterstützung mit Amazon GameLift

Amazon GameLift bietet eine Reihe von SDKs, die Sie mit Ihren verwalteten Game-Hosting-Lösungen verwenden können. Verwenden Sie Amazon GameLift SDKs, um Multiplayer-Spieleservern, Spielclients und Spieledienste, die mit dem GameLift Amazon-Hosting-Service interagieren müssen, die erforderlichen Funktionen hinzuzufügen.

Aktuelle Informationen zu Amazon GameLift SDK-Versionen und SDK-Kompatibilität finden Sie unter GameLift Versionshinweise von Amazon.

Für benutzerdefinierte Spieleserver

Erstellen und implementieren Sie benutzerdefinierte 64-Bit-Spieleserver mit dem Amazon GameLift Server SDK. Spieleserver, die in das Server-SDK integriert und für das Hosting bereitgestellt werden,

können mit dem GameLift Amazon-Service kommunizieren, um Spielsitzungen zu starten und zu verwalten. Informationen zur Integration des Server-SDK finden Sie in den Themen unterVorbereiten von Spielen für Amazon GameLift.

Betriebssysteme für die Entwicklung

- Windows
- Linux

Unterstützte Programmiersprachen

Amazon GameLift stellt das Server-SDK für die folgenden Sprachen bereit. Laden Sie Server-SDKs herunter. Detaillierte versionsspezifische Informationen finden Sie in den Readme-Dateien, die in jedem Paket enthalten sind.

- C++-Server-SDK
  - SDK-Referenz
  - SDK-Integration
- C#-Server-SDK (Versionen unterstützen möglicherweise .NET 4 und .NET 6)
  - SDK-Referenz
  - SDK-Integration
- Go
  - SDK-Referenz
  - SDK-Integration

Unterstützte Spiele-Engines

Verwenden Sie sprachspezifische SDKs mit allen Engines, die C++-, C#- oder Go-Bibliotheken unterstützen. Darüber hinaus GameLift stellt Amazon diese Game Engine-Plugins zur Verfügung: GameLift Amazon-Plugins herunterladen

- Unity
  - Das C#-Server-SDK-Plugin für Unity ist ein leichtes Plugin mit vorgefertigten Bibliotheken, die Sie mit dem Unity-Paketmanager installieren können. Verwenden Sie dieses Plugin mit den folgenden Unity-Versionen: 2020.3 LTS, 2021.3 LTS und 2022.3 LTS für Windows und Mac OS.

Es unterstützt die Profile .NET Framework und .NET Standard von Unity mit.NET Standard 2.1 und .NET 4.x.

  - Integrieren Sie Amazon GameLift in ein Unity-Projekt

- Das eigenständige Plugin für Unity 2021.3 LTS und 2022.3 LTS ist ein voll ausgestattetes Plugin mit den für Unity entwickelten C#-SDK-Bibliotheken und GUI-Elementen für die Konfiguration und Bereitstellung von Amazon-Ressourcen für das Hosting. GameLift
  - Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x
  - Amazon GameLift Server-SDK-Referenz für C#

- Unreal Engine
  - Das C++-Server-SDK-Plugin für Unreal ist ein leichtes Plugin, das aus C++-Unreal-Quellcode besteht, den Sie in Bibliotheken für die Verwendung mit den Unreal Engine-Versionen 4, 5 und 5.1 integrieren können.
    - Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt
    - Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x
  - Das eigenständige Plugin für Unreal Engine 5.0, 5.1 und 5.2 ist ein Plugin mit vollem Funktionsumfang, das die SDK-Bibliotheken und das SDK für den C++ for Unreal-Server enthält. AWS Das Plugin ist im Unreal-Editor installiert und enthält UI-Elemente und unterstützendes Material für die Konfiguration und Bereitstellung von GameLift Amazon-Ressourcen für das Hosting.
    - Integrieren von Spielen mit dem Amazon GameLift -Plugin für Unreal Engine
    - Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x

Betriebssyssteme für Spiel-Server

Verwenden Sie das Amazon GameLift Server SDK, um Spieleserver für die folgenden Plattformen zu erstellen:

- Windows Server 2016
- Amazon Linux 2023
- Amazon Linux 2 (AL2)
- Windows Server 2012 (siehe GameLift Häufig gestellte Fragen zu Amazon für Windows 2012)
- Amazon Linux (AL1) (siehe GameLift Häufig gestellte Fragen zu Amazon für AL1)

Für kundenspezifische Kundenservices

Erstellen Sie benutzerdefinierte 64-Bit-Client-Services mithilfe des AWS SDK mit der GameLift Amazon-API. Dieses SDK ermöglicht es Kundendiensten, Spielsitzungen zu verwalten und Spieler mit Spielen zu verbinden, die auf Amazon gehostet werden GameLift. Laden Sie zunächst das AWS SDK herunter. Weitere Informationen zur Verwendung des SDK mit Amazon GameLift finden Sie in der Amazon GameLift API-Referenz.

Für Realtime-Server

Konfigurieren und implementieren Sie Echtzeitserver, um Ihre Multiplayer-Spiele zu hosten. Verwenden Sie das Amazon Realtime Client SDK, damit Ihre Spieleclients eine Verbindung zu GameLift Realtime-Servern herstellen können. Spieleclients verwenden dieses SDK, um Nachrichten mit einem Realtime-Server und mit anderen Spieleclients auszutauschen, die eine Verbindung zum Server herstellen. Laden Sie zunächst das Amazon GameLift Realtime Client SDK herunter. Informationen zur Konfiguration finden Sie unterIntegration eines Game-Clients für Realtime Server.

SDK -Unterstützung

Das Realtime Client SDK enthält Quelltexte für die folgenden Sprachen:

- C# (.NET)

Entwicklungsumgebungen

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

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

Betriebssyssteme für Spiel-Server

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

- Amazon Linux
- Amazon Linux 2

Verwalten der Kosten für das Hosten von Spielen

Ihre AWS Rechnung spiegelt Ihre Kosten für das Hosting von Spielen wider. Sie können die geschätzten Gebühren für den aktuellen Monat und die endgültigen Gebühren für die vorherigen Monate in der Fakturierungskonsole unter https://console.aws.amazon.com/billing/ einsehen. Weitere Informationen zu Tools und Ressourcen, die Sie bei der Verwaltung Ihrer AWS Kosten unterstützen, finden Sie im AWS Billing -Benutzerhandbuch. Dieser Leitfaden kann Ihnen helfen, Ihren Ressourcenverbrauch zu überprüfen, die zukünftige Nutzung festzulegen und Ihre Skalierungsanforderungen zu ermitteln.

Berücksichtigen Sie insbesondere diese Tipps, um die Kosten von Amazon- GameLift Services zu verwalten.

Erstellen von Abrechnungswarnungen zur Überwachung der Nutzung

Richten Sie eine Warnung zur Nutzung des AWS kostenlosen Kontingents für ein, um Sie zu benachrichtigen, wenn sich Ihre Nutzung den Grenzwerten des kostenlosen Kontingents für für Amazon GameLift und andere nähert oder diese überschreitetAWS-Services. Sie können die Warnungen so konfigurieren, dass sie basierend auf Ihrem Nutzungsniveau Maßnahmen ergreifen. Sie können Ihr Budget beispielsweise automatisch auf Null setzen, wenn Ihr ein Limit für das kostenlose Kontingent erreicht.

Sie können auch Amazon- CloudWatch Fakturierungswarnungen festlegen, um Benachrichtigungen zu erhalten, wenn die Nutzung benutzerdefinierte Schwellenwerte erreicht.

Weitere Informationen finden Sie unter diesen Themen im AWS Billing -Benutzerhandbuch:

- Nachverfolgen Ihrer Nutzung des AWS kostenlosen Kontingents für
- Präferenzen für Fakturierungswarnungen

Nachverfolgen der Kosten pro Amazon- GameLift Flotte

Verwenden Sie AWS Kostenzuordnungs-Tags, um Ihre Spiel-Hosting-Kosten basierend auf GameLift Amazon EC2-Flotten und anderen EC2-Ressourcen zu organisieren und zu verfolgen. Indem Sie Ihre Flotten einzeln oder nach Gruppen markieren, können Sie Kostenzuordnungsberichte erstellen, die die Kosten basierend auf dem zugewiesenen Tag kategorisieren. Sie können diese Art von Bericht verwenden, um zu ermitteln, wie Flotten zu Ihren Hosting-Kosten beitragen. Sie können Tags auch verwenden, um Ansichten im AWS Cost Explorer zu filtern.

Weitere Informationen enthalten die folgenden Themen:

- Verwenden von AWS Kostenzuordnungs-Tags, AWS BillingBenutzerhandbuch
- Analysieren Ihrer Kosten mit AWS Cost Explorer , AWS Cost ManagementBenutzerhandbuch

Setzen Sie ungenutzte Flottenkapazität auf Null

Flotten können auch dann weiterhin Kosten verursachen, wenn sie keine Hosting-Spielsitzungen verwenden. Um unnötige Gebühren zu vermeiden, skalieren Sie Ihre Flotte auf Null, wenn sie nicht verwendet wird. Wenn Sie Auto Scaling verwenden, setzen Sie diese Aktivität aus und legen Sie die Flottenkapazität manuell fest.

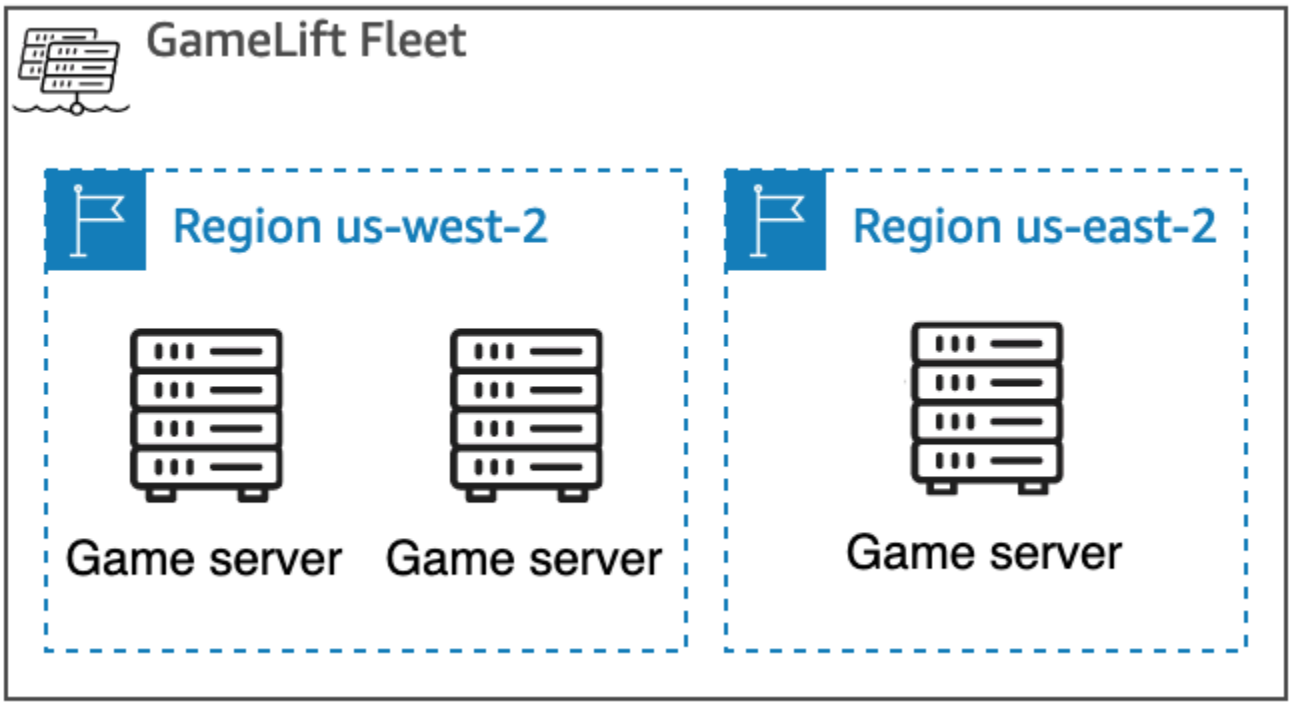
Amazon GameLift -Hosting-Standorte

Amazon GameLift ist in mehreren AWS-Regionen und Local Zones verfügbar. Eine vollständige Liste der Standorte finden Sie unter Amazon GameLift-Endpunkte und -Kontingente im Allgemeine AWS-Referenz.

Amazon GameLift -Hosting

Wenn Sie eine Amazon- GameLift Flotte erstellen, GameLift erstellt Amazon die Ressourcen der Flotte in Ihrem aktuellen AWS-Region. Amazon GameLift ruft diese Region als Heimatregion der Flotte auf. Um eine Flotte zu verwalten, greifen Sie von ihrer Heimatregion aus darauf zu.

Flotten mit mehreren Standorten stellen Instances zusätzlich zur Heimatregion der Flotte an anderen Standorten bereit. Mit Flotten mit mehreren Standorten können Sie die Kapazität für jeden Standort einzeln verwalten und Spielsitzungen nach Standort platzieren. Flotten mit mehreren Standorten können Remote-Standorte in jeder Region oder lokalen Zone haben, die Amazon GameLift unterstützt. Das folgende Diagramm zeigt eine Flotte mit mehreren Standorten mit Ressourcen in zwei Regionen. Im Diagramm enthält die `us-west-2` Region zwei Spieleserver und die `us-east-2` Region hat einen Spieleserver.



Wenn Sie eine Flotte mit mehreren Standorten mit Instances in Regionen verwenden möchten, die nicht standardmäßig aktiviert sind, müssen Sie diese Regionen in Ihrem aktivierenAWS-Konto. Außerdem muss Ihre Amazon GameLift -Administratorrichtlinie die `ec2:DescribeRegions` Aktion zulassen. Weitere Informationen zu Regionen, die nicht standardmäßig aktiviert sind, und wie Sie sie aktivieren können, finden Sie unter Verwalten AWS-Regionen von im Allgemeine AWS-Referenz. Ein Richtlinienbeispiel mit Regionen, die standardmäßig nicht aktiviert sind, finden Sie unter Beispiele für Administratorberechtigungen.

> **Important**
>
> Um Regionen zu verwenden, die standardmäßig nicht aktiviert sind, aktivieren Sie sie in Ihrem AWS-Konto.
>
> - Flotten mit Regionen, die nicht aktiviert sind und die Sie vor dem 28. Februar 2022 erstellt haben, sind davon nicht betroffen.
> - Um neue Flotten mit mehreren Standorten zu erstellen oder vorhandene Flotten mit mehreren Standorten zu aktualisieren, aktivieren Sie zunächst alle Regionen, die Sie verwenden möchten.

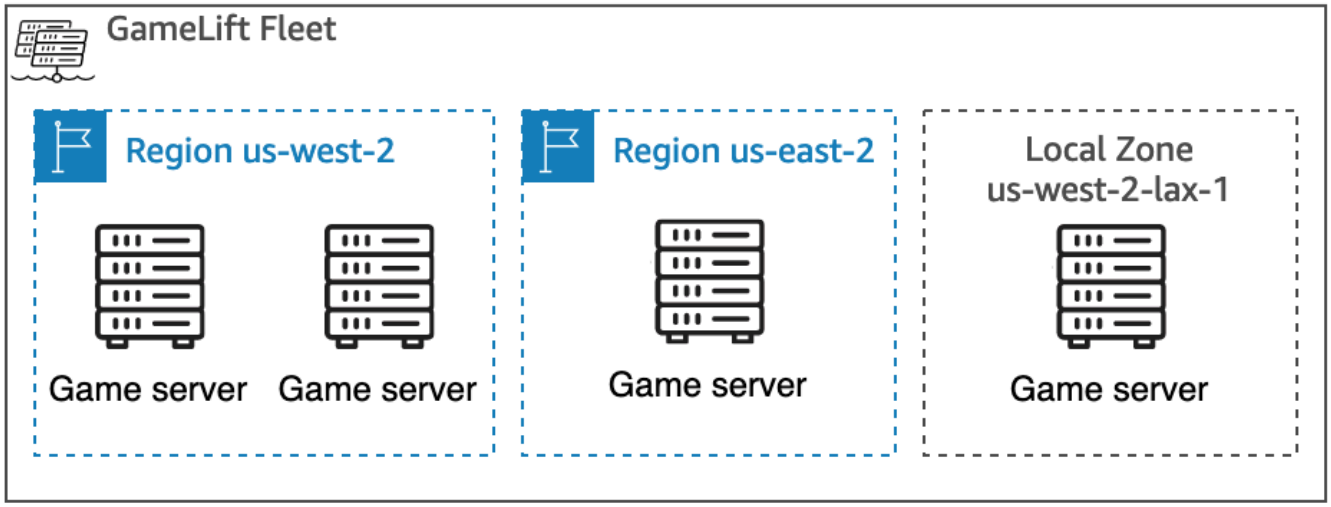
Für die Platzierung von Spielsitzungen können Sie Spielsitzungswarteschlangen an jedem von Amazon GameLift unterstützten Ort erstellen. Amazon GameLift platziert Spielsitzungen von dem Ort aus, an dem Sie die Warteschlange erstellt haben.

Local Zones

Eine Local Zone ist eine Erweiterung eines AWS-Region in geografischer Nähe zu Ihren Benutzern. Local Zones haben ihre eigenen Verbindungen zum Internet. Local Zones unterstützen auch , AWS Direct Connect sodass Ressourcen, die in einer Local Zone erstellt wurden, lokale Benutzer mit Kommunikation mit niedriger Latenz bedienen können. Weitere Informationen finden Sie unter AWS-Local-Zones.

Der Code für eine Local Zone ist ihr Regionscode, gefolgt von einer Kennung, die ihren physischen Standort angibt. Die `us-west-2-lax-1` Local Zone befindet sich beispielsweise in Los Angeles. Eine Liste der verfügbaren Local Zones finden Sie unter Verfügbare Local Zones.

Amazon GameLift hostet Ihre Spiele an jedem der Standorte, die Sie für Ihre Flotte auswählen. Das folgende Diagramm zeigt eine Flotte mit zwei Spieleservern in der `us-west-2` Region, einem Spieleserver in der `us-east-2` Region und einem Spieleserver in der `us-west-2-lax-1` Local Zone.



Verfügbare Local Zones

In der folgenden Tabelle sind die verfügbaren Local Zones und ihre physischen Standorte aufgeführt.

Lokale Zone	Standort (Metropolregion)	
us-east-1-atl-1	Atlanta	
us-east-1-chi-1	Chicago	
us-east-1-dfw-1	Dallas	
us-east-1-iah-1	Houston	
us-east-1-mci-1	Kansas City	
us-west-2-den-1	Denver	
us-west-2-lax-1	Los Angeles	
us-west-2-phx-1	Phoenix	

Amazon GameLift Anywhere

Sie können Amazon GameLift Anywhere verwenden, um Flotten mit Ihrer eigenen Hardware zu erstellen und Ihre Spiele-Builds, Skripts, Spieleserver und Clients mit Amazon zu verwalten GameLift. Amazon GameLift Anywhere ist in allen Regionen verfügbar, die Amazon GameLift unterstützt. Weitere Informationen zum Erstellen einer -AnywhereFlotte und zum Testen Ihrer Spieleserver-Integration finden Sie unter Erstellen Sie eine GameLift Anywhere Amazon-Flotte und Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten.

Mit Amazon GameLift Anywhere erstellen Sie benutzerdefinierte Standorte, die den physischen Standort der Hardware darstellen, die Sie zum Hosten Ihrer GameLift integrierten Amazon-Spielserver verwenden.

Amazon GameLift FlexMatch

Für können FlexMatchSie Match-generierte Spielsitzungen an jedem von Amazon GameLift unterstützten Ort hosten. Tatsächliche Matchmaking-Aktivitäten finden in der stattAWS-Region, in der Sie sich für die Erstellung Ihrer Matchmaker-Ressourcen entschieden haben. Amazon- GameLift Routen ordnen Anfragen dem Matchmaker zu und verarbeiten sie an diesem Standort. Weitere Informationen zu Amazon GameLift FlexMatchfinden Sie unter Was ist Amazon GameLift FlexMatch?

AWS-Regionen , die - FlexMatch Ressourcen unterstützen

Amazon GameLift in China

Wenn Sie Amazon GameLift für Ressourcen in der Region China (Peking), betrieben von Sinnet, oder der Region China (Ningxia), betrieben von NWCD, verwenden, müssen Sie über ein separates AWS (China)-Konto verfügen. Beachten Sie, dass einige Funktionen in den China-Regionen nicht verfügbar sind. Weitere Informationen zur Verwendung von Amazon GameLift in diesen Regionen finden Sie in den folgenden Ressourcen:

- Amazon Web Services in China
- Amazon GameLift (Erste Schritte mit Amazon Web Services in China)

Erste Schritte mit Amazon GameLift

Wir empfehlen Ihnen, die folgenden Beispiele auszuprobieren, bevor Sie Amazon GameLift für Ihr eigenes Spiel verwenden. Das Beispiel für einen benutzerdefinierten Gameserver bietet Ihnen Erfahrung mit dem Hosting von Spielen in der GameLift Amazon-Konsole. Das Beispiel für Echtzeitserver zeigt dir, wie du ein Spiel mithilfe von Realtime Servern für das Hosten vorbereitest.

Informationen zu den ersten Schritten mit Amazon GameLift für Ihr eigenes Spiel finden Sie unterRoadmap für GameLift verwaltetes Hosting von Amazon.

Beispiel für einen benutzerdefinierten Gameserver

Dieses Beispiel zeigt ein benutzerdefiniertes Live-Spiel auf AmazonGameLift. Das Beispiel führt Sie durch die folgenden Schritte:

- Erstellen eines Beispiel-Game-Builds.
- Erstellung einer Flotte für den Betrieb des Spielservers.
- Über den Beispiel-Spielclient wird eine Verbindung zum Spieleserver hergestellt.
- Überprüfung der Flotten- und Spielsitzungsmetriken.

Nach diesen Schritten kannst du mehrere Spielclients starten und das Spiel spielen, um Hosting-Daten zu generieren. Anschließend können Sie die GameLift Amazon-Konsole erkunden, um Ihre Hosting-Ressourcen einzusehen, Kennzahlen zu verfolgen und mit Möglichkeiten zur Skalierung der Hosting-Kapazität zu experimentieren.

Melden Sie sich zunächst bei der GameLiftAmazon-Konsole an.

Realtime Servers — Beispielspiel

Dieses Beispiel ist ein komplettes Multiplayer-Spiel namens Mega Frog Race, das den Quellcode enthält. Das Beispiel zeigt, wie du deinen Spielclient in Realtime Server integrierst. Sie können dieses Beispielspiel auch als Ausgangspunkt verwenden, um mit anderen GameLift Amazon-Funktionen zu experimentieren, wie FlexMatch z.

Ein praktisches Tutorial findest du im for Games-Blog unter Server für Multiplayer-Handyspiele mit nur wenigen Zeilen erstellen. JavaScript AWS

Den Quellcode von Mega Frog Race finden Sie im GitHubRepository.

Der Quellcode umfasst die folgenden Teile:

- Spielclient — Ein Quellcode für den C++-Spieleclient, erstellt in Unity. Der Spielclient erhält Verbindungsinformationen zur Spielsitzung, stellt eine Verbindung zum Server her und tauscht Updates mit anderen Spielern aus.
- Backend-Service — Ein Quellcode für eine AWS Lambda Funktion, die direkte API-Aufrufe an Amazon GameLift verwaltet.
- Echtzeitskript — Eine Quellskriptdatei, die eine Flotte von Echtzeitservern für das Spiel konfiguriert. Dieses Skript enthält die Mindestkonfiguration, die für Echtzeitserver erforderlich ist, um mit Amazon zu kommunizieren GameLift und Spiele zu hosten.

Roadmap für GameLift verwaltetes Hosting von Amazon

Dieses Thema hilft Ihnen bei der Auswahl der verschiedenen GameLift Amazon-Hosting-Optionen für Ihr sitzungsbasiertes Multiplayer-Spiel. In den restlichen Themen in diesem Abschnitt erfahren Sie, wie Sie Amazon GameLift für Ihr verwaltetes Hosting verwenden können.

Bevor Sie sich darauf vorbereiten, Ihr Spiel in die Produktion zu bringen, füllen Sie den Fragebogen zur Markteinführung aus, um mit dem GameLift Amazon-Team zusammenzuarbeiten.

Themen

Wählen Sie eine Hosting-Option

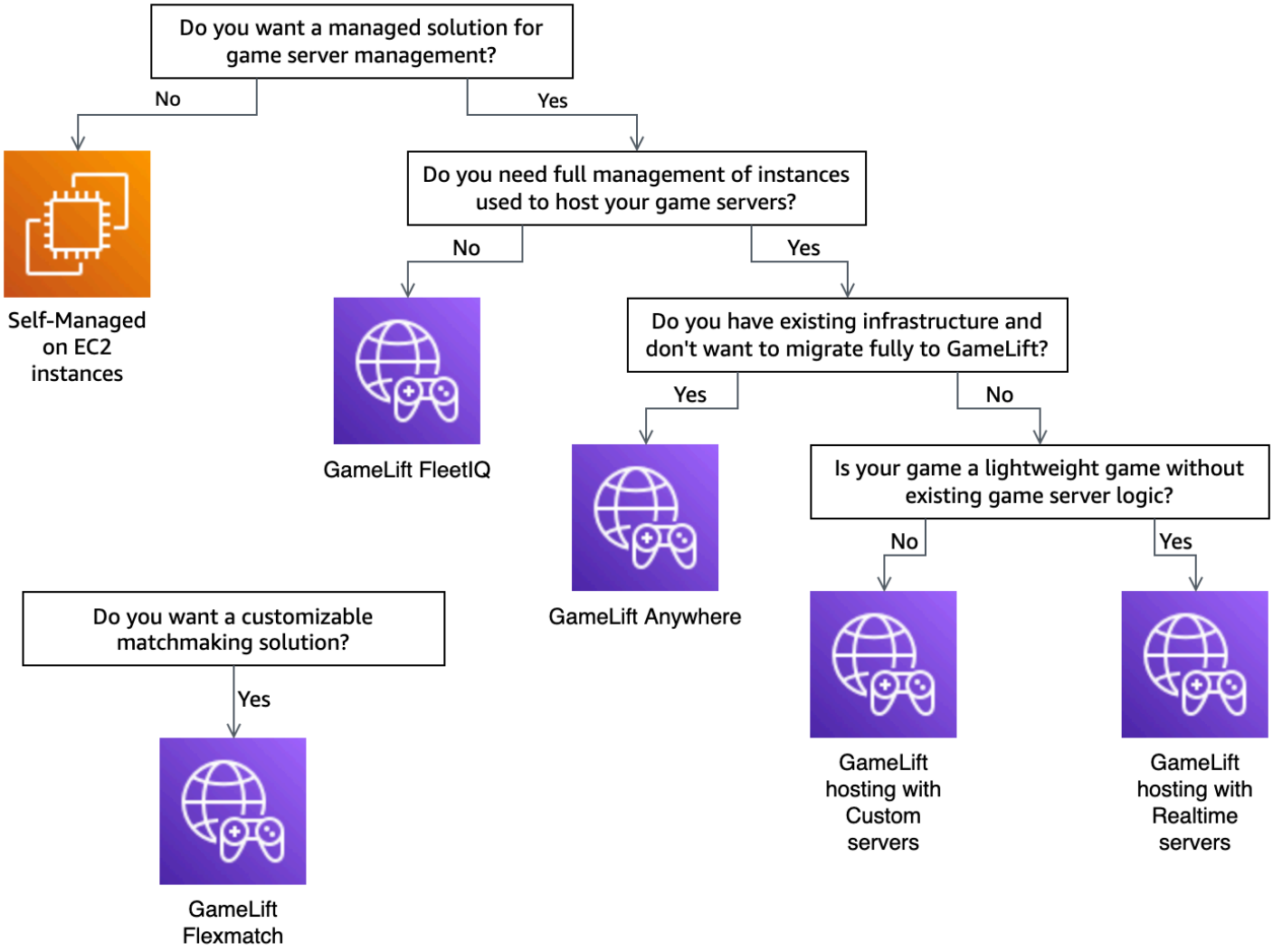
Das folgende Flussdiagramm enthält Fragen, die Sie zur richtigen GameLift Amazon-Lösung für Ihren Anwendungsfall führen sollen.

1. Möchten Sie eine verwaltete Lösung für die Verwaltung von Gameservern?
   - Ja — Fahren Sie mit Schritt zwei fort.
   - Nein — Ziehen Sie selbstverwaltete Spieleserver auf Amazon EC2-Instances in Betracht.
2. Benötigst du die volle Kontrolle über die Instanzen, die deine Spielserver hosten?
   - Ja — ziehen Sie Amazon GameLift FleetIQ in Betracht.
   - Nein — Fahren Sie mit Schritt 3 fort.
3. Haben Sie eine bestehende Infrastruktur, die Sie mit Amazon nutzen möchtenGameLift?
   - Ja — ziehen Sie Amazon in Betracht GameLiftAnywhere.

- Nein — Fahren Sie mit Schritt vier fort.

4. Ist dein Spiel ohne bestehende Spielserverlogik leichtgewichtig?

   - Ja — Ziehen Sie Echtzeitserver in Betracht.
   - Nein — Ziehen Sie benutzerdefinierte Server in Betracht.



Hier finden Sie weitere Informationen zu einigen der im Flussdiagramm genannten GameLift Amazon-Hosting-Optionen:

Amazonas GameLift Anywhere

Verwenden Sie Amazon GameLiftAnywhere, um Ihre Spiele auf Ihrer eigenen Hardware zu hosten und dabei die GameLift Amazon-Verwaltungstools zu nutzen. Du kannst Anywhere Flotten

auch verwenden, um deine Spielserver iterativ zu testen. Weitere Informationen finden Sie unter Erstellen Sie eine GameLift Anywhere Amazon-Flotte.

Verwaltetes Amazon GameLift

Es gibt zwei Optionen für verwaltetes GameLift Amazon-Hosting:

Benutzerdefinierte Server — Amazon GameLift hostet Ihren benutzerdefinierten Server, auf dem Ihre Gameserver-Binärdatei ausgeführt wird.

Echtzeitserver — Amazon GameLift hostet Ihren leichten Gameserver.

Amazon GameLift FleetIQ

Im Flussdiagramm bezieht sich eine Lift-and-Shift-Migration auf eine Migration, bei der Sie keine Änderungen an der Spielarchitektur vornehmen können. Die Nutzung von Amazon GameLift FleetIQ erfordert weniger Änderungen an Ihrer bestehenden Bereitstellung und bietet GameLift Amazon-Tools für das Flottenmanagement. Weitere Informationen finden Sie im Amazon GameLift FleetIQ Developer Guide.

Wenn Sie sich für Amazon GameLift Anywhere oder Managed Amazon entscheidenGameLift, fahren Sie fortBereite dein Spiel für Amazon vor GameLift.

Bereite dein Spiel für Amazon vor GameLift

In diesem Thema werden die Schritte beschrieben, mit denen Sie Ihr Multiplayer-Spiel für die Integration mit verwaltetem GameLift Amazon-Hosting vorbereiten. Um Ihr Spiel vorzubereiten, müssen Sie die Kommunikation zwischen dem Spiel und Amazon aktivierenGameLift.

Bereite deinen eigenen Gameserver vor

Um Spielsitzungen zu starten und zu beenden und um andere Aufgaben ausführen zu können, muss ein Gameserver Amazon GameLift über seinen Status informieren können. Um die Kommunikation mit Amazon zu aktivierenGameLift, fügen Sie Ihrem Gameserver-Projekt Code hinzu. Weitere Informationen finden Sie unter Integrieren Sie Spiele mit benutzerdefinierten Spieleservern.

1. Bereite deinen benutzerdefinierten Gameserver für das Hosting bei Amazon vorGameLift.

   - Holen Sie sich das Amazon GameLift Server SDK und erstellen Sie es für Ihre bevorzugte Programmiersprache und Game-Engine.

- Fügen Sie Ihrem Gameserver-Projekt Code hinzu, um die Kommunikation mit Amazon zu aktivierenGameLift.

2. Bereite deinen Spielclient so vor, dass er eine Verbindung zu von Amazon GameLift gehosteten Spielesitzungen herstellt.

   - Füge das AWS SDK zu deinem Backend-Service und deinem Game-Client-Projekt hinzu. Weitere Informationen finden Sie unter Amazon GameLift SDKs für Client Services herunterladen.
   - Füge Funktionen hinzu, um Informationen über Spielsitzungen abzurufen, neue Spielsitzungen zu platzieren und Platz für Spieler in einer Spielsitzung zu reservieren.
   - (Optional) FlexMatch Für Spielersuche verwenden. Weitere Informationen finden Sie unter FlexMatchIntegration mit GameLift Amazon-Hosting.

Bereite deinen Realtime-Server vor

Amazon GameLift Realtime Servers bietet eine leichte Serverlösung, die Sie an Ihr Spiel anpassen können. Ein Echtzeitserver bietet dieselben Vorteile, die Amazon für Spieleserver GameLift bietet, jedoch mit reduzierter Anpassungsfähigkeit des Spielservers.

Erstellen Sie ein Echtzeit-Skript für das Hosting bei AmazonGameLift.

Echtzeitskripte enthalten Ihre Serverkonfiguration und optionale benutzerdefinierte Spiellogik. Echtzeitserver sind so konzipiert, dass sie Spielsitzungen starten und beenden, Spielerverbindungen akzeptieren und die Kommunikation mit Amazon GameLift und zwischen Spielern in einem Spiel verwalten. Es gibt auch Hooks, mit denen Sie benutzerdefinierte Serverlogik für Ihr Spiel hinzufügen können. Echtzeitserver verwenden Node.js und. JavaScript Weitere Informationen erhalten Sie unter Erstellen eines Echtzeit-Skripts und Testen Sie Ihre Integration mit Amazon GameLift.

Testen Sie Ihre Integration mit Amazon GameLift

Amazon GameLift unterstützt schnelle Iterationen beim Testen Ihrer Spieleserver. Dieses Thema führt Sie durch die verfügbaren Testarten.

Benutzerdefinierte Spielserver

Verwenden Sie AmazonGameLift, um Hardware überall in Ihrer Umgebung in Ihre GameLift Amazon-Game-Hosting-Architektur zu integrieren. Amazon GameLift Anywhere registriert Ihre

Hardware bei Amazon GameLift in einer Anywhere Flotte, sodass Sie sie mit Ihrem eigenen lokalen Entwicklungscomputer testen können. Weitere Informationen zum Testen mit Amazon GameLift Anywhere finden Sie unterTesten Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten. Weitere Informationen zur Verwendung von Amazon GameLift Anywhere für das Hosten Ihrer Spiele mit lokalen Lösungen finden Sie unterAuswahl von GameLift Amazon-Rechenressourcen.

Server in Echtzeit

Mit Realtime Servern können Sie Ihre Skripte jederzeit aktualisieren. Wenn Sie ein Realtime-Skript aktualisieren, GameLift verteilt Amazon die neue Version innerhalb weniger Minuten an Ihre Hosting-Ressourcen. Nachdem Amazon das neue Skript GameLift bereitgestellt hat, verwenden alle neuen Spielsitzungen die neue Skriptversion. Nachdem Amazon das neue Skript GameLift bereitgestellt hat, können Sie sofort mit dem Testen beginnen. Weitere Informationen zu Echtzeitservern finden Sie unter Integration von Spielen mit Amazon GameLift Realtime Servern

Planen und implementieren Sie Ihre GameLift Amazon-Ressourcen

Verwenden Sie die folgenden Tipps, um Ihren globalen Einsatz von GameLift Amazon-Ressourcen zu planen. Informationen darüber, wo Sie Ihre Spiele bei Amazon hosten könnenGameLift, finden Sie unterAmazon GameLift -Hosting-Standorte.

Führen Sie die folgenden Aufgaben aus, um Ihre GameLift Amazon-Ressourcen bereitzustellen:

- Verpacke deinen Gameserver und lade ihn auf Amazon GameLift oder auf deine Hardware hoch. Wenn Sie Ihren Server auf Amazon hochladenGameLift, laden Sie ihn nur auf die Startseite AWS-Region Ihrer Flotte hoch. Amazon verteilt den Server GameLift automatisch an andere Standorte in der Flotte. Weitere Informationen finden Sie unter Upload von Builds und Skripten auf Amazon GameLift.
- Entwerfen und implementieren Sie eine GameLift Amazon-Flotte für Ihr Spiel. Legen Sie fest, welche Art von Computerressourcen verwendet werden sollen, an welchen Standorten die Bereitstellung erfolgen soll, ob Warteschlangen verwendet werden sollen, und weitere Optionen. Weitere Informationen finden Sie unter Leitfaden zur GameLift Amazon-Flottenplanung.
- Erstelle Warteschlangen, um die Platzierungen neuer Spielsitzungen und Strategien für Spot-Instances zu verwalten. Weitere Informationen finden Sie unter Entwerfen Sie eine Warteschlange für Spielsitzungen.
- Nutze automatische Skalierung, um die Hosting-Kapazität deiner Flotte entsprechend der erwarteten Spielernachfrage zu verwalten. Weitere Informationen finden Sie unter Skalierung der GameLift Amazon-Hosting-Kapazität.

- Verwenden Sie FlexMatch Matchmaking-Regeln für Ihr Spiel. Weitere Informationen finden Sie unter FlexMatchIntegration mit GameLift Amazon-Hosting.

Stellen Sie Ihre GameLift Amazon-Ressourcen automatisch bereit

Um die globale Bereitstellung Ihrer GameLift Amazon-Ressourcen zu optimieren, empfehlen wir, dass Sie Infrastructure as Code (IaC) verwenden, um die Ressourcen zu definieren. Da Amazon AWS CloudFormation Vorlagen GameLift unterstützt, können Sie in den Vorlagen Parameter für alle bereitstellungsspezifischen Konfigurationen festlegen.

Um die Bereitstellung Ihrer AWS CloudFormation Stacks zu verwalten, empfehlen wir außerdem die Verwendung von Tools und Diensten für kontinuierliche Integration und kontinuierliche Bereitstellung (CI/CD) wie. AWS CodePipeline Diese helfen Ihnen bei der automatischen oder genehmigten Bereitstellung, wann immer Sie eine Gameserver-Binärdatei erstellen.

Im Folgenden finden Sie einige gängige Schritte der Bereitstellung von GameLift Amazon-Ressourcen für eine neue Gameserverversion, die Sie mithilfe eines CI/CD-Tool oder -Dienstes automatisieren können:

- Erstellen und testen Sie Ihre Gameserver-Binärdatei.
- Laden Sie die Binärdatei auf Amazon GameLift oder Ihre Hardware hoch.
- Einsatz neuer Flotten im neuen Gebäude.
- Nachdem Sie die neuen Flotten bereitgestellt haben, entfernen Sie die Flotten der vorherigen Version aus Ihrer GameLift Amazon-Warteschlange und ersetzen Sie sie durch die neuen Flotten.
- Nachdem die vorherige Version erfolgreich alle Spielsitzungen beendet hat, werden die AWS CloudFormation Stapel dieser Flotten gelöscht.

Sie können die auch verwendenAWS Cloud Development Kit (AWS CDK), um Ihre GameLift Amazon-Ressourcen zu definieren. Weitere Informationen zum AWS CDK finden Sie im AWS Cloud Development Kit (AWS CDK)Entwicklerhandbuch für.

Gestalte deinen Spiele-Client-Dienst

Wir empfehlen Ihnen, einen Spiele-Client-Dienst zu implementieren, der Ihre Spieler authentifiziert und mit der GameLift Amazon-API kommuniziert. Durch die Implementierung eines benutzerdefinierten Spiele-Client-Dienstes kannst du:

- Passen Sie die Authentifizierung für Ihre Spieler an.
- Kontrollieren Sie, wie Amazon Spielsitzungen GameLift abgleicht und startet.
- Verwenden Sie Ihre Spielerdatenbank für Spielerattribute wie die Fähigkeitsbewertung für die Spielersuche, anstatt dem Client zu vertrauen.

Die Verwendung eines Game-Client-Dienstes reduziert auch die Sicherheitsrisiken, die dadurch entstehen, dass Spieleclients direkt mit Ihrer GameLift Amazon-API interagieren.

Deine Spieler authentifizieren

Du kannst Amazon Cognito und Spielersitzungs-IDs verwenden, um deine Spielclients zu authentifizieren. Verwenden Sie Amazon Cognito-Benutzerpools, um den Lebenszyklus und die Eigenschaften Ihrer Spieleridentitäten zu verwalten.

Wenn Sie es vorziehen, erstellen Sie eine benutzerdefinierte Identitätslösung und hosten Sie sie aufAWS. Sie können Lambda-Autorisierer auch für benutzerdefinierte Autorisierungslogik mit API Gateway verwenden.

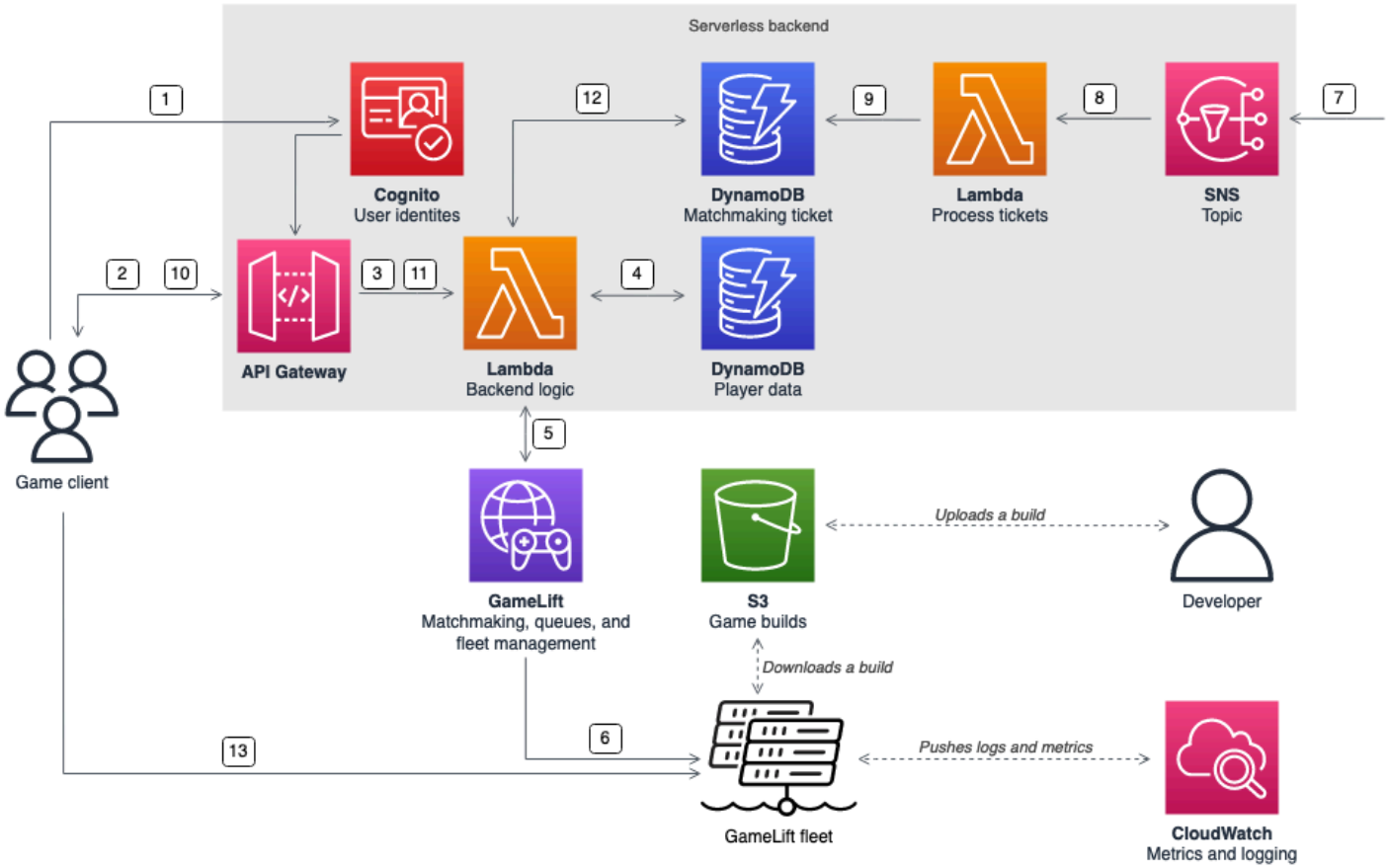
Zusätzliche Ressourcen:

- Verwenden von Identitätspools (föderierte Identitäten) (Amazon Cognito Developer Guide)
- Erste Schritte mit Benutzerpools (Amazon Cognito Developer Guide)
- So richten Sie die Spielerauthentifizierung mit Amazon Cognito ein (AWSfür Spieleblog)

Eigenständige Spielesitzungsserver mit serverlosem Backend

Mithilfe einer serverlosen Client-Service-Architektur kann das Backend den Status von Matchmaking-Tickets aus einer hoch skalierbaren Datenbank einsehen, anstatt direkt auf die Amazon-API zuzugreifen. GameLift

Das folgende Diagramm zeigt ein serverloses Backend, mit dem Spieler Spielen zugeordnet werdenAWS-Services, die auf GameLift Amazon-Flotten laufen. Die folgende Liste enthält eine Beschreibung für jedes nummerierte Callout im Diagramm. Um dieses Beispiel auszuprobieren, klicken Sie auf AWS „Sitzungsbasiertes Multiplayer-Game-Hosting“. GitHub



1. Der Spielclient fordert eine Amazon Cognito-Benutzeridentität aus einem Amazon Cognito-Identitätspool an.
2. Der Spielclient erhält temporäre Zugangsdaten und fordert über eine Amazon API Gateway-API eine Spielsitzung an.
3. API Gateway ruft eine AWS Lambda Funktion auf.
4. Die Lambda-Funktion fordert Spielerdaten aus einer Amazon DynamoDB-NoSQL-Tabelle an. Die Funktion stellt die Amazon Cognito-Identität in den Anforderungskontextdaten bereit.
5. Die Lambda-Funktion fordert über Amazon GameLift FlexMatch Matchmaking ein Match an.
6. FlexMatchordnet eine Gruppe von Spielern mit einer geeigneten Latenz zu und fordert dann über eine GameLift Amazon-Warteschlange eine Platzierung der Spielsitzung an. In der Warteschlange befinden sich Flotten mit einem oder mehreren AWS-Region Standorten.
7. Nachdem Amazon GameLift die Sitzung an einem der Standorte der Flotte veranstaltet hat, GameLift sendet Amazon eine Eventbenachrichtigung an ein Thema des Amazon Simple Notification Service (Amazon SNS).
8. Eine Lambda-Funktion empfängt das Amazon SNS-Ereignis und verarbeitet es.

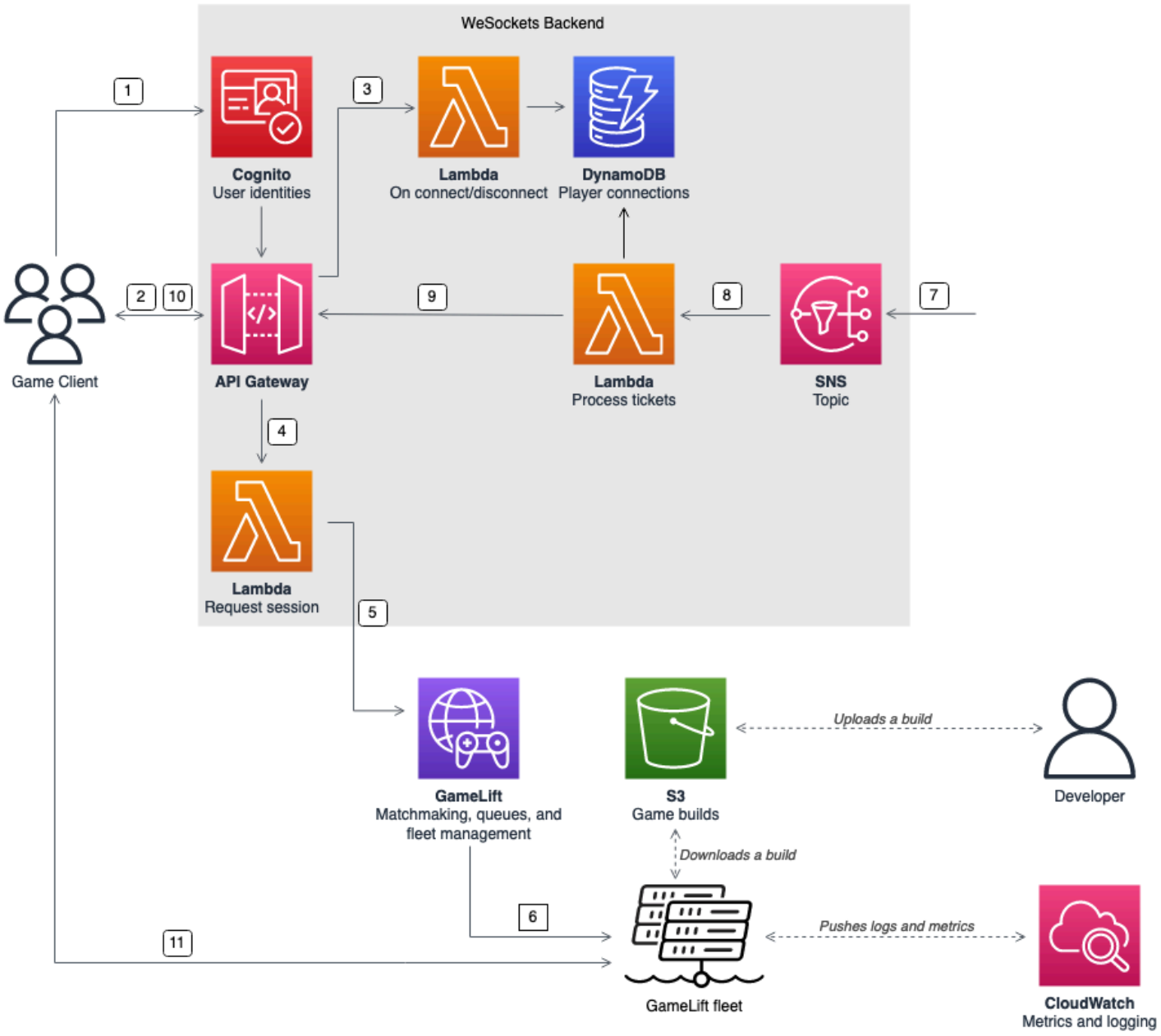
9. Wenn es sich bei dem Matchmaking-Ticket um ein `MatchmakingSucceeded` Ereignis handelt, schreibt die Lambda-Funktion das Ergebnis zusammen mit dem Port und der IP-Adresse des Spielservers in eine DynamoDB-Tabelle.
10. Der Spielclient sendet eine signierte Anfrage an API Gateway, um den Status des Matchmaking-Tickets in einem bestimmten Intervall einzusehen.
11. API Gateway verwendet eine Lambda-Funktion, die den Matchmaking-Ticketstatus überprüft.
12. Die Lambda-Funktion überprüft die DynamoDB-Tabelle, um festzustellen, ob das Ticket erfolgreich war. Wenn dies erfolgreich war, sendet die Funktion den Port und die IP-Adresse des Spielservers zusammen mit der Sitzungs-ID des Spielers an den Client zurück. Wenn das Ticket nicht erfolgreich war, sendet die Funktion eine Antwort, in der bestätigt wird, dass das Spiel noch nicht bereit ist.
13. Der Spielclient stellt über TCP oder UDP eine Verbindung zum Spieleserver her, indem er den Port und die IP-Adresse verwendet, die der Backend-Dienst bereitstellt. Der Spielclient sendet dann die Sitzungs-ID des Spielers an den Spieleserver, der die ID dann mithilfe des Amazon GameLift Server SDK validiert.

Eigenständige Spielesitzungsserver mit einem WebSocket basierten Backend

Mithilfe einer auf Amazon API Gateway WebSocket basierenden Architektur können Sie mithilfe von serverinitiierten Nachrichten Matchmaking-Anfragen stellen WebSockets und Push-Benachrichtigungen senden, wenn das Matchmaking abgeschlossen ist. Diese Architektur verbessert die Leistung, indem sie eine bidirektionale Kommunikation zwischen dem Client und dem Server ermöglicht.

Weitere Informationen zur Verwendung von WebSock API-Gateway-APIs finden Sie unter Arbeiten mit WebSocket APIs.

Das folgende Diagramm zeigt eine WebSocket basierte Backend-Architektur, die API Gateway und andere verwendet, um Spieler den Spielen AWS-Services zuzuordnen, die auf GameLift Amazon-Flotten laufen. Die folgende Liste enthält eine Beschreibung für jedes nummerierte Callout im Diagramm.



1. Der Spielclient fordert eine Amazon Cognito-Benutzeridentität aus einem Amazon Cognito-Identitätspool an.
2. Der Spielclient signiert eine WebSocket Verbindung zu einer API Gateway-API mit den Amazon Cognito-Anmeldeinformationen.
3. API Gateway ruft eine AWS Lambda Funktion für die Verbindung auf. Die Funktion speichert die Verbindungsinformationen in einer Amazon DynamoDB-Tabelle.
4. Der Spielclient sendet über die API Gateway-API über die WebSocket Verbindung eine Nachricht an eine Lambda-Funktion, um eine Sitzung anzufordern.

5. Eine Lambda-Funktion empfängt die Nachricht und fordert dann über Amazon GameLift FlexMatch Matchmaking ein Match an.
6. Nach FlexMatch Spielen FlexMatch fordert eine Gruppe von Spielern über eine GameLift Amazon-Warteschlange eine Platzierung in einer Spielsitzung an.
7. Nachdem Amazon GameLift die Sitzung an einem der Standorte der Flotte veranstaltet hat, GameLift sendet Amazon eine Eventbenachrichtigung an ein Thema des Amazon Simple Notification Service (Amazon SNS).
8. Eine Lambda-Funktion empfängt das Amazon SNS-Ereignis und verarbeitet es.
9. Wenn es sich bei dem Matchmaking-Ticket um ein `MatchmakingSucceeded` Event handelt, fordert die Lambda-Funktion die richtige Spielerverbindung von DynamoDB an. Die Funktion sendet dann über die API Gateway-API über die WebSocket Verbindung eine Nachricht an den Spielclient. In dieser Architektur fragt der Spielclient den Status der Spielersuche nicht aktiv ab.
10. Der Spielclient erhält über die WebSocket Verbindung den Port und die IP-Adresse des Spielservers zusammen mit der Spielersitzungs-ID.
11. Der Spielclient stellt über TCP oder UDP über den Port und die IP-Adresse, die der Backend-Dienst bereitstellt, eine Verbindung zum Spieleserver her. Der Spielclient sendet auch die Sitzungs-ID des Spielers an den Spieleserver, der die ID dann mithilfe des Amazon GameLift Server SDK validiert.

Metriken und Protokollierung für Amazon einrichten GameLift

Sie können die von Ihren GameLift Amazon-Spieleservern und Ressourcen gesammelten Daten verwenden, um Anomalien zu identifizieren. Sie können auch Kennzahlen verwenden, um die Leistung zu verbessern.

Zu den wichtigsten Bereichen, die Amazon beachten sollte, GameLift gehören:

- GameLiftAmazon-Servicekennzahlen — Amazon GameLift stellt CloudWatch Amazon-Metriken zu Ihren Ressourcen bereit, einschließlich Spieleservern, Flotten, Warteschlangen und. FlexMatch Sie finden diese Metriken in der GameLift Amazon-Konsole und der CloudWatch Konsole. Weitere Informationen zu GameLift Amazon-Metriken CloudWatch finden Sie unterÜberwachen Sie Amazon GameLift mit Amazon CloudWatch.
- Spieleserver-Metriken — Amazon GameLift kann nicht auf Ihre Gameserver-Metriken zugreifen. Du kannst jedoch mithilfe des CloudWatch Agenten benutzerdefinierte Metriken CloudWatch direkt von deinem Spielserver an senden. Sie können auch die Fleet-Rolle AWS Identity and Access

Management (IAM) und das AWS SDK verwenden, um Metriken direkt an diese zu CloudWatch senden. Ein Beispiel für die Konfiguration von Metriken findest du unter Sitzungsbasiertes Multiplayer-Game-Hosting, aktiviert AWS. GitHub Dieses Repository enthält ein Beispiel für eine CloudWatch Agentenkonfiguration und Code für einen C#-StatsD-Client.

- Spieleserverprotokolle — Verwenden Sie die Amazon Server SDK-Konfiguration, um Ihre Gameserver-Logdateien auf dem GameLift Spieleserver zu konfigurieren. Sie können Amazon CloudWatch Logs auch als Protokollverwaltungslösung in Echtzeit verwenden und Protokolle mit dem CloudWatch Agenten konfigurieren. Weitere Informationen finden Sie unter Protokollieren von Servernachrichten in Amazon GameLift.

Checklisten zum Start von Spielen

Sie können diese Checklisten verwenden, um die Phasen der Bereitstellung Ihres Spiels zu überprüfen. In den Checklisten sind die mit [Kritisch] markierten Punkte für Ihren Produktionsstart von entscheidender Bedeutung.

Themen

Onboarding

Nutze die folgende Checkliste, um die Elemente zu verfolgen, die für das Onboarding deines Spiels für GameLift Amazon-Hosting erforderlich sind. Artikel, die mit [Kritisch] gekennzeichnet sind, sind für Ihren Produktionsstart von entscheidender Bedeutung.

- [Kritisch] Füllen Sie den GameLift Amazon-Onboarding-Fragebogen in der GameLiftAmazon-Konsole aus.
- [Kritisch] Entwerfen und implementieren Sie einen Backend-Service für Spielclients, um mit Ihren Spielservern zu interagieren.
- [Kritisch] Erstellen Sie AWS Identity and Access Management (IAM-) Rollen, die Sie Amazon GameLift Server-Instances für den Zugriff auf andere AWS Ressourcen bereitstellen.

- [Kritisch] Entwurf und Implementierung eines Failovers AWS-Regionen für andere Formulare FlexMatch und Warteschlangen.
- Plane den Einsatz von Flotten an deine Zielorte und berücksichtige dabei die Warteschlange und die Flottenstruktur deines Spiels.
- Automatisieren Sie Ihre Bereitstellung mithilfe von Infrastructure as Code (IaC) mit AWS CloudFormation und demAWS Cloud Development Kit (AWS CDK).
- Erfassen Sie Protokolle und Analysen mit Amazon CloudWatch und Amazon Simple Storage Service (Amazon S3).

Testen

Verwenden Sie die folgende Checkliste, um die Testergebnisse zu verfolgen, während Sie Ihr Spiel mit GameLift Amazon-Hosting entwickeln. Artikel, die mit [Kritisch] gekennzeichnet sind, sind für Ihren Produktionsstart von entscheidender Bedeutung.

- [Kritisch] Füllen Sie den Fragebogen zur Markteinführung aus und senden Sie den ausgefüllten Fragebogen an das GameLift Amazon-Launch-Team. Den Fragebogen zur Markteinführung finden Sie in der GameLiftAmazon-Konsole.
- [Kritisch] Beantragen Sie Erhöhungen der GameLift Amazon-Servicekontingente und anderer AWS-Service Kontingente, damit Ihre Live-Umgebung an die Produktionsanforderungen angepasst werden kann.
- [Kritisch] Stellen Sie sicher, dass die offenen Ports der Live-Flotten dem Bereich der Ports entsprechen, die Ihre Server verwenden könnten.
- [Kritisch] Schließen Sie den RDP-Port 3389 und den SSH-Port 22.
- Entwickeln Sie einen Plan für das DevOps Management Ihres Spiels. Wenn Sie Amazon CloudWatch Logs oder CloudWatch benutzerdefinierte Amazon-Metriken verwenden, definieren Sie Alarme für schwerwiegende oder kritische Probleme in der Serverflotte. Simulieren Sie Fehler und testen Sie die Runbooks.
- Stellen Sie sicher, dass die Anzahl der Server, die auf einer Instance bei voller Auslastung laufen, innerhalb der Funktionen des Serverinstanztyps liegt.
- Passen Sie Ihre Skalierungspolitik so an, dass Sie zunächst konservativer vorgehen und mehr ungenutzte Kapazität bereitstellen, als Sie für erforderlich halten. Sie können später die Kosten optimieren. Erwägen Sie den Einsatz einer zielorientierten Skalierungsrichtlinie mit 20 Prozent ungenutzter Kapazität.

- Verwenden Sie FlexMatch Latenzregeln, um Spieler zusammenzubringen, die sich geografisch in der Nähe derselben befinden. AWS-Region Testen Sie mit synthetischen Latenzdaten von Ihrem Lasttest-Client, wie sich dies unter Last verhält.
- Testen Sie Ihre Infrastruktur für Spielerauthentifizierung und Spielsitzungen, um zu sehen, ob sie effektiv skaliert werden kann, um der Nachfrage gerecht zu werden.
- Stellen Sie sicher, dass ein Server, der mehrere Tage lang in Betrieb war, immer noch Verbindungen annehmen kann.
- Erhöhen Sie Ihren AWS Support Tarif auf Business oder Enterprise, damit er Ihnen bei Problemen oder Ausfällen weiterhelfen AWS kann.

Starten

Verwenden Sie die folgende Checkliste, um die Veröffentlichungen Ihres bei Amazon GameLift gehosteten Spiels zu verfolgen. Artikel, die mit [Kritisch] gekennzeichnet sind, sind für Ihren Produktionsstart von entscheidender Bedeutung.

- [Kritisch] Setze die Flottenschutzrichtlinie auf vollen Schutz für alle aktiven Flotten, damit aktive Spielsitzungen nicht durch Herunterskalieren unterbrochen werden.
- [Kritisch] Legen Sie die maximalen Flottengrößen fest, die hoch genug sind, um der erwarteten Spitzennachfrage zumindest gerecht zu werden. Wir empfehlen Ihnen, Ihre maximale Größe für unvorhergesehene Nachfrage zu verdoppeln.
- Ermutigen Sie Ihr gesamtes Entwicklerteam, an der Launch-Veranstaltung teilzunehmen, und überwachen Sie Ihren Spielstart in einem Launch Room.
- Überwachen Sie die Latenz und das Spielererlebnis der Spieler.

Nach dem Start

Verwenden Sie die folgende Checkliste, um die Artikel für Ihr Spiel zu verfolgen, die nach der Veröffentlichung auf Amazon gehostet wurden. GameLift

- Passen Sie die Skalierungsregeln an, um ungenutzte Kapazität zu minimieren.
- Ändern Sie FlexMatch die Regeln oder fügen Sie zusätzliche Standorte hinzu, je nach Ihren Latenzanforderungen.

- Optimieren Sie die ausführbare Serverdatei, da sich ihre Leistungseffizienz direkt auf die Flottenkosten auswirkt. Um mehr Spielesitzungen mit derselben Infrastruktur auszuführen, erhöhen Sie die Anzahl der Serverprozesse pro Instanz.
- Nutze deine Analysedaten, um die Weiterentwicklung voranzutreiben, das Spielererlebnis und die Lebensdauer von Spielen zu verbessern und die Monetarisierung zu optimieren.

Vorbereiten von Spielen für Amazon GameLift

Richten Sie die Kommunikation zwischen Ihrem Spiel und Amazon ein GameLift, um Ihr Multiplayer-Spiel auf das Hosting auf Amazon vorzubereiten GameLift. Die Themen in diesem Abschnitt bieten detaillierte Hilfe bei der Integration Ihres Spiels mit Amazon GameLift, benutzerdefinierten Spieleservern und Echtzeitservern sowie für das Hinzufügen von Matchmaking mit FlexMatch.

Themen

Integrieren Sie Spiele mit benutzerdefinierten Spieleservern

Amazon GameLift bietet ein vollständiges Toolset zur Vorbereitung Ihrer Multiplayer-Spiele und benutzerdefinierte Spieleserver für die Ausführung auf AmazonGameLift. Die Amazon GameLift SDKs enthalten Bibliotheken, die Spieleclients und Server benötigen, um mit Amazon GameLift zu kommunizieren. Weitere Informationen zu den SDKs und wo sie erhältlich sind, finden Sie unterEntwicklungsunterstützung mit Amazon GameLift.

Die Themen in diesem Abschnitt enthalten detaillierte Anweisungen zum Hinzufügen von GameLift Amazon-Funktionen zu Ihrem Spieleclient und Gameserver, bevor Sie sie bei Amazon GameLift bereitstellen. Eine vollständige Roadmap, wie Sie Ihr Spiel bei Amazon zum Laufen bringen könnenGameLift, finden Sie unterRoadmap für GameLift verwaltetes Hosting von Amazon.

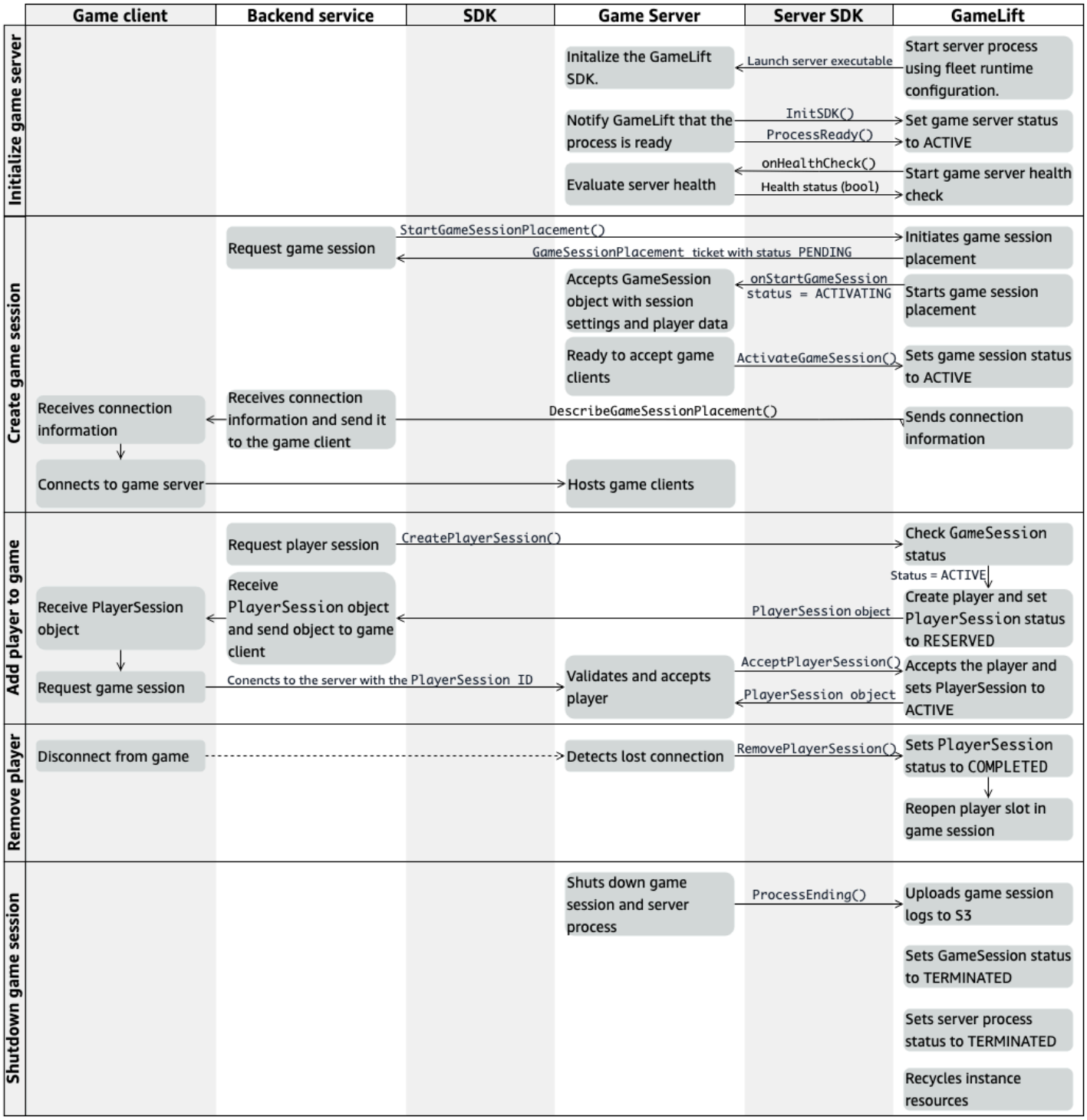
Themen

- Testen Sie Ihre Integration mit Amazon GameLift Local

Interaktionen zwischen Amazon GameLift und Game-Client und Server

In diesem Thema werden die Interaktionen zwischen dem Spielclient, einem Backend-Service, einem Spieleserver und Amazon GameLift beschrieben.

Das folgende Diagramm veranschaulicht die Interaktionen zwischen dem Spielclient, dem Backend-Service, dem Amazon GameLift SDK, dem verwalteten EC2-Spieleserver, dem Amazon GameLift Server-SDK und Amazon. GameLift Eine detaillierte Beschreibung der gezeigten Interaktionen finden Sie in den folgenden Abschnitten auf dieser Seite.



Initialisiere einen Gameserver

In den folgenden Schritten werden die Interaktionen beschrieben, die auftreten, wenn Sie Ihren Spielserver für die Ausrichtung von Spielesitzungen vorbereiten.

1. Amazon GameLift startet die ausführbare Serverdatei auf einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance.
2. Der Gameserver ruft:

   a. `InitSDK()`, um das Server-SDK zu initialisieren.

   b. `ProcessReady()`um die Bereitschaft zur Spielsitzung, Verbindungsinformationen und den Speicherort der Spielsitzungsprotokolldateien mitzuteilen.

   Der Serverprozess wartet dann auf einen Rückruf von Amazon. GameLift
3. Amazon GameLift aktualisiert den Status des Serverprozesses, `ACTIVE` um die Platzierung von Spielsitzungen zu ermöglichen.
4. Amazon GameLift beginnt mit dem Aufrufen des `onHealthCheck` Callbacks und ruft ihn weiterhin regelmäßig auf, solange der Serverprozess aktiv ist. Der Serverprozess kann innerhalb einer Minute einen fehlerfreien oder fehlerfreien Zustand melden.

Erstellen einer Spielsitzung

Nachdem du deinen Gameserver initialisiert hast, finden die folgenden Interaktionen statt, wenn du Spielsitzungen für deine Spieler erstellst.

1. Der Backend-Dienst ruft den SDK-Vorgang `StartGameSessionPlacement()` auf.
2. Amazon GameLift erstellt ein neues `GameSessionPlacement` Ticket mit Status `PENDING` und sendet es an den Backend-Service zurück.
3. Der Backend-Service ruft den Status eines Platzierungstickets aus einer Warteschlange ab. Weitere Informationen finden Sie unter Richten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein.
4. Amazon GameLift beginnt mit der Platzierung von Spielsitzungen, indem es eine geeignete Flotte auswählt und in einer Flotte mit `0` Spielsitzungen nach einem aktiven Serverprozess sucht. Wenn Amazon einen Serverprozess GameLift lokalisiert, GameLift geht Amazon wie folgt vor:

   a. Erstellt ein `GameSession` Objekt mit den Einstellungen der Spielsitzung und den Spielerdaten aus der Platzierungsanfrage mit einem `ACTIVATING` Status.

   b. Ruft den on`StartGameSession` Callback auf dem Serverprozess auf. Amazon GameLift leitet Informationen an das `GameSession` Objekt weiter, die darauf hinweisen, dass der Serverprozess die Spielsitzung einrichten kann.

   c. Ändert die Anzahl der Spielsitzungen des Serverprozesses auf`1`.

5. Der Serverprozess führt die `onStartGameSession` Callback-Funktion aus. Wenn der Serverprozess bereit ist, Spielerverbindungen anzunehmen, ruft er an `ActivateGameSession()` und wartet auf Spielerverbindungen.
6. Amazon GameLift aktualisiert das `GameSession` Objekt mit Verbindungsinformationen für den Serverprozess. (Diese Information beinhaltet die Porteinstellung, mit der gemeldet wurde`ProcessReady()`.) Amazon ändert GameLift auch den Status in`ACTIVE`.
7. Der Backend-Service ruft `DescribeGameSessionPlacement()` an, um den aktualisierten Ticketstatus zu ermitteln. Der Backend-Dienst verwendet dann die Verbindungsinformationen, um den Spielclient mit dem Serverprozess zu verbinden und der Spielsitzung beizutreten.

Füge einen Spieler zu einem Spiel hinzu

Diese Sequenz beschreibt den Vorgang des Hinzufügens eines Spielers zu einer bestehenden Spielsitzung. Spielersitzungen können auch als Teil einer Platzierungsanfrage für eine Spielsitzung angefordert werden.

1. Der Backend-Dienst ruft den Client-API-Vorgang `CreatePlayerSession()` mit einer Spielesitzungs-ID auf.
2. Amazon GameLift überprüft den Status der Spielsitzung (muss sein`ACTIVE`) und sucht in der Spielsitzung nach einem freien Spielerplatz. Wenn ein Slot verfügbar ist, GameLift geht Amazon wie folgt vor:

   a. Erstellt ein neues `PlayerSession` Objekt und setzt den Status auf`RESERVED`.

   b. Reagiert auf die Backend-Serviceanfrage mit dem `PlayerSession` Objekt.

3. Der Backend-Dienst verbindet den Spielclient mit der Spielersitzungs-ID direkt mit dem Serverprozess.
4. Der Server ruft den Server-API-Vorgang `AcceptPlayerSession()` auf, um die Player-Sitzungs-ID zu überprüfen. Wenn es validiert wird, GameLift übergibt Amazon das `PlayerSession` Objekt an den Serverprozess. Der Serverprozess akzeptiert die Verbindung oder weist sie zurück.
5. Amazon GameLift macht einen der folgenden Schritte:

   a. Wenn die Verbindung akzeptiert wird, GameLift setzt Amazon den `PlayerSession` Status auf`ACTIVE`.

b. Wenn innerhalb von 60 Sekunden nach dem ursprünglichen `CreatePlayerSession()` Anruf des Backend-Servers keine Antwort eingeht, GameLift ändert Amazon den `PlayerSession` Status auf `TIMEDOUT` und öffnet den Spielerplatz in der Spielsitzung erneut.

Einen Spieler entfernen

Wenn Spieler aus einer Spielsitzung entfernt werden, um Platz für neue Spieler zu schaffen, finden die folgenden Interaktionen statt.

1. Ein Spieler trennt die Verbindung zum Spiel.
2. Der Server erkennt den Verbindungsverlust und ruft den Server-API-Vorgang auf`RemovePlayerSession()`.
3. Amazon GameLift ändert den `PlayerSession` Status auf `COMPLETED` und öffnet den Spielerplatz in der Spielsitzung erneut.

Schließe die Spielsitzung

Diese Abfolge von Interaktionen tritt auf, wenn ein Serverprozess die aktuelle Spielsitzung beendet.

1. Der Server fährt die Spielsitzung und den Server herunter.
2. Der Server ruft Amazon `ProcessEnding()` anGameLift.
3. Amazon GameLift geht wie folgt vor:

   a. Lädt Spielsitzungsprotokolle auf Amazon Simple Storage Service (Amazon S3) hoch.

   b. Ändert den `GameSession` Status in`TERMINATED`.

   c. Ändert den Serverprozessstatus in`TERMINATED`.

   d. Recycelt Instanzressourcen.

Integriere deinen Gameserver mit Amazon GameLift

Nachdem Ihr benutzerdefinierter Gameserver bereitgestellt wurde und auf GameLift Amazon-Instances ausgeführt wird, muss er in der Lage sein, mit Amazon GameLift (und möglicherweise anderen Ressourcen) zu interagieren. In diesem Abschnitt wird beschrieben, wie Sie Ihre Gameserver-Software in Amazon integrierenGameLift.

> **Note**
>
> In diesen Anweisungen wird davon ausgegangen, dass Sie ein Gameserverprojekt erstellt haben AWS-Konto und dass Sie über ein bestehendes Gameserver-Projekt verfügen.

Die Themen in diesem Abschnitt beschreiben, wie Sie mit den folgenden Integrationsaufgaben umgehen:

- Stellen Sie die Kommunikation zwischen Amazon GameLift und Ihren Spielservern her.
- Generieren und verwenden Sie ein TLS-Zertifikat, um eine sichere Verbindung zwischen dem Spielclient und dem Spieleserver herzustellen.
- Erteile deiner Gameserver-Software die Erlaubnis, mit anderen AWS Ressourcen zu interagieren.
- Erlaube den Gameserverprozessen, Informationen über die Flotte abzurufen, auf der sie laufen.

Themen

Füge Amazon GameLift zu deinem Gameserver hinzu

Ihr benutzerdefinierter Gameserver muss mit Amazon kommunizierenGameLift, da jeder Gameserverprozess in der Lage sein muss, auf Ereignisse zu reagieren, die Amazon GameLift startet. Ihr Gameserver muss Amazon auch über den Serverprozessstatus und die Spielerverbindungen auf dem GameLift Laufenden halten. Weitere Informationen darüber, wie Ihr Gameserver, Ihr Backend-Service, Ihr Spieleclient und Amazon GameLift zusammenarbeiten, um das Game-Hosting zu verwalten, finden Sie unterInteraktionen zwischen Amazon GameLift und Game-Client und Server.

Um Ihren Gameserver auf die Interaktion mit Amazon vorzubereitenGameLift, fügen Sie das Amazon GameLift Server SDK zu Ihrem Gameserver-Projekt hinzu und integrieren Sie die in diesem Thema beschriebenen Funktionen. Das Server-SDK ist in mehreren Sprachen verfügbar. Weitere Informationen zum Amazon GameLift Server SDK finden Sie unterEntwicklungsunterstützung mit Amazon GameLift.

Server-SDK-API-Referenzen:

- Amazon GameLift Server SDK 5.x-Referenz für C++
- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity
- Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x

Initialisieren Sie den Serverprozess

Fügen Sie Code hinzu, um die Kommunikation mit Amazon herzustellen GameLift und um zu melden, dass der Serverprozess bereit ist, eine Spielsitzung zu veranstalten. Dieser Code muss vor jedem GameLift Amazon-Code ausgeführt werden.

1. Initialisieren Sie den Amazon GameLift API-Client durch einen Anruf`InitSdk()`. Um einen Serverprozess auf einer GameLift Anywhere Amazon-Rechenressource zu initialisieren, rufen Sie `InitSdk()` mit dem folgenden Befehl auf: `ServerParameters`

   - Die URL des Websockets, der für die Verbindung zu deinem Gameserver verwendet wird.
   - Die ID des Prozesses, der zum Hosten Ihres Spieleservers verwendet wurde.
   - Die ID des Rechners, der die Prozesse Ihres Gameservers hostet.
   - Die ID der GameLift Flotte, die Ihre GameLift Anywhere Amazon-Rechenleistung enthält.
   - Das durch den GameLift Amazon-Vorgang generierte AutorisierungstokenGetComputeAuthToken.

   > **Note**
   >
   > Um einen Spieleserver auf einer von Amazon GameLift verwalteten Amazon EC2-Instance zu initialisieren, ServerParameters verwenden Sie den Standardkonstruktor `InitSDK()` (C++) (C#) (C#) (Unreal) Amazon GameLift richtet die Rechenumgebung ein und stellt automatisch GameLift für Sie eine Verbindung zu Amazon her.

2. Informieren Sie AmazonGameLift, dass ein Serverprozess bereit ist, eine Spielesitzung zu hosten. Rufen Sie `ProcessReady()` (C++) (C#) (Unreal) (C++) mit den folgenden Informationen auf. (Beachten Sie, dass Sie pro Serverprozess `ProcessReady()` nur einmal aufrufen sollten).

   - Die Portnummer, die der Serverprozess verwendet. Der Backend-Dienst stellt den Spielclients die Portnummer und eine IP-Adresse zur Verfügung, um sich mit dem Serverprozess zu verbinden und an einer Spielsitzung teilzunehmen.

- Der Speicherort von Dateien, wie z. B. Spielsitzungsprotokollen, die Amazon speichern GameLift soll. Der Serverprozess generiert diese Dateien während einer Spielsitzung. Sie werden vorübergehend auf der Instanz gespeichert, auf der der Serverprozess ausgeführt wird, und gehen verloren, wenn die Instanz heruntergefahren wird. Alle von Ihnen aufgelisteten Dateien werden auf Amazon hochgeladenGameLift. Sie können über die GameLiftAmazon-Konsole oder durch Aufrufen der GameLift Amazon-API-Operation GetGameSessionLogUrl () auf diese Dateien zugreifen.
- Die Namen der Callback-Funktionen, die Amazon für Ihren Serverprozess aufrufen GameLift kann. Ihr Gameserver muss diese Funktionen implementieren. Weitere Informationen finden Sie unter (C++) (C#) (Unreal) (C++)
  - (Optional) `onHealthCheck` — Amazon GameLift ruft diese Funktion regelmäßig auf, um einen Statusbericht vom Server anzufordern.
  - `onStartGameSession`— Amazon GameLift ruft diese Funktion als Antwort auf die Client-Anfrage auf CreateGameSession().
  - `onProcessTerminate`— Amazon GameLift zwingt den Serverprozess zum Stoppen und sorgt dafür, dass er ordnungsgemäß heruntergefahren wird.
  - (Optional) `onUpdateGameSession` — Amazon GameLift übermittelt ein aktualisiertes Spielsitzungsobjekt an den Spieleserver oder stellt eine Statusaktualisierung bei einer Anfrage zur Verfüllung von Matches bereit. Die FlexMatchBackfill-Funktion erfordert diesen Callback.

Du kannst auch einen Spielserver einrichten, um sicher auf AWS Ressourcen zuzugreifen, die dir gehören oder die du kontrollierst. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

(Optional) Serverprozessstatus melden

Füge Code zu deinem Gameserver hinzu, um die Callback-Funktion `onHealthCheck()` zu implementieren. Amazon GameLift ruft diese Callback-Methode regelmäßig auf, um Gesundheitskennzahlen zu sammeln. Gehen Sie wie folgt vor, um diese Callback-Funktion zu implementieren:

- Beurteilen Sie den Integritätsstatus des Serverprozesses. Sie können beispielsweise den Serverprozess als fehlerhaft melden, wenn externe Abhängigkeiten fehlgeschlagen sind.

- Schließen Sie die Zustandsprüfung ab, und antworten Sie auf den Callback innerhalb von 60 Sekunden. Wenn Amazon GameLift innerhalb dieser Zeit keine Antwort erhält, wird der Serverprozess automatisch als fehlerhaft eingestuft.
- Gibt einen booleschen Wert zurück: true für gesund, false für ungesund.

Wenn Sie keinen Healthcheck-Callback implementieren, GameLift betrachtet Amazon den Serverprozess als fehlerfrei, es sei denn, der Server reagiert nicht.

Amazon GameLift verwendet die Integrität von Serverprozessen, um fehlerhafte Prozesse zu beenden und Ressourcen freizugeben. Wenn ein Serverprozess weiterhin als fehlerhaft gemeldet wird oder bei drei aufeinanderfolgenden Zustandsprüfungen nicht reagiert, GameLift kann Amazon den Prozess beenden und einen neuen starten. Amazon GameLift erfasst Kennzahlen zum Zustand der Serverprozesse einer Flotte.

(Optional) Holen Sie sich ein TLS-Zertifikat

Wenn der Serverprozess auf einer Flotte läuft, für die die TLS-Zertifikatsgenerierung aktiviert ist, kannst du das TLS-Zertifikat abrufen, um eine sichere Verbindung mit einem Spielclient herzustellen und die Client-Server-Kommunikation zu verschlüsseln. Eine Kopie des Zertifikats wird auf der Instance gespeichert. Rufen Sie (C++) (C#) `GetComputeCertificate( )`(Unreal) (C++) (C++) (C#) (Unreal).

Starte eine Spielsitzung

Fügen Sie Code zur Implementierung der Callback-Funktion `onStartGameSession` hinzu. Amazon GameLift ruft diesen Callback auf, um eine Spielsitzung auf dem Server zu starten.

Die `onStartGameSession` Funktion verwendet ein GameSessionObjekt als Eingabeparameter. Dieses Objekt enthält wichtige Informationen zur Spielsitzung, z. B. die maximale Spieleranzahl. Es kann auch Spieldaten und Spielerdaten enthalten. Die Funktionsimplementierung sollte die folgenden Aufgaben erfüllen:

- Initiieren Sie Aktionen, um auf der Grundlage der `GameSession` Eigenschaften eine neue Spielsitzung zu erstellen. Der Spielserver muss mindestens die Spielesitzungs-ID zuordnen, auf die die Spielclients verweisen, wenn sie sich mit dem Serverprozess verbinden.
- Verarbeiten Sie Spieldaten und Spielerdaten nach Bedarf. Diese Daten befinden sich im `GameSession` Objekt.
- Informieren Sie AmazonGameLift, wenn eine neue Spielsitzung für die Aufnahme von Spielern bereit ist. Rufen Sie die Server-API-Operation `ActivateGameSession( )`(C++) (C#) (Unreal)

(C++) auf. Als Reaktion auf einen erfolgreichen Anruf GameLift ändert Amazon den Status der Spielsitzung auf`ACTIVE`.

(Optional) Bestätige einen neuen Spieler

Wenn du den Status von Spielersitzungen verfolgst, füge Code hinzu, um einen neuen Spieler zu validieren, wenn er sich mit einem Spielserver verbindet. Amazon GameLift verfolgt die aktuellen Spieler und die verfügbaren Spielautomaten für Spielsitzungen.

Zur Bestätigung muss ein Spielclient, der Zugriff auf die Spielsitzung anfordert, eine Spielersitzungs-ID angeben. Amazon generiert diese ID GameLift automatisch, wenn ein Spieler mit StartGameSessionPlacement() oder StartMatchmaking() darum bittet, einem Spiel beizutreten. Die Spielersitzung reserviert dann einen freien Platz in einer Spielsitzung.

Wenn der Gameserverprozess eine Verbindungsanfrage des Spielclients empfängt, ruft er `AcceptPlayerSession()` (C++) (C#) (Unreal) (C++) mit der Spielersitzungs-ID auf. Daraufhin GameLift überprüft Amazon, ob die Sitzungs-ID des Spielers einem offenen Slot entspricht, der in der Spielsitzung reserviert wurde. Nachdem Amazon die Player-Sitzungs-ID GameLift überprüft hat, akzeptiert der Serverprozess die Verbindung. Der Spieler kann dann an der Spielsitzung teilnehmen. Wenn Amazon die Player-Sitzungs-ID GameLift nicht überprüft, verweigert der Serverprozess die Verbindung.

(Optional) Das Ende einer Spielersitzung melden

Wenn du den Status von Spielersitzungen verfolgst, füge einen Code hinzu, um Amazon zu benachrichtigen, GameLift wenn ein Spieler die Spielsitzung verlässt. Dieser Code sollte ausgeführt werden, wenn der Serverprozess eine aufgegebene Verbindung erkennt. Amazon GameLift verwendet diese Benachrichtigung, um die aktuellen Spieler und die verfügbaren Slots in der Spielsitzung zu verfolgen.

Um unterbrochene Verbindungen zu verarbeiten, fügen Sie in Ihrem Code einen Aufruf zur Server-API-Operation `RemovePlayerSession()` (C++) (C#) (Unreal) (C++) mit der entsprechenden Player-Sitzungs-ID hinzu.

Beenden Sie eine Spielsitzung

Fügen Sie der Sequenz zum Herunterfahren des Serverprozesses Code hinzu, um Amazon zu benachrichtigenGameLift, wenn eine Spielsitzung endet. Um Hosting-Ressourcen zu recyceln und zu aktualisieren, GameLift fährt Amazon Serverprozesse herunter, nachdem die Spielsitzung abgeschlossen ist.

Rufen Sie zu Beginn des Codes zum Herunterfahren des Serverprozesses die Server-API-Operation `ProcessEnding()`(C++) (C#) (Unreal) (C++) auf. Dieser Aufruf informiert Amazon darüberGameLift, dass der Serverprozess heruntergefahren wird. Amazon GameLift ändert den Status der Spielsitzung und den Serverprozessstatus auf`TERMINATED`. Nach dem Anruf `ProcessEnding()` kann der Prozess gefahrlos beendet werden.

Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses

Fügen Sie Code hinzu, um den Serverprozess als Reaktion auf eine Benachrichtigung von Amazon herunterzufahrenGameLift. Amazon GameLift sendet diese Benachrichtigung, wenn der Serverprozess durchweg als fehlerhaft gemeldet wird oder wenn die Instance, in der der Serverprozess ausgeführt wird, beendet wird. Amazon GameLift kann eine Instance im Rahmen eines Kapazitätsabbaus oder als Reaktion auf eine Spot-Instance-Unterbrechung stoppen.

Um eine Benachrichtigung über das Herunterfahren zu verarbeiten, nimm die folgenden Änderungen an deinem Gameservercode vor:

- Implementieren Sie die Callback-Funktion on`ProcessTerminate()`. Diese Funktion sollte den Code aufrufen, der den Serverprozess herunterfährt. Wenn Amazon diesen Vorgang GameLift aufruft, werden Spot-Instance-Unterbrechungen mit einer zweiminütigen Vorwarnung angezeigt. Dieser Hinweis gibt dem Serverprozess Zeit, die Verbindung der Spieler ordnungsgemäß zu trennen, die Spielstatusdaten beizubehalten und andere Bereinigungsaufgaben durchzuführen.
- Rufen Sie die Server-API-Operation `GetTerminationTime()` (C++) (C#) (Unreal) (C ++) über den Shutdown-Code Ihres Spielservers auf. Wenn Amazon einen Aufruf zum Stoppen des Serverprozesses gesendet GameLift hat, wird die geschätzte Abbruchzeit `GetTerminationTime()` zurückgegeben.
- Rufen Sie zu Beginn des Codes zum Herunterfahren Ihres Spielservers die Server-API-Operation `ProcessEnding()`(C++) (C#) (Unreal) (C++) auf. Dieser Aufruf informiert Amazon darüberGameLift, dass der Serverprozess heruntergefahren wird, und Amazon ändert GameLift dann den Serverprozessstatus in. `TERMINATED` Nach dem Anruf `ProcessEnding()` kann der Prozess gefahrlos beendet werden.

Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten

Wenn Sie einen Spieleserver-Build für den Einsatz auf GameLift Amazon-Flotten erstellen, möchten Sie vielleicht, dass die Anwendungen in Ihrem Game-Build direkt und sicher mit anderen AWS Ressourcen kommunizieren, die Sie besitzen. Da Amazon Ihre Spiele-Hosting-Flotten GameLift

verwaltet, müssen Sie Amazon GameLift eingeschränkten Zugriff auf diese Ressourcen und Dienste gewähren.

Einige Beispielszenarien umfassen:

- Verwenden Sie einen CloudWatch Amazon-Agenten, um Metriken, Protokolle und Traces von verwalteten EC2-Flotten und Flotten zu sammeln Anywhere
- Senden Sie Instance-Protokolldaten an Amazon Logs. CloudWatch
- Besorgen Sie sich Spieldateien, die in einem Amazon Simple Storage Service (Amazon S3) - Bucket gespeichert sind.
- Lesen und Schreiben von Spieldaten (wie Spielmodi oder Inventar), die in einer Amazon DynamoDB DynamoDB-Datenbank oder einem anderen Datenspeicherdienst gespeichert sind.
- Senden Sie mithilfe von Amazon Simple Queue Service (Amazon SQS) Signale direkt an eine Instance.
- Greifen Sie auf benutzerdefinierte Ressourcen zu, die auf Amazon Elastic Compute Cloud (Amazon EC2) bereitgestellt und ausgeführt werden.

Amazon GameLift unterstützt die folgenden Methoden zur Einrichtung des Zugriffs:

- Greifen Sie mit einer IAM-Rolle auf AWS Ressourcen zu
- Greifen Sie mit VPC-Peering auf AWS Ressourcen zu

Greifen Sie mit einer IAM-Rolle auf AWS Ressourcen zu

Verwenden Sie eine IAM-Rolle, um anzugeben, wer auf Ihre Ressourcen zugreifen kann, und um Beschränkungen für diesen Zugriff festzulegen. Vertrauenswürdige Parteien können eine Rolle „übernehmen“ und temporäre Sicherheitsanmeldedaten erhalten, die sie zur Interaktion mit den Ressourcen berechtigen. Wenn die Parteien API-Anfragen im Zusammenhang mit der Ressource stellen, müssen sie die Anmeldeinformationen angeben.

Gehen Sie wie folgt vor, um den Zugriff einzurichten, der durch eine IAM-Rolle gesteuert wird:

1. Erstellen Sie die IAM-Rolle
2. Ändern Sie Anwendungen, um Anmeldeinformationen abzurufen
3. Ordnen Sie der IAM-Rolle eine Flotte zu

Erstellen Sie die IAM-Rolle

In diesem Schritt erstellen Sie eine IAM-Rolle mit einer Reihe von Berechtigungen zur Steuerung des Zugriffs auf Ihre AWS Ressourcen und einer Vertrauensrichtlinie, die Amazon die GameLift Rechte zur Nutzung der Rollenberechtigungen einräumt.

Anweisungen zur Einrichtung der IAM-Rolle finden Sie unter. Einrichten einer IAM-Servicerolle für Amazon GameLift Wählen Sie bei der Erstellung der Berechtigungsrichtlinie bestimmte Dienste, Ressourcen und Aktionen aus, mit denen Ihre Anwendungen funktionieren müssen. Es hat sich bewährt, den Umfang der Berechtigungen so weit wie möglich einzuschränken.

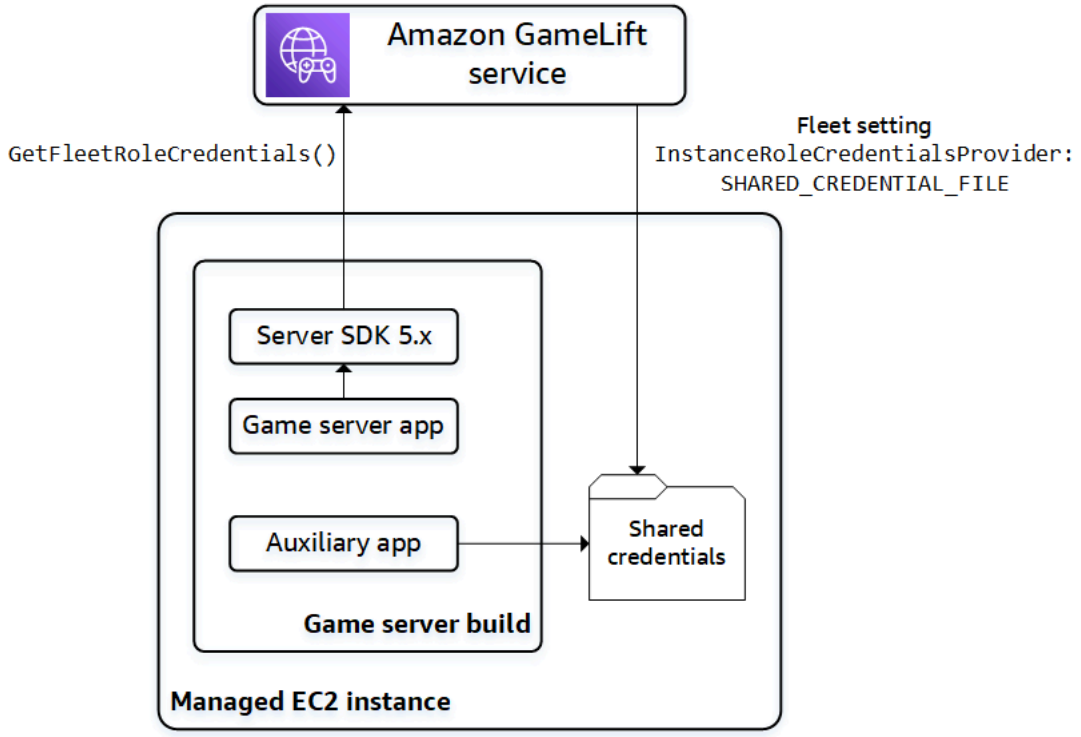
Nachdem Sie die Rolle erstellt haben, notieren Sie sich den Amazon-Ressourcennamen (ARN) der Rolle. Sie benötigen die Rolle ARN bei der Flottenerstellung.

Ändern Sie Anwendungen, um Anmeldeinformationen abzurufen

In diesem Schritt konfigurieren Sie Ihre Anwendungen so, dass sie Sicherheitsanmeldeinformationen für die IAM-Rolle abrufen und diese bei der Interaktion mit Ihren AWS Ressourcen verwenden.
In der folgenden Tabelle können Sie anhand (1) des Anwendungstyps und (2) der Server-SDK-Version, die Ihr Spiel für die Kommunikation mit Amazon GameLift verwendet, bestimmen, wie Sie Ihre Anwendungen ändern können.

	Spieleserver-Anwendungen	Andere Anwendungen
Verwenden Sie das Server-SDK Version 5.x	Rufen Sie die Server-SDK-Methode `GetFleetRoleCredentials()` von Ihrem Spielservercode aus auf.	Fügen Sie der Anwendung Code hinzu, um Anmeldeinformationen aus einer gemeinsam genutzten Datei auf der Flotteninstanz abzurufen.
Verwenden Sie das Server-SDK Version 4 oder früher	Call AWS Security Token Service (AWS STS) `AssumeRole` mit der Rolle ARN.	Call AWS Security Token Service (AWS STS) `AssumeRole` mit der Rolle ARN.

Für Spiele, die in das Server-SDK 5.x integriert sind, zeigt dieses Diagramm, wie Anwendungen in Ihrem bereitgestellten Spiel-Build Anmeldeinformationen für die IAM-Rolle abrufen können.



Rufen Sie **`GetFleetRoleCredentials()`** (Server-SDK 5.x) auf

Rufen Sie in Ihrem Spieleservercode, der bereits in das GameLift Amazon-Server-SDK 5.x integriert sein sollte, `GetFleetRoleCredentials` (C++) (C#) (Unreal) auf, um einen Satz temporärer Anmeldeinformationen abzurufen. Wenn die Anmeldeinformationen ablaufen, können Sie sie mit einem weiteren Aufruf von aktualisieren. `GetFleetRoleCredentials`

Verwenden Sie gemeinsam genutzte Anmeldeinformationen (Server-SDK 5.x)

Fügen Sie für Nicht-Serveranwendungen, die mit Spielserver-Builds mithilfe des Server-SDK 5.x bereitgestellt werden, Code hinzu, um die in einer gemeinsam genutzten Datei gespeicherten Anmeldeinformationen abzurufen und zu verwenden. Amazon GameLift generiert für jede Flotteninstanz ein Anmeldeinformationsprofil. Die Anmeldeinformationen können von allen Anwendungen auf der Instance verwendet werden. Amazon aktualisiert die temporären Anmeldeinformationen GameLift kontinuierlich.

Sie müssen eine Flotte so konfigurieren, dass die Datei mit den gemeinsamen Anmeldeinformationen bei der Flottenerstellung generiert wird.

Geben Sie in jeder Anwendung, die die Datei mit den gemeinsamen Anmeldeinformationen verwenden muss, den Speicherort und den Profilnamen wie folgt an:

Windows:

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "C:\\Credentials\\credentials"
```

Linux:

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "/local/credentials/credentials"
```

Beispiel: Richten Sie einen CloudWatch Agenten ein, der Metriken für GameLift Amazon-Flotteninstanzen sammelt

Wenn Sie einen CloudWatch Amazon-Agenten verwenden möchten, um Metriken, Protokolle und Rückverfolgungen aus Ihren GameLift Amazon-Flotten zu sammeln, verwenden Sie diese Methode, um den Agenten zu autorisieren, die Daten an Ihr Konto zu senden. Führen Sie in diesem Szenario die folgenden Schritte aus:

1. Rufen Sie die CloudWatch `config.json` Agentendatei ab oder schreiben Sie sie.
2. Aktualisieren Sie die `common-config.toml` Datei für den Agenten, um den Namen der Anmeldeinformationsdatei und den Profilnamen wie oben beschrieben zu identifizieren.
3. Richten Sie das Build-Installationsskript Ihres Gameservers ein, um den CloudWatch Agenten zu installieren und zu starten.

Verwenden Sie **`AssumeRole()`** (Server-SDK 4)

Fügen Sie Ihren Anwendungen Code hinzu, um die IAM-Rolle zu übernehmen und Anmeldeinformationen für die Interaktion mit Ihren AWS Ressourcen zu erhalten. Jede Anwendung, die auf einer GameLift Amazon-Flotteninstanz mit Server-SDK 4 oder früher ausgeführt wird, kann die IAM-Rolle übernehmen.

Im Anwendungscode muss die Anwendung vor dem Zugriff auf eine AWS Ressource die `AssumeRole` API-Operation AWS Security Token Service (AWS STS) aufrufen und den Rollen-ARN angeben. Dieser Vorgang gibt einen Satz temporärer Anmeldeinformationen zurück, die die Anwendung zum Zugriff auf die AWS Ressource autorisieren. Weitere Informationen finden Sie unter Verwenden von temporären Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.

Ordnen Sie der IAM-Rolle eine Flotte zu

Nachdem du die IAM-Rolle erstellt und die Anwendungen in deinem Gameserver-Build aktualisiert hast, um die Zugangsdaten abzurufen und zu verwenden, kannst du eine Flotte bereitstellen. Wenn du die neue Flotte konfigurierst, lege die folgenden Parameter fest:

- InstanceRoleArn— Setzen Sie diesen Parameter auf den ARN der IAM-Rolle.
- InstanceRoleCredentialsProvider— GameLift Um Amazon aufzufordern, eine Datei mit gemeinsamen Anmeldeinformationen für jede Flotteninstanz zu generieren, setzen Sie diesen Parameter auf`SHARED_CREDENTIAL_FILE`.

Sie müssen diese Werte festlegen, wenn Sie die Flotte erstellen. Sie können später nicht aktualisiert werden.

Greifen Sie mit VPC-Peering auf AWS Ressourcen zu

Sie können Amazon Virtual Private Cloud (Amazon VPC) -Peering verwenden, um zwischen Anwendungen, die auf einer GameLift Amazon-Instance ausgeführt werden, und einer anderen AWS Ressource zu kommunizieren. Eine VPC ist ein von Ihnen definiertes virtuelles privates Netzwerk, das eine Reihe von Ressourcen umfasst, die über Ihr AWS-Konto verwaltet werden. Jede GameLift Amazon-Flotte hat ihre eigene VPC. Mit VPC-Peering können Sie eine direkte Netzwerkverbindung zwischen der VPC für Ihre Flotte und für Ihre anderen Ressourcen herstellen. AWS

Amazon GameLift optimiert den Prozess der Einrichtung von VPC-Peering-Verbindungen für Ihre Spieleserver. Es übernimmt Peering-Anfragen, aktualisiert Routing-Tabellen und konfiguriert die Verbindungen nach Bedarf. Anweisungen zum Einrichten von VPC-Peering für Ihre Spieleserver finden Sie unter. VPC-Peering für Amazon GameLift

Integrieren Sie Ihren Spielclient in Amazon GameLift

Die Themen in diesem Abschnitt beschreiben die verwalteten GameLift Amazon-Funktionen, die Sie einem Backend-Service hinzufügen können. Ein Backend-Service erledigt die folgenden Aufgaben:

- Fordert Informationen über aktive Spielesitzungen von Amazon anGameLift.
- Verbinden eines Spielers mit einer bestehenden Spielsitzung.
- Erzeugt eine neue Spielsitzung und nimmt Spieler daran teil.
- Ändert die Metadaten für eine bestehende Spielsitzung.

Weitere Informationen darüber, wie Spieleclients mit Amazon GameLift und den auf Amazon laufenden Spieleservern interagierenGameLift, finden Sie unterInteraktionen zwischen Amazon GameLift und Game-Client und Server.

Voraussetzungen

- Eine AWS-Konto
- Ein Gameserver-Build, der auf Amazon hochgeladen wurdeGameLift.
- Eine Flotte für das Hosten deiner Spiele.

Themen

Füge Amazon GameLift zu deinem Spielclient hinzu

Integrieren Sie Amazon GameLift in Spielkomponenten, die Informationen zu Spielsitzungen benötigen, erstellen Sie neue Spielsitzungen und fügen Sie Spieler zu Spielen hinzu. Abhängig von Ihrer Spielarchitektur ist diese Funktionalität in Backend-Diensten enthalten, die Aufgaben wie Spielerauthentifizierung, Spielersuche oder Platzierung von Spielsitzungen übernehmen.

> **Note**
>
> Detaillierte Informationen zur Einrichtung von Matchmaking für Ihr von Amazon GameLift gehostetes Spiel finden Sie im Amazon GameLift FlexMatch Developer Guide.

Amazon GameLift auf einem Backend-Service einrichten

Fügen Sie Code hinzu, um einen GameLift Amazon-Client zu initialisieren und wichtige Einstellungen zu speichern. Dieser Code muss vor jedem Code ausgeführt werden, der von Amazon abhängig ist GameLift.

1. Richten Sie eine Client-Konfiguration ein. Verwenden Sie die Standard-Client-Konfiguration oder erstellen Sie ein benutzerdefiniertes Client-Konfigurationsobjekt. Weitere Informationen finden Sie unter AWS::Client::ClientConfiguration(C++) oder AmazonGameLiftConfig(C#).

Eine Client-Konfiguration spezifiziert eine Zielregion und einen Endpunkt, die bei der Kontaktaufnahme mit Amazon verwendet GameLift werden sollen. Die Region gibt die Menge der bereitgestellten Ressourcen (Flotten, Warteschlangen und Matchmaker) an, die verwendet werden sollen. In der Standard-Client-Konfiguration ist der Standort auf die Region USA Ost (Nord-Virginia) festgelegt. Um eine andere Region zu verwenden, erstellen Sie eine benutzerdefinierte Konfiguration.

2. Initialisieren Sie einen GameLift Amazon-Client. Verwenden Sie Aws:GameLift::: GameLiftClient () (C++) oder AmazonGameLiftClient() (C#) mit einer Standard-Client-Konfiguration oder einer benutzerdefinierten Client-Konfiguration.
3. Fügen Sie einen Mechanismus hinzu, um eine eindeutige Kennung für jeden Spieler zu generieren. Weitere Informationen finden Sie unter Generieren Sie Spieler-IDs.
4. Erfassen und speichern Sie die folgenden Informationen:

   - Zielflotte — Bei vielen GameLift Amazon-API-Anfragen muss eine Flotte angegeben werden. Verwenden Sie dazu entweder eine Flotten-ID oder eine Alias-ID, die auf die Zielflotte verweist. Es hat sich bewährt, Flottenaliase zu verwenden, damit Sie Spieler von einer Flotte zur anderen wechseln können, ohne Ihre Backend-Dienste aktualisieren zu müssen.
   - Zielwarteschlange — Geben Sie für Spiele, die Warteschlangen mit mehreren Flotten verwenden, um neue Spielsitzungen zu platzieren, den Namen der Warteschlange an, die verwendet werden soll.
   - AWS Anmeldeinformationen — Bei allen Anrufen bei Amazon GameLift müssen Anmeldeinformationen für den AWS-Konto Hoster des Spiels angegeben werden. Sie erwerben diese Anmeldeinformationen, indem Sie einen Spielerbenutzer erstellen, wie unter beschriebenRichte den programmatischen Zugriff für dein Spiel ein. Gehen Sie je nachdem, wie Sie den Zugriff für den Player-Benutzer verwalten, wie folgt vor:
     - Wenn Sie eine Rolle zur Verwaltung der Benutzerberechtigungen von Spielern verwenden, fügen Sie Code hinzu, um die Rolle zu übernehmen, bevor Sie eine GameLift Amazon-API aufrufen. Die Anfrage zur Übernahme der Rolle gibt einen Satz temporärer Sicherheitsanmeldedaten zurück. Weitere Informationen finden Sie unter Wechseln zu einer IAM-Rolle (AWS API) im IAM-Benutzerhandbuch.
     - Wenn Sie über langfristige Sicherheitsanmeldedaten verfügen, konfigurieren Sie Ihren Code so, dass gespeicherte Anmeldeinformationen gefunden und verwendet werden. Weitere Informationen finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. Informationen zum Speichern von Anmeldeinformationen finden Sie in den AWS API-Referenzen für (C++) und (.NET).

- Wenn Sie über temporäre Sicherheitsanmeldedaten verfügen, fügen Sie mithilfe von AWS Security Token Service (AWS STS) Code hinzu, um die Anmeldeinformationen regelmäßig zu aktualisieren, wie unter Verwenden temporärer Sicherheitsanmeldedaten mit den AWS SDKs im IAM-Benutzerhandbuch beschrieben. Der Code muss neue Anmeldeinformationen anfordern, bevor die alten ablaufen.

Holen Sie sich Spielesitzungen

Füge Code hinzu, um verfügbare Spielsitzungen zu entdecken und die Einstellungen und Metadaten der Spielsitzungen zu verwalten.

Suche nach aktiven Spielsitzungen

SearchGameSessionsDient zum Abrufen von Informationen zu einer bestimmten Spielsitzung, zu allen aktiven Sitzungen oder zu Sitzungen, die einer Reihe von Suchkriterien entsprechen. Dieser Aufruf gibt für jede aktive Spielsitzung ein GameSessionObjekt zurück, das Ihrer Suchanfrage entspricht.

Verwenden Sie Suchkriterien, um eine gefilterte Liste aktiver Spielsitzungen abzurufen, an denen Spieler teilnehmen können. Beispielsweise können Sie Sitzungen wie folgt filtern:

- Ausgeschlossen sind Spielsitzungen, die voll sind:`CurrentPlayerSessionCount = MaximumPlayerSessionCount`.
- Wählen Sie Spielsitzungen basierend auf der Dauer der Sitzung aus: Auswerten`CreationTime`.
- Finden Sie Spielsitzungen, die auf einer benutzerdefinierten Spieleigenschaft basieren:`gameSessionProperties.gameMode = "brawl"`.

Spielsitzungen verwalten

Verwenden Sie eine der folgenden Operationen, um Informationen zu Spielsitzungen abzurufen oder zu aktualisieren.

- DescribeGameSessionDetails() — Ruft zusätzlich zu den Informationen zur Spielsitzung den Schutzstatus einer Spielsitzung ab.
- UpdateGameSession() — Ändern Sie die Metadaten und Einstellungen einer Spielsitzung nach Bedarf.
- GetGameSessionLogUrl— Greifen Sie auf gespeicherte Spielsitzungsprotokolle zu.

Spielsitzungen erstellen

Fügen Sie Code zum Starten neuer Spielsitzungen auf den von Ihnen bereitgestellten Flotten hinzu und stellen Sie diese Spielern zur Verfügung. Es gibt zwei Möglichkeiten, Spielsitzungen zu erstellen, je nachdem, ob du dein Spiel in mehreren AWS-Regionen oder in einer einzigen Region bereitstellst.

Erstelle eine Spielsitzung in einer Warteschlange mit mehreren Standorten

Wird verwendet StartGameSessionPlacement, um eine Anfrage für eine neue Spielsitzung in eine Warteschlange zu stellen. Um diesen Vorgang zu verwenden, erstellen Sie eine Warteschlange. Dies bestimmt, wo Amazon die neue Spielsitzung GameLift platziert. Weitere Informationen zu Warteschlangen und deren Verwendung finden Sie unterGameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten.

Wenn Sie eine Platzierung für Spielsitzungen erstellen, geben Sie den Namen der zu verwendenden Warteschlange, einen Namen für die Spielsitzung, eine maximale Anzahl gleichzeitiger Spieler und einen optionalen Satz von Spieleigenschaften an. Sie können optional auch eine Liste von Spielern angeben, die automatisch an der Spielsitzung teilnehmen sollen. Wenn Sie Daten zur Spielerlatenz für relevante Regionen angeben, GameLift verwendet Amazon diese Informationen, um die neue Spielsitzung auf einer Flotte zu platzieren, die den Spielern das ideale Spielerlebnis bietet.

Die Platzierung von Spielsitzungen ist ein asynchroner Prozess. Nachdem Sie eine Anfrage gestellt haben, können Sie sie erfolgreich ausführen lassen oder eine Zeitüberschreitung einleiten. Sie können die Anfrage auch jederzeit stornieren, indem Sie StopGameSessionPlacement. Rufen Sie an, um den Status Ihrer Platzierungsanfrage zu überprüfen DescribeGameSessionPlacement.

Erstelle eine Spielsitzung auf einer bestimmten Flotte

Wird verwendet CreateGameSession, um eine neue Sitzung auf einer bestimmten Flotte zu erstellen. Diese synchrone Operation gelingt oder schlägt fehl, abhängig davon, ob die Flotte über Ressourcen für das Hosten einer neuen Spielsitzung verfügt. Nachdem Amazon die neue Spielsitzung GameLift erstellt und ein GameSessionObjekt zurückgegeben hat, können Sie Spieler zu ihr hinzufügen.

Wenn Sie diesen Vorgang verwenden, geben Sie eine Flotten- oder Alias-ID, einen Sitzungsnamen und die maximale Anzahl gleichzeitiger Spieler für das Spiel an. Optional können Sie eine Gruppe von Spieleigenschaften einschließen. Spieleigenschaften werden in einer Reihe von Schlüssel-Wert-Paaren definiert.

Wenn Sie die Amazon GameLift Resource Protection-Funktion verwenden, um die Anzahl der Spielsitzungen zu begrenzen, die ein Spieler erstellen kann, geben Sie die Spieler-ID des Erstellers der Spielsitzung an.

Nehmen Sie mit einem Spieler an einer Spielsitzung teil

Füge Code hinzu, um einen Spielerplatz in einer aktiven Spielsitzung zu reservieren und Spielclients mit Spielsitzungen zu verbinden.

1. Reservieren Sie einen Spielerplatz in einer Spielsitzung

   Um einen Spielerplatz zu reservieren, erstellen Sie eine neue Spielersitzung für die Spielsitzung. Weitere Informationen zu Spielersitzungen finden Sie unterSo verbinden sich Spieler mit Spielen.

   Es gibt zwei Möglichkeiten, neue Spielersitzungen zu erstellen:

   - StartGameSessionPlacementDient zum Reservieren von Slots für einen oder mehrere Spieler in der neuen Spielsitzung.
   - Reservieren Sie Spielerplätze für einen oder mehrere Spieler, die eine Spielsitzungs-ID verwenden CreatePlayerSessionoder CreatePlayerSessionsmit einer Spielsitzungs-ID ausgestattet sind.

   Amazon überprüft GameLift zunächst, ob die Spielsitzung neue Spieler akzeptiert und ob Spielerplätze verfügbar sind. Bei Erfolg GameLift reserviert Amazon einen Slot für den Spieler, erstellt die neue Spielersitzung und gibt ein PlayerSessionObjekt zurück. Dieses Objekt enthält den DNS-Namen, die IP-Adresse und den Port, die ein Spielclient benötigt, um eine Verbindung zur Spielsitzung herzustellen.

   Eine Spielersitzungsanfrage muss eine eindeutige ID für jeden Spieler enthalten. Weitere Informationen finden Sie unter Generieren Sie Spieler-IDs.

   Eine Spielersitzung kann eine Reihe von benutzerdefinierten Spielerdaten enthalten. Diese Daten werden im neu erstellten Spielersitzungsobjekt gespeichert, das Sie durch Aufrufen von DescribePlayerSessions() abrufen können. Amazon übergibt dieses Objekt GameLift auch an den Spieleserver, wenn der Spieler eine direkte Verbindung zur Spielsitzung herstellt. Wenn Sie Sitzungen mit mehreren Spielern anfordern, geben Sie für jeden Spieler eine Reihe von Spielerdaten an, die der Spieler-ID in der Anfrage zugeordnet sind.

2. Mit einer Spielsitzung verbinden

   Fügen Sie dem Spiele-Client Code hinzu, um das `PlayerSession`-Objekt abzurufen, das die Verbindungsinformationen der Spielsitzung enthält. Verwenden Sie diese Informationen, um eine direkte Verbindung zum Server herzustellen.

- Sie können eine Verbindung über den angegebenen Port und den DNS-Namen oder die IP-Adresse herstellen, die dem Serverprozess zugewiesen wurde.
- Wenn für Ihre Flotten die Generierung von TLS-Zertifikaten aktiviert ist, stellen Sie mithilfe des DNS-Namens und -Ports eine Verbindung her.
- Wenn dein Spieleserver eingehende Spielerverbindungen validiert, gib die Sitzungs-ID des Spielers an.

Nach dem Herstellen der Verbindung kommunizieren der Spielclient und der Serverprozess direkt, ohne Amazon einzubeziehen GameLift. Der Server unterhält die Kommunikation mit Amazon GameLift, um den Verbindungsstatus der Spieler, den Gesundheitsstatus und mehr zu melden. Wenn der Spieleserver eingehende Spieler bestätigt, überprüft er, ob die Sitzungs-ID des Spielers mit einem reservierten Slot in der Spielsitzung übereinstimmt, und akzeptiert oder verweigert die Spielerverbindung. Wenn der Spieler die Verbindung trennt, meldet der Serverprozess den Verbindungsabbruch.

Verwenden Sie die Eigenschaften der Spielsitzung

Ihr Spielclient kann mithilfe einer Spieleigenschaft Daten an eine Spielsitzung weitergeben. Spieleigenschaften sind Schlüssel-Wert-Paare, die dein Spieleserver hinzufügen, lesen, auflisten und ändern kann. Sie können eine Spieleigenschaft übergeben, wenn Sie eine neue Spielsitzung erstellen, oder zu einem späteren Zeitpunkt, wenn die Spielsitzung aktiv ist. Eine Spielsitzung kann bis zu 16 Spieleigenschaften enthalten. Spieleigenschaften können nicht gelöscht werden.

Ihr Spiel bietet beispielsweise die folgenden Schwierigkeitsstufen: `NoviceEasy`,`Intermediate`, und`Expert`. Ein Spieler wählt `Easy` und beginnt dann das Spiel. Ihr Spielclient fordert eine neue Spielsitzung von Amazon an, GameLift indem Sie entweder `StartGameSessionPlacement` oder, `CreateGameSession` wie in den vorherigen Abschnitten beschrieben, verwenden. In der Anfrage übergibt der Client Folgendes:`{"Key": "Difficulty", "Value":"Easy"}`.

Als Antwort auf die Anfrage GameLift erstellt Amazon ein `GameSession` Objekt, das die angegebene Spieleigenschaft enthält. Amazon weist GameLift dann einen verfügbaren Spieleserver an, die neue Spielsitzung zu starten, und übergibt das `GameSession` Objekt. Der Spieleserver startet eine Spielsitzung mit einem `Difficulty` von`Easy`.

Weitere Informationen

- GameProperty Datentyp

- SearchGameSessions() Beispiele
- UpdateGameSession() GameProperties Parameter

Generieren Sie Spieler-IDs

Amazon GameLift verwendet eine Spielersitzung, um einen Spieler darzustellen, der mit einer Spielsitzung verbunden ist. Amazon GameLift erstellt jedes Mal eine Spielersitzung, wenn ein Spieler mithilfe eines in Amazon integrierten Spielclients eine Verbindung zu einer Spielsitzung herstelltGameLift. Wenn ein Spieler ein Spiel verlässt, endet die Spielersitzung. Amazon verwendet Spielersitzungen GameLift nicht wieder.

Das folgende Codebeispiel generiert nach dem Zufallsprinzip eindeutige Spieler-IDs:

```
bool includeBrackets = false;
bool includeDashes = true;
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
 includeDashes);
```

Weitere Informationen zu Spielersitzungen finden Sie unterDaten zu Spiel- und Spielersitzungen anzeigen.

Game-Engines und Amazon GameLift

Sie können den verwalteten GameLift Amazon-Dienst mit den meisten gängigen Game-Engines verwenden, die C++- oder C#-Bibliotheken unterstützen, einschließlich O3DE, Unreal Engine und Unity. Erstellen Sie die Version, die Sie für Ihr Spiel benötigen. Informationen zu Build-Anweisungen und Mindestanforderungen finden Sie in den README-Dateien jeder Version. Weitere Informationen zu verfügbaren Amazon GameLift SDKs, unterstützten Entwicklungsplattformen und Betriebssystemen finden Sie unter Entwicklungsunterstützung mit Amazon GameLift Für Gameserver.

Zusätzlich zu den maschinenspezifischen Informationen in diesem Thema finden Sie in den folgenden Themen weitere Hilfe zur Integration von Amazon GameLift in Ihre Spieleserver, Clients und Dienste:

- Roadmap für GameLift verwaltetes Hosting von Amazon— Ein sechsstufiger Workflow für die erfolgreiche Integration von Amazon GameLift in Ihr Spiel und die Einrichtung von Hosting-Ressourcen.

- Füge Amazon GameLift zu deinem Gameserver hinzu— Detaillierte Anweisungen zur Integration von Amazon GameLift in einen Gameserver.
- Füge Amazon GameLift zu deinem Spielclient hinzu – Detaillierte Anleitung zur Integration in einen Spiele-Client oder -Service, einschließlich der Erstellung von Spiel-Sitzungen und Einbindung von Spielern in Spiele.

O3DE

Spiel-Server

Bereite deine Gameserver GameLift mithilfe des Amazon GameLiftServer SDK für C++ auf das Hosting bei Amazon vor. Weitere Hilfe bei der Integration der benötigten Funktionalitäten in Ihren Spiel-Server finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Spiele-Clients und -Services

Ermöglichen Sie Ihren Spielclients und/oder Spieleddiensten, mit dem GameLift Amazon-Dienst zu interagieren, z. B. um verfügbare Spielsitzungen zu finden oder neue zu erstellen und Spieler zu Spielen hinzuzufügen. Die wichtigsten Client-Funktionen werden im AWSSDK für C++ bereitgestellt. Informationen zur Integration von Amazon GameLift in Ihr O3DE-Spieleprojekt finden Sie unter Amazon GameLift zu einem O3DE-Spielclient und -Server hinzufügen und. Füge Amazon GameLift zu deinem Spielclient hinzu

Unreal Engine

Spiel-Server

Bereite deine Gameserver für das Hosting bei Amazon vor, GameLift indem du das Amazon GameLift Server SDK für Unreal Engine zu deinem Projekt hinzufügst und die erforderliche Serverfunktionalität implementierst. Hilfe beim Einrichten des Unreal Engine-Plug-ins und Hinzufügen von GameLift Amazon-Code finden Sie unterIntegrieren Sie Amazon GameLift in ein Unreal Engine-Projekt.

Spiele-Clients und -Services

Ermöglichen Sie Ihren Spielclients und/oder Spieleddiensten, mit dem GameLift Amazon-Dienst zu interagieren, z. B. um verfügbare Spielsitzungen zu finden oder neue zu erstellen und Spieler zu Spielen hinzuzufügen. Die wichtigsten Client-Funktionen werden im AWSSDK für C++ bereitgestellt. Informationen zur Integration von Amazon GameLift in Ihr Unreal Engine-Spieleprojekt finden Sie unterFüge Amazon GameLift zu deinem Spielclient hinzu.

Unity

Spiel-Server

Bereiten Sie Ihre Gameserver für das Hosting bei Amazon vor, GameLift indem Sie das Amazon GameLift Server SDK für C# zu Ihrem Projekt hinzufügen und die erforderlichen Serverfunktionen implementieren. Hilfe beim Einrichten mit Unity und Hinzufügen von GameLift Amazon-Code finden Sie unterIntegrieren Sie Amazon GameLift in ein Unity-Projekt.

Spiele-Clients und -Services

Ermöglichen Sie Ihren Spielclients und/oder Spielediensten, mit dem GameLift Amazon-Dienst zu interagieren, z. B. um verfügbare Spielsitzungen zu finden oder neue zu erstellen und Spieler zu Spielen hinzuzufügen. Die Kernclientfunktionalität wird über AWS SDK for .NET bereitgestellt. Informationen zur Integration von Amazon GameLift in Ihr Unity-Spieleprojekt finden Sie unterFüge Amazon GameLift zu deinem Spielclient hinzu.

Andere Motoren

Eine vollständige Liste der Amazon GameLift SDKs, die für Spieleserver und Clients verfügbar sind, finden Sie unterthe section called “Unterstützung bei der Entwicklung”.

Amazon GameLift zu einem O3DE-Spielclient und -Server hinzufügen

Sie können O3DE verwenden, eine plattformübergreifende Open-Source-3D-Engine in Echtzeit, um leistungsstarke interaktive Erlebnisse zu erstellen, einschließlich Spielen und Simulationen. Der O3DE-Renderer und die Tools sind in ein modulares Framework eingebettet, das Sie mit Ihren bevorzugten Entwicklungstools modifizieren und erweitern können.

Das modulare Framework verwendet Gems, die Bibliotheken mit Standardschnittstellen und Assets enthalten. Wähle deine eigenen Edelsteine aus, um zu entscheiden, welche Funktionen du je nach deinen Anforderungen hinzufügen möchtest.

Das Amazon GameLift Gem bietet die folgenden Funktionen:

GameLiftAmazon-Integration

Ein Framework zur Erweiterung der O3DE-Netzwerkebene und zur Verwendung des Multiplayer-Gems mit der GameLift dedizierten Serverlösung von Amazon. Das Gem bietet Integrationen sowohl mit dem Amazon GameLift Server SDK als auch mit dem AWS SDK-Client (um den GameLift Amazon-Dienst selbst aufzurufen).

Build- und Paketverwaltung

Anweisungen zum Verpacken und optionalen Hochladen des dedizierten Server-Builds und einer AWS Cloud Development Kit (AWS CDK) (AWS CDK) -Anwendung zum Einrichten und Aktualisieren von Ressourcen.

Amazon GameLift Gem einrichten

Folgen Sie den Verfahren in diesem Abschnitt, um das Amazon GameLift Gem in O3DE einzurichten.

Voraussetzungen

- Richten Sie Ihr AWS Konto für Amazon einGameLift. Weitere Informationen finden Sie unter Richten Sie ein AWS-Konto.
- Richten Sie AWS Anmeldeinformationen für O3DE ein. Weitere Informationen finden Sie unter Konfiguration von AWS Anmeldeinformationen.
- Richte das AWS CLI und einAWS CDK. Für weitere Informationen, AWS Command Line Interfaceund AWS Cloud Development Kit (AWS CDK).

Schalten Sie das Amazon GameLift Gem und seine Abhängigkeiten ein

1. Öffnen Sie den Projektmanager.
2. Öffnen Sie das Menü unter Ihrem Projekt und wählen Sie Projekteinstellung bearbeiten... .
3. Wähle „Edelsteine konfigurieren“.
4. Schalten Sie das Amazon GameLift Gem und die folgenden abhängigen Gems ein:
   - AWSCore Gem — Stellen Sie das Framework für die Verwendung AWS-Services in O3DE bereit.
   - Multiplayer Gem — Bietet Multiplayer-Funktionalität durch Erweiterung des Netzwerk-Frameworks.

Fügen Sie die statische Amazon GameLift Gem-Bibliothek ein

1. Fügen Sie die `Gem::AWSGameLift.Server.Static` als `BUILD_DEPENDENCIES` für Ihr Project Server-Ziel ein.

```
ly_add_target(
```

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

2. Stellen `AWSGameLiftService` Sie für Ihre Project Server-Systemkomponente die Option Erforderlich ein.

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

3. (Optional) Um GameLift Amazon-Serviceanfragen in C++ zu stellen, fügen Sie sie `Gem::AWSGameLift.Client.Static` in das `BUILD_DEPENDENCIES` für Ihr Kundenziel ein.

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

Integriere dein Spiel und deinen dedizierten Server

Verwalte Spielsitzungen innerhalb deines Spiels und auf deinem dedizierten Spielserver mit der Session Management Integration. Informationen zur Unterstützung FlexMatch finden Sie unter FlexMatchIntegration.

Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt

In diesem Thema wird erklärt, wie Sie das Amazon GameLift C++-Server-SDK-Plugin für Unreal Engine einrichten und in Ihre Spieleprojekte integrieren.

Zusätzliche Ressourcen:

- Server-SDK-Plugin für die Unreal-Download-Seite
- Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x
- the section called “Unterstützung bei der Entwicklung”

Voraussetzungen

Bevor Sie fortfahren, stellen Sie sicher, dass Sie die folgenden Voraussetzungen überprüft haben:

Voraussetzungen

- Ein Computer, auf dem Unreal Engine ausgeführt werden kann. Weitere Informationen zu den Anforderungen der Unreal Engine finden Sie in der Dokumentation zu den Hardware- und Softwarespezifikationen von Unreal Engine.
- Microsoft Visual Studio 2019 oder neuere Version.
- CMake Version 3.1 oder höher.
- Python-Version 3.6 oder höher.
- Ein Git-Client, der auf dem PATH verfügbar ist.
- Ein Epic-Games-Konto. Eröffnen Sie ein Konto auf der offiziellen Unreal Engine-Website.
- Ein GitHub Konto, das mit Ihrem Unreal Engine-Konto verknüpft ist. Weitere Informationen finden Sie unter Zugreifen auf den Unreal Engine-Quellcode GitHub auf der Unreal Engine-Website.

> **Note**
>
> Amazon unterstützt GameLift derzeit die folgenden Versionen von Unreal Engine:
>
> - 4.22

- 4,23
- 4,24
- 4,25
- 4,26
- 4,27
- 5.1.0
- 5.1.1
- 5.2
- 5.3

Unreal Engine aus dem Quellcode erstellen

Standardversionen des Unreal Engine-Editors, die über den Epic Launcher heruntergeladen wurden, erlauben nur das Erstellen von Unreal-Client-Anwendungen. Um eine Unreal-Serveranwendung zu erstellen, müssen Sie die Unreal Engine mithilfe des Unreal Engine Github-Repos aus dem Quellcode herunterladen und erstellen. Weitere Informationen finden Sie im Tutorial Unreal Engine from Source erstellen auf der Unreal Engine-Dokumentationswebsite.

> **Note**
>
> Falls du das noch nicht getan hast, befolge die Anweisungen unter Zugriff auf den Unreal Engine-Quellcode unter, GitHub um dein Konto mit deinem Epic GitHub Games-Konto zu verknüpfen.

Um den Unreal Engine-Quellcode in deine Entwicklungsumgebung zu klonen

1. Klonen Sie die Unreal Engine-Quelle in Ihre Entwicklungsumgebung in einem Zweig Ihrer Wahl.

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

2. Sieh dir das Tag der Version an, die du zur Entwicklung deines Spiels verwendest. Im folgenden Beispiel wird beispielsweise die Unreal Engine-Version 5.1.1 ausgecheckt:

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

3. Navigieren Sie zum Stammordner des lokalen Repositorys. Wenn Sie sich im Stammordner befinden, führen Sie die folgende Datei aus:`Setup.bat`.
4. Führen Sie im Stammordner auch die Datei aus:`GenerateProjectFiles.bat`.
5. Nachdem Sie die Dateien aus den vorherigen Schritten ausgeführt haben, wird eine Unreal Engine-Lösungsdatei,`UE5.sln`, erstellt. Öffnen Sie Visual Studio und öffnen Sie die `UE5.sln` Datei im Visual Studio-Editor.
6. Öffnen Sie in Visual Studio das Menü Ansicht und wählen Sie die Option Solution Explorer. Dadurch wird das Kontextmenü des Unreal-Projektknotens geöffnet. Klicken Sie im Solution Explorer-Fenster mit der rechten Maustaste auf die `UE5.sln` Datei (sie kann auch als „Nur“ aufgeführt werden`UE5`) und wählen Sie dann Build, um das Unreal-Projekt mit dem Win64-Ziel des Development Editors zu erstellen.

   > **Note**
   >
   > Es kann über eine Stunde dauern, bis der Build abgeschlossen ist.

Sobald der Build abgeschlossen ist, können Sie den Unreal Development Editor öffnen und ein Projekt erstellen oder importieren.

Konfiguriere dein Unreal-Projekt für das Plugin

Folgen Sie diesen Schritten, um das Amazon GameLift Server-SDK-Plugin für Unreal Engine für Ihre Gameserver-Projekte vorzubereiten.

Um dein Projekt für das Plugin zu konfigurieren

1. Navigieren Sie bei geöffnetem Visual Studio zum Solution Explorer-Bereich und wählen Sie die `UE5` Datei aus, um das Kontextmenü für das Unreal-Projekt zu öffnen. Wählen Sie im Kontextmenü die Option Als Startprojekt festlegen.
2. Wählen Sie oben in Ihrem Visual Studio-Fenster die Option Debugging starten (grüner Pfeil).

   Diese Aktion startet Ihre neue, im Quellcode erstellte Instanz von Unreal Editor. Weitere Informationen zur Verwendung des Unreal Editors finden Sie unter Unreal Editor Interface auf der Unreal Engine-Dokumentationswebsite.
3. Schließen Sie das Visual Studio-Fenster, das Sie geöffnet haben, da der Unreal Editor ein weiteres Visual Studio-Fenster öffnet, das das Unreal-Projekt und Ihr Spielprojekt enthält.

4. Führen Sie im Unreal-Editor einen der folgenden Schritte aus:
   - Wählen Sie ein vorhandenes Unreal-Projekt aus, das Sie in Amazon GameLift integrieren möchten.
   - Erstellt ein neues Projekt. Um mit dem GameLift Amazon-Plugin für Unreal zu experimentieren, versuchen Sie es mit der Third-Person-Vorlage der Unreal Engine. Weitere Informationen zu dieser Vorlage finden Sie unter Vorlage für Dritte auf der Unreal Engine-Dokumentationswebsite.

     Alternativ können Sie ein neues Projekt mit den folgenden Einstellungen konfigurieren:

     - C++
     - Mit Starter-Inhalten
     - Desktop
     - Ein Projektname. In den Beispielen zu diesem Thema haben wir unserem Projekt einen Namen gegeben`GameLiftUnrealApp`.
5. Navigieren Sie im Solution Explorer von Visual Studio zum Speicherort Ihres Unreal-Projekts. Suchen Sie im `Source` Ordner Unreal nach einer Datei mit dem Namen. *`Your-application-name`*`.Target.cs`

   Zum Beispiel: `GameLiftUnrealApp.Target.cs`.
6. Erstellen Sie eine Kopie dieser Datei und geben Sie der Kopie einen Namen:*`Your-application-name`*`Server.Target.cs`.
7. Öffnen Sie die neue Datei und nehmen Sie die folgenden Änderungen vor:
   - Ändern Sie das `class` und so`constructor`, dass es dem Dateinamen entspricht.
   - Ändern Sie das `Type` von `TargetType.Game` zu`TargetType.Server`.
   - Die endgültige Datei wird wie das folgende Beispiel aussehen:

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

```
    }
```

Ihr Projekt ist jetzt so konfiguriert, dass es das Amazon GameLift Server SDK-Plugin akzeptiert.

Die nächste Aufgabe besteht darin, die C++-Server-SDK-Bibliotheken für Unreal zu erstellen, damit Sie sie in Ihr Projekt importieren können.

Um die C++-Server-SDK-Bibliotheken für Unreal zu erstellen

1. Laden Sie das Amazon GameLift C++-Server-SDK-Plugin für Unreal herunter.

   > **Note**
   >
   > Das Speichern des SDK in das Standard-Download-Verzeichnis kann zu einem Build-Fehler führen, da der Pfad die Obergrenze von 260 Zeichen überschreitet. Beispiel: `C:\Users\Administrator\Downloads\GameLift-SDK-Release-06_15_2023\GameLift-Cpp-ServerSDK-5.0.4`
   > Wir empfehlen, das SDK beispielsweise in ein anderes Verzeichnis zu verschieben`C:\GameLift-Cpp-ServerSDK-5.0.4`.

2. Laden Sie OpenSSL herunter und installieren Sie es. Weitere Informationen zum Herunterladen von OpenSSL finden Sie in der Github OpenSSL-Build- und Installationsdokumentation.

   Weitere Informationen finden Sie in der Dokumentation zu OpenSSL Notes für Windows-Plattformen.

   > **Note**
   >
   > Die Version von OpenSSL, die Sie zum Erstellen des GameLift Amazon-Server-SDK verwenden, sollte mit der Version von OpenSSL übereinstimmen, die von Unreal zum Paketieren Ihres Spieleservers verwendet wird. Versionsinformationen finden Sie im Unreal-Installationsverzeichnis. `...Engine\Source\ThirdParty\OpenSSL`

3. Erstellen Sie mit den heruntergeladenen Bibliotheken die C++-Server-SDK-Bibliotheken für Unreal Engine.

   Kompilieren Sie im `GameLift-Cpp-ServerSDK-<version>` Verzeichnis im heruntergeladenen SDK mit dem `-DBUILD_FOR_UNREAL=1` Parameter und erstellen Sie das Server-SDK. Die folgenden Beispiele zeigen, wie Sie mit kompilieren`cmake`.

Führen Sie die folgenden Befehle in Ihrem Terminal aus:

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

Der Windows-Build erstellt die folgenden Binärdateien in dem `out\gamelift-server-sdk\Release` Ordner:

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

Kopieren Sie die beiden Bibliotheksdateien in den `ThirdParty\GameLiftServerSDK\Win64` Ordner im Amazon GameLift Unreal Engine-Plugin-Paket.

Gehen Sie wie folgt vor, um das GameLift Amazon-Plugin in Ihr Beispielprojekt zu importieren.

Importieren Sie das GameLift Amazon-Plugin

1. Suchen Sie den `GameLiftServerSDK` Ordner, den Sie im vorherigen Verfahren aus dem Plugin extrahiert haben.
2. Suchen Sie den Stammordner `Plugins` in Ihrem Spielprojekt. (Falls der Ordner nicht existiert, erstelle ihn dort.)
3. Kopieren Sie den `GameLiftServerSDK` Ordner in den`Plugins`.

   Dadurch kann das Unreal-Projekt das Plugin sehen.
4. Füge das Amazon GameLift Server SDK Plugin zur `.uproject` Datei des Spiels hinzu.

   Im Beispiel wird die App aufgerufen`GameLiftUnrealApp`, also die Datei`GameLiftUnrealApp.uproject`.
5. Bearbeiten Sie die `.uproject` Datei, um das Plugin zu Ihrem Spielprojekt hinzuzufügen.

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

```
    }
]
```

6. Stellen Sie sicher, dass ModuleRules das Spiel vom Plugin abhängig ist. Öffnen Sie die `.Build.cs` Datei und fügen Sie die Amazon GameLiftServer SDK-Abhängigkeit hinzu. Diese Datei finden Sie unter*Your-application-name*/Source/*/Your-application-name/*.

   Der Dateipfad für das Tutorial lautet beispielsweise. `../GameLiftUnrealApp/Source/GameLiftUnrealApp/GameLiftUnrealApp.Build.cs`

7. Am Ende der Liste von hinzufügen`"GameLiftServerSDK"`. `PublicDependencyModuleNames`

```
using UnrealBuildTool;
using System.Collections.Generic;
public class GameLiftUnrealApp : ModuleRules
  {
    public GameLiftUnrealApp(TargetInfo Target)
    {
      PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
 "Engine", "InputCore", "GameLiftServerSDK" });
      bEnableExceptions = true;
    }
  }
```

Das Plugin sollte jetzt für Ihre Anwendung funktionieren. Fahren Sie mit dem nächsten Abschnitt fort, um GameLift Amazon-Funktionen in Ihr Spiel zu integrieren.

Fügen Sie Ihrem Unreal-Projekt GameLift Amazon-Servercode hinzu

Sie haben Ihre Unreal Engine-Umgebung konfiguriert und eingerichtet und können jetzt einen Spieleserver in Amazon GameLift integrieren. Der in diesem Thema vorgestellte Code führt erforderliche Aufrufe an den GameLift Amazon-Service durch. Es implementiert auch eine Reihe von Rückruffunktionen, die auf Anfragen des GameLift Amazon-Service antworten. Weitere Informationen zu den einzelnen Funktionen und zur Funktionsweise des Codes finden Sie unter Serverprozess initialisieren. Weitere Informationen zu den in diesem Code verwendeten SDK-Aktionen und Datentypen finden Sie unter. Amazon GameLift Server-SDK-Referenz für Unreal Engine

Gehen Sie wie folgt vor GameLift, um einen Spieleserver mit Amazon zu initialisieren.

> **Note**
>
> Der GameLift Amazon-spezifische Code, der im folgenden Abschnitt bereitgestellt wird, hängt von der Verwendung eines `WITH_GAMELIFT` Präprozessor-Flags ab. Dieses Kennzeichen ist nur dann wahr, wenn diese beiden Bedingungen erfüllt sind:
>
> - `Target.Type == TargetRules.TargetType.Server`
> - Die Plugins haben die Amazon GameLift Server-SDK-Binärdateien gefunden.
>
> Dadurch wird sichergestellt, dass nur Unreal Server-Builds die Backend-API GameLift von Amazon aufrufen. Außerdem können Sie damit Code schreiben, der für all die verschiedenen Unreal-Ziele, die Ihr Spiel möglicherweise erzeugt, ordnungsgemäß ausgeführt wird.

Integrieren Sie einen Spieleserver mit Amazon GameLift

1. Öffnen Sie in Visual Studio die `.sln` Datei für Ihre Anwendung. In unserem Beispiel `GameLiftUnrealApp.sln` befindet sich die Datei im Stammordner.
2. Suchen Sie bei geöffneter Lösung die *`Your-application-name`*`GameMode.h` Datei Ihrer Anwendung. Beispiel: `GameLiftUnrealAppGameMode.h`.
3. Ändern Sie die Header-Datei so, dass sie dem folgenden Beispielcode entspricht. Achten Sie darauf, "GameLiftUnrealApp" durch Ihren eigenen Anwendungsnamen zu ersetzen.

```
#pragma once

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

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();
```

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

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

    void InitGameLift();
};
```

4. Öffnen Sie die zugehörige *Your-application-name*`GameMode.cpp` Quelldatei. In unserem Beispiel:`GameLiftUnrealAppGameMode.cpp`. und ändern Sie den Code so, dass er dem folgenden Beispielcode entspricht. Achten Sie darauf, "GameLiftUnrealApp" durch Ihren eigenen Anwendungsnamen zu ersetzen.

   Dieses Beispiel zeigt, wie Sie alle erforderlichen Elemente für die Integration mit Amazon hinzufügen GameLift, wie unter Amazon GameLift zu Ihrem Spieleserver hinzufügen beschrieben. Dies umfasst:

   - Initialisierung eines Amazon GameLift API-Clients.
   - Implementierung von Rückruffunktionen zur Beantwortung von Anfragen des GameLift Amazon-Service, einschließlich `OnStartGameSession``OnProcessTerminate`, und`onHealthCheck`.
   - Rufen Sie ProcessReady () mit einem bestimmten Port an, um Amazon zu benachrichtigen, GameLiftservice wenn Sie bereit sind, Spielsitzungen zu veranstalten.

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

#include "UObject/ConstructorHelpers.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode()
{
    // set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/
ThirdPerson/Blueprints/BP_ThirdPersonCharacter"));
    if (PlayerPawnBPClass.Class != NULL)
```

```
    {
        DefaultPawnClass = PlayerPawnBPClass.Class;
    }
}

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

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

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

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

    //AuthToken returned from the "aws gamelift get-compute-auth-token" API. Note
 this will expire and require a new call to the API after 15 minutes.
    if (FParse::Value(FCommandLine::Get(), TEXT("-authtoken="),
 serverParameters.m_authToken))
    {
        UE_LOG(GameServerLog, Log, TEXT("AUTH_TOKEN: %s"),
 *serverParameters.m_authToken)
    }

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

    //The Anywhere Fleet ID.
    if (FParse::Value(FCommandLine::Get(), TEXT("-fleetid="),
 serverParameters.m_fleetId))
    {
```

```
        UE_LOG(GameServerLog, Log, TEXT("FLEET_ID: %s"),
*serverParameters.m_fleetId)
    }

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

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

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

    //Implement callback function onStartGameSession
    //GameLift sends a game session activation request to the game server
    //and passes a game session object with game properties and other settings.
    //Here is where a game server takes action based on the game session object.
    //When the game server is ready to receive incoming player connections,
    //it invokes the server SDK call ActivateGameSession().
    auto onGameSession = [=](Aws::GameLift::Server::Model::GameSession gameSession)
    {
        FString gameSessionId = FString(gameSession.GetGameSessionId());
        UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*gameSessionId);
        gameLiftSdkModule->ActivateGameSession();
    };

    m_params.OnStartGameSession.BindLambda(onGameSession);

    //Implement callback function OnProcessTerminate
    //GameLift invokes this callback before shutting down the instance hosting this
game server.
    //It gives the game server a chance to save its state, communicate with
services, etc.,
```

```
    //and initiate shut down. When the game server is ready to shut down, it
 invokes the
    //server SDK call ProcessEnding() to tell GameLift it is shutting down.
    auto onProcessTerminate = [=]()
    {
        UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
        gameLiftSdkModule->ProcessEnding();
    };

    m_params.OnTerminate.BindLambda(onProcessTerminate);

    //Implement callback function OnHealthCheck
    //GameLift invokes this callback approximately every 60 seconds.
    //A game server might want to check the health of dependencies, etc.
    //Then it returns health status true if healthy, false otherwise.
    //The game server must respond within 60 seconds, or GameLift records 'false'.
    //In this example, the game server always reports healthy.
    auto onHealthCheck = []()
    {
        UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
        return true;
    };

    m_params.OnHealthCheck.BindLambda(onHealthCheck);

    //The game server gets ready to report that it is ready to host game sessions
    //and that it will listen on port 7777 for incoming player connections.
    m_params.port = 7777;

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

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

5. Erstellen Sie ein Spielprojekt für die beiden folgenden Zieltypen: Entwicklungseditor und Entwicklungsserver.

> **Note**
>
> Sie müssen die Lösung nicht neu erstellen. Erstellen Sie stattdessen nur das Projekt unter dem `Games` Ordner, der dem Namen Ihrer App entspricht. Andernfalls erstellt Visual Studio das gesamte UE5-Projekt neu, was bis zu einer Stunde dauern kann.

6. Sobald beide Builds abgeschlossen sind, schließen Sie Visual Studio und öffnen Sie die `.uproject` Datei Ihres Projekts, um sie im Unreal Editor zu öffnen.
7. Packen Sie im Unreal Editor den Server-Build Ihres Spiels. Um ein Ziel auszuwählen, gehe zu Plattformen, Windows und wähle ***our-application-nameY-Server*** aus.
8. Um mit der Erstellung der Serveranwendung zu beginnen, wechseln Sie zu Plattformen, Windows und wählen Sie Paketprojekt aus. Wenn der Build abgeschlossen ist, sollten Sie über eine ausführbare Datei verfügen. In unserem Beispiel lautet der Dateiname`GameLiftUnrealAppServer.exe`.
9. Beim Erstellen einer Serveranwendung in Unreal Editor werden zwei ausführbare Dateien erzeugt. Eine befindet sich im Stammverzeichnis des Build-Ordners des Spiels und dient als Wrapper für die eigentliche ausführbare Serverdatei.

   Wenn Sie eine GameLift Amazon-Flotte mit Ihrem Server-Build erstellen, empfehlen wir, dass Sie die eigentliche ausführbare Serverdatei als Startpfad für die Laufzeitkonfiguration angeben. Beispielsweise könnten Sie in Ihrem Spiele-Build-Ordner eine `GameLiftFPS.exe` Datei im Stammverzeichnis und eine weitere unter haben`\GameLiftFPS\Binaries \Win64\GameLiftFPSServer.exe`. Wenn Sie eine Flotte erstellen, empfehlen wir, den Startpfad für die Laufzeitkonfiguration zu verwenden`C:\GameLiftFPS\Binaries \Win64\GameLiftFPSServer.exe`.
10. Stellen Sie sicher, dass Sie die erforderlichen UDP-Ports auf der GameLift Amazon-Flotte öffnen, damit der Spieleserver mit den Spieleclients kommunizieren kann. Standardmäßig verwendet Unreal Engine den Port`7777`. Weitere Informationen finden Sie UpdateFleetPortSettingsim Referenzhandbuch GameLift zur Amazon Service API.
11. Erstellen Sie eine `install.bat` Datei für Ihren Spiel-Build. Dieses Installationsskript wird immer dann ausgeführt, wenn der Spiel-Build auf einer GameLift Amazon-Flotte bereitgestellt wird. Hier ist eine `install.bat` Beispieldatei:

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

Für einige Versionen von Unreal Engine `install.bat` sollte das stattdessen

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

> **Note**
>
> Der Dateipfad zur `<>PrereqSetup_x64.exe` Datei lautet`Engine\Extras\Redist\en-us`.

12. Jetzt können Sie Ihren Spiel-Build verpacken und auf Amazon hochladen GameLift.

    Die Version von OpenSSL, die du mit deinem Spiel-Build packst, muss mit der Version übereinstimmen, die die Game-Engine beim Aufbau des Spielservers verwendet hat. Vergewissere dich, dass du die richtige OpenSSL-Version mit deinem Gameserver-Build packst. Für das Windows-Betriebssystem ist `.dll` das OpenSSL-Format.

    > **Note**
    >
    > Package Sie die OpenSSL-DLLs in Ihren Gameserver-Build. Achte darauf, dieselbe Version von OpenSSL zu packen, die du beim Aufbau des Spielservers verwendet hast.
    >
    > - `libssl-1_1-x64.dll`
    >
    >   `libcrypto-1_1-x64.dll`
    >
    > Package Sie Ihre Abhängigkeiten zusammen mit der ausführbaren Datei Ihres Spieleservers in das Stammverzeichnis einer Zip-Datei. Zum Beispiel sollten sich `openssl-lib` DLLs im selben Verzeichnis wie die `.exe` Datei befinden.

Nächste Schritte

Sie haben Ihre Unreal Engine-Umgebung konfiguriert und eingerichtet und können nun damit beginnen, Amazon GameLift in Ihr Spiel zu integrieren.

Weitere Informationen zum Hinzufügen von Amazon GameLift zu deinem Spiel findest du hier:

- Füge Amazon GameLift zu deinem Gameserver hinzu
- Amazon GameLift Server-SDK-Referenz für Unreal Engine

Anweisungen zum Testen Ihres Spiels finden Sie unter Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten.

Integrieren Sie Amazon GameLift in ein Unity-Projekt

In diesem Thema wird erklärt, wie Sie das Amazon GameLift C# Server SDK-Plugin für Unity einrichten und in Ihre Spielprojekte integrieren.

Zusätzliche Ressourcen:

- Downloadseite für das Amazon GameLift Server-SDK
- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity
- the section called “Unterstützung bei der Entwicklung”

Voraussetzungen

Um das Amazon GameLift C#-Server-SDK-Plugin für Unity zu verwenden, benötigen Sie die folgenden Komponenten:

- Eine Entwicklungsumgebung und eine Unity Editor-Version, die das Plugin unterstützt (siheEntwicklungsunterstützung mit Amazon GameLift). Informationen zu Unity-Versionen finden Sie in der Unity-Dokumentation unter Systemanforderungen für Unity.
- Das GameLift Amazon-Server-SDK-Plugin für das Unity-Paket. Dieses Paket enthält das Server-SDK 5+ für C#. Sie können das Paket von dieser Website herunterladen: Erste Schritte mit Amazon. GameLift
- Die vom Drittanbieter abgegrenzte Registrierung. UnityNuGet Dieses Tool verwaltet DLLs von Drittanbietern. Weitere Informationen finden Sie im UnityNuGetGithub-Repository.

Einrichten von UnityNuGet

Wenn Sie Ihr Spieleprojekt noch nicht UnityNuGet eingerichtet haben, führen Sie die folgenden Schritte aus, um das Tool mithilfe des Unity-Paketmanagers zu installieren. Alternativ können Sie die NuGet CLI verwenden, um die DLLs manuell herunterzuladen. Weitere Informationen finden Sie im Amazon GameLift C#-Server-SDK SDK for Unity`README`.

Zur Integration UnityNuGet in Ihr Spielprojekt

1. Öffnen Sie Ihr Projekt im Unity-Editor, gehen Sie zum Hauptmenü und wählen Sie Bearbeiten, Projekteinstellungen aus. Wählen Sie aus den Optionen den Abschnitt Package Manager und öffnen Sie die Gruppe Scoped Registries.
2. Wählen Sie die Schaltfläche + und geben Sie die folgenden Werte für die bereichsbezogene Registrierung ein: UnityNuGet

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

3. Für Benutzer der Unity-Version 2021:

   Suchen Sie nach der Einrichtung nach `Assembly Version Validation` Fehlern UnityNuGet, die in der Unity-Konsole angezeigt werden. Diese Fehler treten auf, wenn Bindungsumleitungen für stark benannte Assemblys in den NuGet Paketen nicht korrekt in Pfade innerhalb Ihres Unity-Projekts aufgelöst werden. Um dieses Problem zu beheben, konfigurieren Sie die Assembly-Versionsvalidierung von Unity:

   a. Gehen Sie im Unity-Editor zum Hauptmenü und wählen Sie Bearbeiten, Projekteinstellungen und öffnen Sie den Bereich Player.
   b. Deaktivieren Sie die Option Assembly-Versionsvalidierung.

Installieren Sie das Plug-in

Gehen Sie wie folgt vor, um das Amazon GameLift C#-Server-SDK-Plugin für Unity zu installieren und die Log4net-Protokollierung zu konfigurieren.

So installieren Sie das Plugin

1. Öffnen Sie Ihr Projekt im Unity-Editor, rufen Sie das Hauptmenü auf und wählen Sie Fenster, Paketmanager aus.
2. Wählen Sie die Schaltfläche +, um ein neues Paket hinzuzufügen. Wählen Sie die Option Paket aus Tarball hinzufügen.
3. Suchen Sie unter Pakete auf Festplatte auswählen nach dem Amazon GameLift C# Server SDK-Plug-In für Unity-Download-Dateien und wählen Sie die Amazon GameLift `.tgz` Server-SDK-Datei aus. Wählen Sie Öffnen, um das Plugin zu installieren.

Das GameLift Amazon-Server-SDK verwendet das Log4net-Framework zur Ausgabe von Protokollnachrichten. Es ist standardmäßig so konfiguriert, dass Nachrichten an das Terminal eines Server-Builds ausgegeben werden. Unity benötigt jedoch eine Konfiguration, um Unterstützung für die Dateiprotokollierung hinzuzufügen. Sie können diese Unterstützung zu Ihrem Projekt hinzufügen, indem Sie das bereitgestellte Beispiel in das Amazon GameLift Server SDK-Paket importieren. Gehen Sie wie folgt vor, um das Beispiel hinzuzufügen und log4net zu konfigurieren:

Um log4net für die Dateiausgabe zu konfigurieren

1. Öffnen Sie Ihr Projekt im Unity-Editor, rufen Sie das Hauptmenü auf und wählen Sie Fenster, Paketmanager aus.
2. Wählen Sie im Dropdownmenü Pakete: In Project aus und wählen Sie dann Amazon GameLift Server SDK aus der Paketliste aus. Dadurch werden die Paketdetails geöffnet.
3. Wählen Sie in den Paketdetails die Gruppenoption Samples aus und klicken Sie auf Importieren.
4. Die `log4net.config` Datei und das dazugehörige `LoggingConfiguration.cs` Skript führen automatisch die Konfiguration aus, die jetzt im `Assets/Samples` Projektordner eingerichtet ist.

> **Note**
>
> Wenn Sie Ihre `log4net.config` Datei in einen anderen Ordner im Projekt verschieben müssen, müssen Sie auch den Dateipfad der Konfigurationsdatei im Skript `LoggingConfiguration.cs` mit dem neuen Pfad aktualisieren. Weitere Informationen finden Sie im log4net-Handbuch zur Konfiguration von log4net.

Ausführlichere Anweisungen und Anleitungen zum Testen finden Sie im Download, der `README` sich im Plugin befindet.

Richten Sie eine GameLift Anywhere Amazon-Flotte zum Testen ein

Sie können Ihre Entwicklungs-Workstation als GameLift Anywhere Amazon-Hosting-Flotte einrichten, um Ihre GameLift Amazon-Integration iterativ zu testen. Mit dieser Konfiguration können Sie Spieleserverprozesse auf Ihrer Workstation starten, Spielerbeitritts- oder Matchmaking-Anfragen an Amazon senden, um Spielsitzungen GameLift zu starten, und Clients mit den neuen Spielsitzungen verbinden. Wenn Sie Ihre eigene Workstation als Hosting-Server einrichten, können Sie alle Aspekte Ihrer Spieleintegration mit Amazon GameLift überwachen.

Anweisungen zur Einrichtung Ihrer Workstation finden Sie unter Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten So führen Sie die folgenden Schritte durch:

1. Erstellen Sie einen benutzerdefinierten Speicherort für Ihre Workstation.
2. Erstellen Sie eine GameLift Anywhere Amazon-Flotte mit Ihrem neuen benutzerdefinierten Standort. Bei Erfolg gibt diese Anfrage eine Flottennummer zurück. Notieren Sie sich diesen Wert, da Sie ihn später benötigen werden.
3. Registrieren Sie Ihre Workstation als Computer in der neuen Anywhere Flotte. Geben Sie einen eindeutigen Computernamen und die IP-Adresse für Ihre Workstation an. Bei Erfolg gibt diese Anfrage einen Service SDK-Endpunkt in Form einer WebSocket URL zurück. Notieren Sie sich diesen Wert, da Sie ihn später benötigen werden.
4. Generieren Sie ein Authentifizierungstoken für Ihren Workstation-Computer. Diese kurzlebige Authentifizierung umfasst das Token und ein Ablaufdatum. Ihr Spieleserver verwendet es, um die Kommunikation mit dem GameLift Amazon-Service zu authentifizieren. Speichern Sie die Authentifizierung auf Ihrem Workstation-Computer, damit Ihre laufenden Spieleserver-Prozesse darauf zugreifen können.

Fügen Sie Ihrem Unity-Projekt GameLift Amazon-Servercode hinzu

Ihr Spieleserver kommuniziert mit dem GameLift Amazon-Service, um Anweisungen zu erhalten und den aktuellen Status zu melden. Um dies zu erreichen, fügen Sie Spieleservercode hinzu, der das GameLift Amazon-Server-SDK verwendet.

Das bereitgestellte Codebeispiel veranschaulicht die grundlegenden erforderlichen Integrationselemente. Es verwendet a`MonoBehavior`, um eine einfache Gameserver-Initialisierung mit Amazon GameLift zu veranschaulichen. Das Beispiel geht davon aus, dass der Spieleserver zum Testen auf einer GameLift Anywhere Amazon-Flotte läuft. Es beinhaltet Code für:

- Initialisieren Sie einen GameLift Amazon-API-Client. Das Beispiel verwendet die Version von `InitSDK()` mit Serverparametern für Ihre Anywhere Flotte und Rechenleistung. Verwenden Sie die WebSocket URL, die Flotten-ID, den Computernamen (Host-ID) und das Authentifizierungstoken, wie im vorherigen Thema definiertRichten Sie eine GameLift Anywhere Amazon-Flotte zum Testen ein.
- Implementieren Sie Rückruffunktionen, um auf Anfragen vom GameLift Amazon-Service zu antworten, einschließlich `OnStartGameSession``OnProcessTerminate`, und`onHealthCheck`.
- Rufen Sie ProcessReady () mit einem bestimmten Port an, um den GameLift Amazon-Service zu benachrichtigen, wenn der Prozess bereit ist, Spielsitzungen abzuhalten.

Der in diesem Thema vorgestellte Code stellt die Kommunikation mit dem GameLift Amazon-Service her und. Es implementiert auch eine Reihe von Rückruffunktionen, die auf Anfragen von antworten. Weitere Informationen zu den einzelnen Funktionen und zur Funktionsweise des Codes finden Sie unter Initialisieren des Serverprozesses. Weitere Informationen zu den in diesem Code verwendeten SDK-Aktionen und Datentypen finden Amazon GameLift Server-SDK-Referenz für C# Sie unter.

Dieses Beispiel zeigt, wie Sie alle erforderlichen Elemente hinzufügen, wie unter Amazon GameLift zu Ihrem Spieleserver hinzufügen beschrieben. Es beinhaltet:

Weitere Informationen zum Hinzufügen von GameLift Amazon-Funktionen finden Sie in den folgenden Themen:

- Füge Amazon GameLift zu deinem Gameserver hinzu
- Amazon GameLift Server-SDK-Referenz für C#

```
using System.Collections.Generic;
using Aws.GameLift.Server;
using UnityEngine;

public class ServerSDKManualTest : MonoBehaviour
{
    //This example is a simple integration that initializes a game server process
    //that is running on an Amazon GameLift Anywhere fleet.
    void Start()
    {
        //Identify port number (hard coded here for simplicity) the game server is
 listening on for player connections
        var listeningPort = 7777;

        //WebSocketUrl from RegisterHost call
        var webSocketUrl = "wss://us-west-2.api.amazongamelift.com";

        //Unique identifier for this process
        var processId = "myProcess";

        //Unique identifier for your host that this process belongs to
        var hostId = "myHost";

        //Unique identifier for your fleet that this host belongs to
        var fleetId = "myFleet";
```

```
        //Authorization token for this host process
        var authToken = "myAuthToken";

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

        //InitSDK establishes a local connection with an Amazon GameLift agent
        //to enable further communication.
        var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
        if (initSDKOutcome.Success)
        {
            //Implement callback functions
            ProcessParameters processParameters = new ProcessParameters(
            //Implement OnStartGameSession callback
                (gameSession) => {
                    //GameLift sends a game session activation request to the game
server
                    //with game session object containing game properties and other
settings.
                    //Here is where a game server takes action based on the game
session object.
                    //When the game server is ready to receive incoming player
connections,
                    //it invokes the server SDK call ActivateGameSession().
                    GameLiftServerAPI.ActivateGameSession();
                },
                (updateGameSession) => {
                    //GameLift sends a request when a game session is updated (such as
for
                    //FlexMatch backfill) with an updated game session object.
                    //The game server can examine matchmakerData and handle new
incoming players.
                    //updateReason explains the purpose of the update.
                },
                () => {
                    //Implement callback function OnProcessTerminate
                    //GameLift invokes this callback before shutting down the instance
hosting this game server.
```

```
                    //It gives the game server a chance to save its state, communicate
with services, etc.,
                    //and initiate shut down. When the game server is ready to shut
down, it invokes the
                    //server SDK call ProcessEnding() to tell GameLift it is shutting
down.
                    GameLiftServerAPI.ProcessEnding();
                },
                () => {
                    //Implement callback function OnHealthCheck
                    //GameLift invokes this callback approximately every 60 seconds.
                    //A game server might want to check the health of dependencies,
etc.
                    //Then it returns health status true if healthy, false otherwise.
                    //The game server must respond within 60 seconds, or GameLift
records 'false'.
                    //In this example, the game server always reports healthy.
                    return true;
                },
                //The game server gets ready to report that it is ready to host game
sessions
                //and that it will listen on port 7777 for incoming player connections.
                listeningPort,
                new LogParameters(new List<string>()
                {
                    //Here, the game server tells GameLift where to find game session
log files.
                    //At the end of a game session, GameLift uploads everything in the
specified
                    //location and stores it in the cloud for access later.
                    "/local/game/logs/myserver.log"
                }));

            //The game server calls ProcessReady() to tell GameLift it's ready to host
game sessions.
            var processReadyOutcome =
GameLiftServerAPI.ProcessReady(processParameters);
            if (processReadyOutcome.Success)
            {
                print("ProcessReady success.");
            }
            else
            {
```

```
                print("ProcessReady failure : " +
 processReadyOutcome.Error.ToString());
            }
        }
        else
        {
            print("InitSDK failure : " + initSDKOutcome.Error.ToString());
        }
    }

    void OnApplicationQuit()
    {
        //Make sure to call GameLiftServerAPI.ProcessEnding() and
 GameLiftServerAPI.Destroy() before terminating the server process.
        //These actions notify Amazon GameLift that the process is terminating and
 frees the API client from memory.
        GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
        GameLiftServerAPI.Destroy();
        if (processEndingOutcome.Success)
        {
            Environment.Exit(0);
        }
        else
        {
            Console.WriteLine("ProcessEnding() failed. Error: " +
 processEndingOutcome.Error.ToString());
            Environment.Exit(-1);
        }
    }
}
```

Weitere Ressourcen

Verwenden Sie die folgenden Ressourcen, um Ihren Spieleserver zu testen und die Funktionalität zu erweitern:

- Richten Sie Ihren Entwicklungscomputer als Amazon GameLift Anywhere-Flotte ein und verwenden Sie ihn für lokale Tests. Siehe Testen Sie Ihre benutzerdefinierte Serverintegration.
- Erstellen Sie Ihren Spieleserver und laden Sie den Build auf Amazon hoch GameLift. Weitere Informationen finden Sie unter Hochladen eines benutzerdefinierten Server-Builds auf Amazon GameLift.

- Stellen Sie Ihren Spieleserver-Build auf einer von Amazon GameLift verwalteten EC2-Flotte bereit. Siehe Eine neue GameLift Amazon-Flotte erstellen.

Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten

Sie können eine GameLift Anywhere Amazon-Flotte verwenden, um Ihre Spieleintegration mit Amazon GameLift iterativ aufzubauen und zu testen. Richte deine eigene Hardware als Anywhere Flotte mit einer Verbindung zum GameLift Amazon-Dienst ein und installiere und betreibe deinen Gameserver darauf. Verwenden Sie eine Test-App, um Szenarien wie das Starten/Stoppen von Spielsitzungen, das Verfolgen von Spielerverbindungen und den Umgang mit Matchmaking-Backfills durchzuführen. Mit einer Anywhere Flotte kannst du deinen Gameserver-Build nach Bedarf aktualisieren und hast vollen Überblick über die Hosting-Aktivitäten.

Sie können GameLift Anywhere Amazon-Flotten mit Spielen nutzen, die in Amazon GameLift Server SDK Version 5 oder höher integriert sind.

Themen

- Erste Entwicklung
- Iteriere auf deinem Gameserver

Erste Entwicklung

Sie haben Ihr Spiel entwickelt und integrieren es in das Amazon GameLift Server SDK. Um Ihre Integration zu testen, könnten Sie jede neue Iteration Ihres Gameserver-Builds auf Amazon hochladen GameLift und eine Flotte erstellen. Alternativ bietet Ihnen die Verwendung einer Anywhere Flotte mit Ihrem Entwicklungs-Laptop eine effizientere Möglichkeit, iteratives Entwickeln und Testen durchzuführen.

Gehen Sie wie folgt vor, um mithilfe der GameLift Amazon-Konsole oder der AWS Command Line Interface (AWS CLI) eine Spielesitzung auf Ihrem Laptop zu erstellen und eine Spielsitzung zu starten. Anywhere

Console

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich unter Hosting die Option Standorte aus.
3. Wählen Sie Standort erstellen aus.

4. Gehen Sie im Dialogfeld „Speicherort erstellen“ wie folgt vor:

   a. Geben Sie einen Standortnamen ein. Dies kennzeichnet den Standort Ihrer Rechenressourcen, die Amazon GameLift verwendet, um Ihre Spiele in Anywhere Flotten auszuführen. Benutzerdefinierte Ortsnamen müssen mit custom- beginnen.

   b. Wählen Sie Erstellen aus.

5. Gehen Sie wie folgt vor, um eine Anywhere Flotte zu erstellen:

   a. Wählen Sie im Navigationsbereich unter Hosting die Option Flotten aus.

   b. Klicken Sie auf der Seite Fleets (Flotten) auf Create fleet (Flotte erstellen).

   c. Wählen Sie im Schritt Berechnungstyp auswählen die Option Anywhere und dann Weiter aus.

   d. Definieren Sie im Schritt Flottendetails definieren Ihre neue Flotte. Weitere Informationen finden Sie unter Erstellen Sie eine neue GameLift Amazon-Flotte.

   e. Wählen Sie im Schritt Standorte auswählen den benutzerdefinierten Speicherort aus, den Sie erstellt haben.

   f. Schließe die verbleibenden Schritte zur Flottenerstellung ab, um deine Anywhere Flotte zu erstellen.

6. Registrieren Sie Ihren Laptop als Rechenressource in der Flotte, die Sie erstellt haben. Verwenden Sie den `register-compute`Befehl (oder die `RegisterCompute`API-Operation). Fügen Sie das im vorherigen Schritt `fleet-id` erstellte hinzu und fügen Sie ein `compute-name` und die Ihres Laptops hinzu`ip-address`.

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

   Beispielausgabe:

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

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

7. Starte eine Debug-Sitzung deines Gameservers.

   a. Holen Sie sich das Autorisierungstoken für Ihren Laptop in der Flotte, die Sie erstellt haben. Verwenden Sie den `get-compute-auth-token`Befehl (oder die `GetComputeAuthToken`API-Operation).

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

   Beispielausgabe:

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

   b. Führe eine Debug-Instanz der ausführbaren Datei deines Gameservers aus. Um die Debug-Instanz auszuführen, muss dein Gameserver anrufenInitSDK(). Nachdem der Prozess bereit ist, eine Spielsitzung zu veranstalten, ruft der Gameserver anProcessReady().

8. Erstelle eine Spielesitzung, um deine erste Integration mit Amazon zu testen GameLiftAnywhere. Verwenden Sie den `create-game-session`Befehl (oder die `CreateGameSession`API-Operation). Geben Sie den benutzerdefinierten Standort der Flotte an.

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

   Beispielausgabe:

   ```
   GameSession {
   ```

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

Amazon GameLift sendet eine `onStartGameSession()` Nachricht an Ihren registrierten Serverprozess. Die Nachricht enthält das `GameSession` Objekt aus dem vorherigen Schritt mit Spieleigenschaften, Spielsitzungsdaten, Matchmaker-Daten und mehr über die Spielsitzung.

9. Füge deinem Gameserver Logik hinzu, sodass dein Serverprozess auf die `onStartGameSession()` Nachricht mit reagiert`ActivateGameSession()`. Der Vorgang sendet eine Bestätigung an AmazonGameLift, dass Ihr Server die Nachricht „Spielsitzung erstellen“ empfangen und akzeptiert hat. Weitere Informationen finden Sie unter Referenz zum Amazon GameLift Server-SDK.

Ihr Gameserver führt jetzt eine Spielsitzung durch, die Sie testen und für die Iteration verwenden können. Fahren Sie mit dem nächsten Abschnitt fort, um zu erfahren, wie Sie auf Ihrem Spielserver iterieren können.

AWS CLI

1. Erstellen Sie mit dem `create-location`Befehl (oder der `CreateLocation`API-Operation) einen benutzerdefinierten Speicherort. Ein benutzerdefinierter Standort kennzeichnet den Standort Ihrer Hardware, die Amazon GameLift verwendet, um Ihre Spiele in Anywhere Flotten auszuführen.

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

Beispielausgabe:

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

2. Erstellen Sie mithilfe des `create-fleet`Befehls (oder der `CreateFleet`API-Operation) eine Anywhere Flotte mit Ihrem benutzerdefinierten Standort. Amazon GameLift erstellt die Flotte in Ihrer Heimatregion und die von Ihnen angegebenen benutzerdefinierten Standorte.

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

Beispielausgabe:

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

3. Registrieren Sie Ihren Laptop als Rechenressource in der Flotte, die Sie erstellt haben. Verwenden Sie den `register-compute`Befehl (oder die `RegisterCompute`API-Operation). Fügen Sie das im vorherigen Schritt `fleet-id` erstellte hinzu und fügen Sie ein `compute-name` und Ihr Laptop ist öffentlich`ip-address`.

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

Beispielausgabe:

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

4. Starte eine Debug-Sitzung deines Gameservers.

   a. Holen Sie sich das Autorisierungstoken für Ihren Laptop in der Flotte, die Sie erstellt haben. Verwenden Sie den `get-compute-auth-token`Befehl (oder die `GetComputeAuthToken`API-Operation).

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

   Beispielausgabe:

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

   b. Führe eine Debug-Instanz der ausführbaren Datei deines Gameservers aus. Um die Debug-Instanz auszuführen, muss dein Gameserver anrufen`InitSDK()`. Nachdem der Prozess bereit ist, eine Spielsitzung zu veranstalten, ruft der Gameserver an`ProcessReady()`.

5. Erstelle eine Spielesitzung, um deine erste Integration mit Amazon zu testen GameLiftAnywhere. Verwenden Sie den `create-game-session`Befehl (oder die `CreateGameSession`API-Operation).

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

   Beispielausgabe:

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

```
    ...
}
```

Amazon GameLift sendet eine `onStartGameSession()` Nachricht an Ihren registrierten Serverprozess. Die Nachricht enthält das `GameSession` Objekt aus dem vorherigen Schritt mit Spieleigenschaften, Spielsitzungsdaten, Matchmaker-Daten und mehr über die Spielsitzung.

6. Füge deinem Gameserver Logik hinzu, sodass dein Serverprozess auf die `onStartGameSession()` Nachricht mit reagiert`ActivateGameSession()`. Der Vorgang sendet eine Bestätigung an AmazonGameLift, dass Ihr Server die Nachricht „Spielsitzung erstellen“ empfangen und akzeptiert hat. Weitere Informationen finden Sie unter Referenz zum Amazon GameLift Server-SDK.

Ihr Gameserver führt jetzt eine Spielsitzung durch, die Sie testen und für die Iteration verwenden können. Fahren Sie mit dem nächsten Abschnitt fort, um zu erfahren, wie Sie auf Ihrem Spielserver iterieren können.

Iteriere auf deinem Gameserver

Stellen Sie sich in diesem Anwendungsfall ein Szenario vor, in dem Sie Ihren Spielserver eingerichtet und getestet haben und einen Fehler gefunden haben. Mit Amazon GameLift Anywhere können Sie Ihren Code iterieren und den aufwändigen Aufbau einer Amazon EC2-Flotte vermeiden.

1. Reinigen Sie Ihre vorhandenen`GameSession`, wenn möglich. Wenn der Spieleserver abstürzt oder nicht anruft`ProcessEnding()`, GameLift räumt Amazon das auf, `GameSession` nachdem der Spieleserver aufgehört hat, Gesundheitschecks zu senden.
2. Nehmen Sie die Codeänderungen auf Ihrem Gameserver vor, kompilieren Sie und bereiten Sie sich auf den nächsten Test vor.
3. Ihre vorherige Anywhere Flotte ist immer noch aktiv und Ihr Laptop ist immer noch als Rechenressource in der Flotte registriert. Um erneut mit dem Testen zu beginnen, erstellen Sie eine neue Debug-Instanz.

    a. Rufen Sie das Autorisierungstoken für Ihren Laptop in der Flotte ab, die Sie erstellt haben. Verwenden Sie den `get-compute-auth-token`Befehl (oder die `GetComputeAuthToken`API-Operation).

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

Beispielausgabe:

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

b. Führe eine Debug-Instanz der ausführbaren Datei deines Gameservers aus. Um die Debug-Instanz auszuführen, muss dein Gameserver anrufen`InitSDK()`. Nachdem der Prozess bereit ist, eine Spielsitzung zu veranstalten, ruft der Gameserver an`ProcessReady()`.

4. Ihre Flotte verfügt jetzt über einen verfügbaren Serverprozess. Erstelle deine Spielsitzung und führe deine nächsten Tests durch. Verwenden Sie den `create-game-session`Befehl (oder die `CreateGameSession`API-Operation).

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

Amazon GameLift sendet eine on`StartGameSession()` Nachricht an Ihren registrierten Serverprozess. Die Nachricht enthält das `GameSession` Objekt aus dem vorherigen Schritt mit Spieleigenschaften, Spielsitzungsdaten, Matchmaker-Daten und mehr über die Spielsitzung.

5. Füge deinem Gameserver Logik hinzu, sodass dein Serverprozess auf die on`StartGameSession()` Nachricht mit reagiert`ActivateGameSession()`. Der Vorgang sendet eine Bestätigung an AmazonGameLift, dass Ihr Server die Nachricht „Spielsitzung erstellen“ empfangen und akzeptiert hat. Weitere Informationen finden Sie unter Referenz zum Amazon GameLift Server-SDK.

Nachdem Sie den Test Ihres Gameservers abgeschlossen haben, können Sie Amazon weiterhin GameLift für die Verwaltung Ihrer Flotte und Ihres Gameservers verwenden. Weitere Informationen finden Sie unter Erstellen Sie eine GameLift Anywhere Amazon-Flotte.

Testen Sie Ihre Integration mit Amazon GameLift Local

> **Note**
>
> Verwenden Sie dieses Testverfahren, wenn Sie eine Version des Amazon GameLift Server SDK verwenden, nämlich Version 4.x oder früher. Ihr Server-SDK-Paket enthält eine kompatible Version von Amazon GameLift Local. Wenn Sie Server-SDK Version 5.x verwenden, finden Sie unter Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten lokale Tests mit einer GameLift Anywhere Amazon-Flotte.

Verwenden Sie Amazon GameLift Local, um eine eingeschränkte Version des verwalteten GameLift Amazon-Dienstes auf einem lokalen Gerät auszuführen und Ihre Spieleintegration damit zu testen. Dieses Tool ist nützlich, wenn Sie die Integration Ihres Spiels als iterative Entwicklung durchführen. Die Alternative — jeden neuen Build auf Amazon hochzuladen GameLift und eine Flotte zu konfigurieren, um Ihr Spiel zu hosten — kann jedes Mal 30 Minuten oder länger dauern.

Mit Amazon GameLift Local können Sie Folgendes überprüfen:

- Ihr Gameserver ist korrekt in das Server-SDK integriert und kommuniziert ordnungsgemäß mit dem GameLift Amazon-Dienst, um neue Spielesitzungen zu starten, neue Spieler aufzunehmen und Gesundheit und Status zu melden.
- Ihr Spielclient ist korrekt in das AWS SDK für Amazon integriert GameLift und kann Informationen über bestehende Spielsitzungen abrufen, neue Spielsitzungen starten, Spieler zu Spielen zusammenführen und eine Verbindung zur Spielsitzung herstellen.

Amazon GameLift Local ist ein Befehlszeilentool, das eine eigenständige Version des verwalteten Amazon-Dienstes startet. GameLift Amazon GameLift Local stellt außerdem ein laufendes Ereignisprotokoll mit der Initialisierung von Serverprozessen, Zustandsprüfungen sowie API-Aufrufen und -Antworten bereit. Amazon GameLift Local erkennt eine Teilmenge der AWS SDK-Aktionen für AmazonGameLift. Sie können Aufrufe über die AWS CLI oder von Ihrem Spiele-Client aus ausführen. Alle API-Aktionen werden lokal genauso ausgeführt wie im GameLift Amazon-Webservice.

Jeder Serverprozess sollte nur eine einzige Spielsitzung hosten. Die Spielsitzung ist die ausführbare Datei, mit der Sie eine Verbindung zu Amazon GameLift Local herstellen. Wenn die Spielsitzung abgeschlossen ist, sollten Sie anrufen `GameLiftServerSDK::ProcessEndning` und den Vorgang beenden. Wenn Sie lokal mit Amazon GameLift Local testen, können Sie mehrere Serverprozesse

starten. Jeder Prozess stellt eine Verbindung zu Amazon GameLift Local her. Sie können dann für jeden Serverprozess eine Spielsitzung erstellen. Wenn deine Spielsitzung endet, sollte dein Gameserverprozess beendet werden. Sie müssen dann manuell einen anderen Serverprozess starten.

Amazon GameLift Local unterstützt die folgenden APIs:

- CreateGameSession
- CreatePlayerSession
- CreatePlayerSessions
- DescribeGameSessions
- DescribePlayerSessions

Amazon GameLift lokal einrichten

Amazon GameLift Local wird als ausführbare `.jar` Datei im Paket mit dem Server-SDK bereitgestellt. Es kann unter Windows oder Linux ausgeführt und mit jeder von Amazon GameLift unterstützten Sprache verwendet werden.

Bevor Sie Local ausführen, müssen Sie außerdem Folgendes installieren.

- Ein Build des Amazon GameLift Server SDK Version 3.1.5 bis 4.x.
- Java 8

Teste einen Gameserver

Wenn du nur deinen Gameserver testen möchtest, kannst du den verwenden, AWS CLI um Game-Client-Aufrufe an den Amazon GameLift Local-Service zu simulieren. Dies reicht aus, um zu überprüfen, ob Ihr Spiel-Server bezüglich der folgenden Punkte entsprechend Ihren Erwartungen funktioniert:

- Der Spieleserver wird ordnungsgemäß gestartet und das Amazon GameLift Server SDK initialisiert.
- Im Rahmen des Startvorgangs teilt der Spieleserver Amazon mit, GameLift dass der Server bereit ist, Spielesitzungen zu hosten.
- Der Gameserver sendet während des Betriebs GameLift jede Minute den Gesundheitsstatus an Amazon.

- Die Spiel-Server reagiert auf Anforderungen, eine neue Spielsitzung zu starten.

1. Starten Sie Amazon GameLift Local.

   Öffnen Sie ein Befehlszeilenfenster, navigieren Sie zu dem Verzeichnis mit der Datei *GameLiftLocal.jar* und führen Sie sie aus. Standardmäßig horcht Local auf Port 8080 auf Anforderungen von Spielclients. Wenn Sie eine andere Portnummer angeben möchten, verwenden Sie den Parameter `-p`. Beispiel:

   ```
   java -jar GameLiftLocal.jar -p 9080
   ```

   Wenn Local startet, sehen Sie Protokolle, die angeben, dass zwei lokale Server gestartet wurden, einer, der auf Ihren Spiel-Server horcht und ein zweiter, der auf Aufrufe seitens des Spielclients oder über die AWS CLI wartet. Die beiden Protokolle berichten fortlaufend weiter über die Aktivitäten auf den beiden lokalen Servern, einschließlich der Kommunikation mit Ihren Spielkomponenten.
2. Starten Sie Ihren Spiel-Server.

   Starte deinen in Amazon GameLift integrierten Gameserver lokal. Sie brauchen dabei nicht den Endpunkts für den Spiel-Server zu ändern.

   Im lokalen Befehlszeilenfenster geben Protokollmeldungen an, dass Ihr Spieleserver eine Verbindung zum Amazon GameLift Local-Service hergestellt hat. Das bedeutet, dass Ihr Gameserver das Amazon GameLift Server SDK (mit`InitSDK()`) erfolgreich initialisiert hat. Das SDK hat `ProcessReady()` mit den angezeigten Protokollpfaden aufgerufen und ist nun im Erfolgsfall bereit, eine Spielsitzung zu hosten. Während der Spieleserver läuft, GameLift protokolliert Amazon jeden Gesundheitsstatusbericht vom Spieleserver. Das folgende Beispiel zu Protokollierungsmeldungen zeigt einen erfolgreich integrierten Spiel-Server:

   ```
   16:50:53,217  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK
    connected: /127.0.0.1:64247
   16:50:53,217  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK pid is 17040,
    sdkVersion is 3.1.5 and sdkLanguage is CSharp
   16:50:53,217  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - NOTE: Only SDK
    versions 3.1.5 and above are supported in GameLiftLocal!
   16:50:53,451  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
    received from: /127.0.0.1:64247 and ackRequest requested? true
   16:50:53,543  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
    data: logPathsToUpload: "C:\\game\\logs"
   ```

```
logPathsToUpload: "C:\\game\\error"
port: 1935

16:50:53,544  INFO || - [HostProcessManager] nioEventLoopGroup-3-1 - Registered new
 process true, true,
16:50:53,558  INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onReportHealth
 received from /127.0.0.1:64247 with health status: healthy
```

Es können u. a. die Fehler- und Warnmeldungen angezeigt werden:

- Fehler: "ProcessReadyhat keinen Prozess mit pID:<process ID>! gefunden Wurde initSDK () aufgerufen?“
- Warnung: „Für einen Prozess mit pID:<process ID>! existiert bereits ein Prozessstatus Wird ProcessReady (...) mehr als einmal aufgerufen?“

3. Starten Sie die AWS CLI.

   Sobald Ihr Spiel-Server erfolgreich `ProcessReady()` ruft, können Sie die Clientaufrufe absetzen. Öffnen Sie ein Befehlszeilenfenster und starten Sie das AWS CLI-Tool. Der verwendet AWS CLI standardmäßig den Amazon GameLift Web Service-Endpunkt. Sie müssen diesen Endpunkt in allen Anfragen mit dem Parameter `--endpoint-url` mit dem lokalen Endpunkt überschreiben, wie im folgenden Beispiel dargestellt.

   ```
   AWS gamelift describe-game-sessions --endpoint-url http://localhost:9080  --fleet-
   id fleet-123
   ```

   Im AWS CLI Befehlszeilenfenster führen `AWS gamelift` Befehle zu Antworten, wie in der AWS CLIBefehlsreferenz dokumentiert.

4. Erstellen Sie eine Spielsitzung.

   Reichen Sie mit dem AWS CLI eine CreateGameSession() Anfrage ein. Die Anforderung sollte der erwarteten Syntax entsprechen. Bei Local kann der `FleetId`-Parameter auf eine beliebige Zeichenfolge eingestellt sein (`^fleet-\S+`).

   ```
   AWS gamelift create-game-session --endpoint-url http://localhost:9080 --maximum-
   player-session-count 2 --fleet-id
       fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
   ```

Im lokalen Befehlszeilenfenster geben Protokollmeldungen an, dass Amazon GameLift Local Ihrem Gameserver einen onStartGameSession Rückruf gesendet hat. Wenn eine Spielsitzung erfolgreich erstellt wurde, antwortet Ihr Spiel-Server durch einen Aufruf von ActivateGameSession.

```
13:57:36,129  INFO || - [SDKInvokerImpl]
        Thread-2 - Finished sending event to game server to start a game session:
        arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6.
        Waiting for ack response.13:57:36,143  INFO || - [SDKInvokerImpl]
        Thread-2 - Received ack response: true13:57:36,144  INFO || -
        [CreateGameSessionDispatcher] Thread-2 - GameSession with id:
        arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6
        created13:57:36,227  INFO || - [SDKListenerImpl]
        nioEventLoopGroup-3-1 - onGameSessionActivate received
 from: /127.0.0.1:60020 and ackRequest
        requested? true13:57:36,230  INFO || - [SDKListenerImpl]
        nioEventLoopGroup-3-1 - onGameSessionActivate data: gameSessionId:
        "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"
```

In dem AWS CLI Fenster GameLift antwortet Amazon mit einem Spielsitzungsobjekt, das eine Spielsitzungs-ID enthält. Beachten Sie, dass der Status der neuen Spielsitzung „Activating“ lautet. Der Status ändert sich zu Aktiv, sobald dein Gameserver aufgerufen wird. ActivateGameSession Wenn Sie sich bezüglich der Statusänderung vergewissern möchten, starten Sie über die AWS CLI den Aufruf DescribeGameSessions().

```
{
    "GameSession": {
      "Status": "ACTIVATING",
      "MaximumPlayerSessionCount": 2,
      "FleetId": "fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d",
      "GameSessionId": "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890",
      "IpAddress": "127.0.0.1",
```

```
      "Port": 1935
    }
}
```

Teste einen Gameserver und einen Client

Im die gesamte Integration Ihres Spiels zu überprüfen (inklusive der Verbindung von Spielern mit Spielen), können Sie gleichzeitig Ihren Spiel-Server und Ihren Spieleclient lokal ausführen. Auf diese Weise können Sie programmatische Aufrufe von Ihrem Spieleclient an Amazon GameLift Local testen. Sie können die folgenden Aktionen überprüfen:

- Der Spielclient stellt erfolgreich AWS SDK-Anfragen an den Amazon GameLift Local Service, unter anderem, um Spielsitzungen zu erstellen, Informationen über bestehende Spielsitzungen abzurufen und Spielersitzungen zu erstellen.
- Die Spiel-Server validiert die Spieler ordnungsgemäß, wenn diese versuchen, einem Spiel beizutreten. Bei validierte Spielern kann der Spiel-Server Daten über den Spieler abrufen (falls implementiert).
- Die Spiel-Server meldet den Verbindungsverlust, wenn ein Spieler das Spiel verlässt.
- Die Spiel-Server meldet das Ende einer Spielsitzung.

1. Starten Sie Amazon GameLift Local.

   Öffnen Sie ein Befehlszeilenfenster, navigieren Sie zu dem Verzeichnis mit der Datei *GameLiftLocal.jar* und führen Sie sie aus. Standardmäßig horcht Local auf Port 8080 auf Anforderungen von Spielclients. Wenn Sie eine andere Portnummer angeben möchten, verwenden Sie den Parameter `-p`. Beispiel:

   ```
   ./gamelift-local -p 9080
   ```

   Wenn Local startet, sehen Sie Protokolle, die angeben, dass zwei lokale Server gestartet wurden, einer, der auf Ihren Spiel-Server horcht und ein zweiter, der auf Aufrufe seitens des Spielclients oder über die AWS CLI wartet.

2. Starten Sie Ihren Spiel-Server.

   Starte deinen in Amazon GameLift integrierten Gameserver lokal. Weitere Informationen zu Meldungsprotokollen finden Sie unter Teste einen Gameserver.

3. Konfigurieren Sie Ihren Spieleclient für Local und starten Sie ihn.

   Um deinen Spielclient mit dem Amazon GameLift Local Service zu verwenden, musst du die folgenden Änderungen an der Einrichtung deines Spielclients vornehmen, wie unter beschriebenAmazon GameLift auf einem Backend-Service einrichten:

   - Ändern Sie das `ClientConfiguration`-Objekt so, dass es auf Ihren lokalen Endpunkt verweist, z. B. `http://localhost:9080`.
   - Legen Sie einen Wert für die Zielflotten-ID fest. Bei Local brauchen Sie keine echte Flotten-ID anzugeben, es reicht aus, eine beliebige gültige Zeichenfolge (`^fleet-\S+`) anzugeben, z. B. `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`.
   - Legen Sie die AWS-Anmeldeinformationen fest. Für Lokale benötigen Sie keine echten AWS-Anmeldeinformationen. Sie können den Zugriffsschlüssel und den geheimen Schlüssels auf eine beliebige Zeichenfolge festlegen.

   Sobald du den Spielclient startest, sollten im lokalen Befehlszeilenfenster Logmeldungen darauf hinweisen, dass der Spielclient initialisiert wurde `GameLiftClient` und erfolgreich mit dem GameLift Amazon-Dienst kommuniziert wurde.

4. Test-Game-Client-Aufrufe an den GameLift Amazon-Dienst.

   Überprüfen Sie, ob Ihr Spieleclient erfolgreich einige oder alle der folgenden API-Aufrufe ausführen kann:

   - CreateGameSession()
   - DescribeGameSessions()
   - CreatePlayerSession()
   - CreatePlayerSessions()
   - DescribePlayerSessions()

   In dem Local-Befehlszeilenfenster werden nur Aufrufe an `CreateGameSession()` mit entsprechenden Meldungen protokolliert. Logmeldungen zeigen, wenn Amazon GameLift Local Ihren Gameserver auffordert, eine Spielesitzung zu starten (`onStartGameSession`Callback), und `ActivateGameSession` wenn Ihr Gameserver sie aufruft, eine erfolgreiche Meldung erhält. In dem AWS CLI-Fenster führen alle API-Aufrufe entweder zu Antworten oder zu Fehlermeldungen, wie dokumentiert.

5. Überprüfen Sie, ob Ihr Spiel-Server die Verbindungen neuer Spieler validiert.

   Stellen Sie nach dem Erstellen einer Spielsitzung und einer Spielersitzung eine direkte Verbindung zu der Spielsitzung her.

   In dem Befehlszeilenfenster von Local sollten Meldungen darauf hinweisen, dass der Spiel-Server eine `AcceptPlayerSession()`-Anfrage gesendet hat, um die neue Spielerverbindung zu überprüfen. Wenn Sie über die AWS CLI `DescribePlayerSessions()` aufrufen, sollte sich der Status der Spielersitzung von „Reserved“ in „Active“ ändern.

6. Stellen Sie sicher, dass Ihr Gameserver den Spiel- und Spielerstatus an den GameLift Amazon-Dienst meldet.

   Damit Amazon GameLift die Spielernachfrage verwalten und Messwerte korrekt melden kann, muss Ihr Gameserver verschiedene Statusmeldungen an Amazon GameLift zurückmelden. Überprüfen Sie, ob Local Ereignisse zu den folgenden Aktionen protokolliert. Sie können die Statusänderungen auch über die AWS CLI nachverfolgen.

   - Der Spieler trennt die Verbindung zu einer Spielsitzung — Die Logmeldungen von Amazon GameLift Local sollten zeigen, dass Ihr Gameserver anruft`RemovePlayerSession()`. Ein AWS CLI-Aufruf an `DescribePlayerSessions()` sollte eine Statusänderung von `Active` nach `Completed` sichtbar machen. Sie können auch `DescribeGameSessions()` aufrufen, um zu überprüfen, ob sich die Anzahl der Spieler in der aktuellen Spielsitzung um 1 vermindert.
   - Die Spielsitzung endet — Die Logmeldungen von Amazon GameLift Local sollten zeigen, dass Ihr Gameserver anruft`TerminateGameSession()`.

     > **Note**
     >
     > Frühere Anweisungen lauteten, anzurufen`TerminateGameSession()`, wenn eine Spielsitzung beendet wurde. Diese Methode ist in Amazon GameLift Server SDK v4.0.1 veraltet. Siehe Beenden Sie eine Spielsitzung.

   - Der Serverprozess ist beendet — Die Logmeldungen von Amazon GameLift Local sollten zeigen, dass Ihr Gameserver anruft`ProcessEnding()`. Ein AWS CLI-Aufruf an `DescribeGameSessions()` sollte eine Statusänderung von `Active` nach `Terminated` (bzw. `Terminating`) sichtbar machen.

Variationen mit lokalen

Beachten Sie bei der Verwendung von Amazon GameLift Local Folgendes:

- Im Gegensatz zum GameLift Amazon-Webservice verfolgt Local nicht den Systemstatus eines Servers und initiiert den `onProcessTerminate` Rückruf. Local beendet lediglich die Protokollierung für den Zustand des Spiel-Servers.
- Bei Aufrufe des AWS-SDK werden Flotten-IDs nicht validiert. Sie können also eine beliebigen Zeichenfolge sein, die den Parameteranforderungen entspricht (`^fleet-\S+`).
- Mit Local erstellte Spielsitzungs-ID haben eine abweichende Struktur. Sie enthalten die Zeichenfolge `local`, wie hier dargestellt:

```
arn:aws:gamelift:local::gamesession/fleet-123/gsess-56961f8e-
db9c-4173-97e7-270b82f0daa6
```

Integration von Spielen mit Amazon GameLift Realtime Servern

Dieses Thema bietet einen Überblick über die Lösung „Managed Amazon GameLift with Realtime Servers“. In der Übersicht wird erklärt, wann diese Lösung für Ihr Spiel geeignet ist und wie Realtime Servers Multiplayer-Spiele unterstützen.

Eine vollständige Roadmap, um dein Spiel zum Laufen zu bringen, findest du unterRoadmap für GameLift verwaltetes Hosting von Amazon.

> **Tip**
>
> Informationen zum Ausprobieren des GameLift Amazon-Gameserver-Hostings finden Sie unterErste Schritte mit Amazon GameLift.

Was sind Echtzeitserver?

Echtzeitserver sind leichte ready-to-go Spieleserver, die Amazon Ihnen zur Verwendung mit Ihren Multiplayer-Spielen zur GameLift Verfügung stellt. Echtzeitserver machen den Entwicklungs-, Test- und Bereitstellungsprozess eines benutzerdefinierten Gameservers überflüssig. Diese Lösung kann dazu beitragen, den Zeit- und Arbeitsaufwand für die Fertigstellung Ihres Spiels zu minimieren.

Schlüsselfunktionen

- Vollständiger Netzwerk-Stack für die Interaktion zwischen Spielclient und Server
- Kernfunktionalität des Gameservers
- Anpassbare Serverlogik
- Live-Updates für Echtzeitkonfigurationen und Serverlogik
- FlexMatchMatchmaking
- Flexible Steuerung der Hosting-Ressourcen

Richten Sie Echtzeitserver ein, indem Sie eine Flotte erstellen und ein Konfigurationsskript bereitstellen. Weitere Informationen zur Erstellung von Echtzeitservern und zur Vorbereitung Ihres Spielclients finden Sie unterBereite deinen Realtime-Server vor.

So verwaltet Realtime Servers Spielsitzungen

Du kannst benutzerdefinierte Logik für die Verwaltung von Spielsitzungen hinzufügen, indem du sie in das Realtime-Skript integrierst. Sie können Code schreiben, um auf serverspezifische Objekte zuzugreifen, ereignisgesteuerte Logik mithilfe von Callbacks hinzufügen oder Logik hinzufügen, die auf ereignislosen Szenarien basiert.

Wie Echtzeit-Clients und -Server interagieren

Während einer Spielsitzung interagieren die Spielclients, indem sie über einen Backend-Dienst Nachrichten an den Realtime-Server senden. Der Backend-Dienst leitet die Nachrichten dann an die Spielclients weiter, um Aktivitäten, Spielstatus und relevante Spieldaten auszutauschen.

Darüber hinaus kannst du anpassen, wie Clients und Server interagieren, indem du dem Echtzeit-Skript Spiellogik hinzufügst. Mit benutzerdefinierter Spiellogik könnte ein Echtzeitserver Callbacks implementieren, um ereignisgesteuerte Antworten zu starten.

Kommunikationsprotokoll

Echtzeitserver und verbundene Spielclients kommunizieren über zwei Kanäle: eine TCP-Verbindung für eine zuverlässige Übertragung und einen UDP-Kanal für eine schnelle Lieferung. Beim Erstellen von Nachrichten wählen Spiele-Clients das zu verwendende Protokoll abhängig von der Art der Nachrichten. Die Nachrichtenzustellung ist standardmäßig auf UDP festgelegt. Wenn kein UDP-Kanal verfügbar ist, GameLift sendet Amazon Nachrichten über TCP als Fallback.

Nachrichteninhalt

Der Nachrichteninhalt besteht aus zwei Elementen: einen erforderlichen Operationscode (opCode) und eine optionale Nutzlast. Der OpCode einer Nachricht identifiziert eine bestimmte Spieleraktivität oder ein bestimmtes Spielereignis, und die Payload liefert zusätzliche Daten zum Operationscode. Beide Elemente werden vom Entwickler definiert. Ihr Spielclient handelt auf der Grundlage der OpCodes in den Nachrichten, die er empfängt.

Spielergruppen

Realtime Servers bietet Funktionen zur Verwaltung von Spielergruppen. Standardmäßig GameLift ordnet Amazon alle Spieler, die eine Verbindung zu einem Spiel herstellen, einer Gruppe „Alle Spieler“ zu. Darüber hinaus können Entwickler zusätzliche Gruppen für ihre Spiele einrichten und Spieler können gleichzeitig Mitglieder in mehreren Gruppen sein. Gruppenmitglieder können Nachrichten senden und Spieldaten mit allen Spielern in der Gruppe teilen. Eine mögliche Verwendung für Gruppen ist die Einrichtung von Spielerteams und die Verwaltung der Teamkommunikation.

Echtzeitserver mit TLS-Zertifikaten

Bei Echtzeitservern sind Serverauthentifizierung und Datenpaketverschlüsselung in den Dienst integriert. Diese Sicherheitsfunktionen werden aktiviert, wenn Sie die Generierung von TLS-Zertifikaten aktivieren. Wenn ein Spielclient versucht, eine Verbindung zu einem Realtime-Server herzustellen, antwortet der Server automatisch mit dem TLS-Zertifikat, das der Client validiert. Amazon GameLift verarbeitet die Verschlüsselung mithilfe von TLS für die TCP (WebSockets) - Kommunikation und DTLS für den UDP-Verkehr.

Anpassen eines Realtime-Servers

Ein Echtzeitserver fungiert als statusloser Relay-Server. Der Echtzeitserver leitet Pakete mit Nachrichten und Spieldaten zwischen den mit dem Spiel verbundenen Spielclients weiter. Der Realtime-Server wertet jedoch keine Nachrichten aus, verarbeitet keine Daten und führt keine Spiellogik durch. Auf diese Weise behält jeder Spielclient seinen eigenen Überblick über den Spielstatus und informiert andere Spieler über den Relay-Server über Updates. Jeder Spiele-Client ist dafür verantwortlich, diese Updates zu übernehmen und seinen eigenen Spielstatus abzugleichen.

Sie können Ihre Server anpassen, indem Sie die Echtzeit-Skriptfunktion erweitern. Mithilfe der Spiellogik können Sie zum Beispiel ein Spiel erstellen, das den Status des Spiels kontrolliert und auf dem Server verlässlich ist.

Amazon GameLift definiert eine Reihe von serverseitigen Callbacks für Echtzeit-Skripts. Implementieren Sie diese Callbacks, um Ihrem Server ereignisgesteuerte Funktionen hinzuzufügen. Beispielsweise ist Folgendes möglich:

- Einen Spieler zu authentifizieren, wenn ein Spiele-Client versucht, eine Verbindung mit dem Server herzustellen
- Überprüfe auf Anfrage, ob ein Spieler einer Gruppe beitreten kann.
- Bestimme, wann Nachrichten von einem bestimmten Spieler oder an einen Zielspieler übermittelt werden sollen, oder führe als Reaktion eine zusätzliche Verarbeitung durch.
- Benachrichtige alle Spieler, wenn ein Spieler eine Gruppe verlässt oder die Verbindung zum Server trennt.
- Sehen Sie sich den Inhalt von Spielsitzungsobjekten oder Nachrichtenobjekten an und verwenden Sie die Daten.

Bereitstellung und Aktualisierung von Echtzeitservern

Ein wesentlicher Vorteil von Realtime Servern ist die Möglichkeit, Ihre Skripte jederzeit zu aktualisieren. Wenn Sie ein Skript aktualisieren, GameLift verteilt Amazon die neue Version innerhalb von Minuten an alle Hosting-Ressourcen. Nachdem Amazon das neue Skript GameLift bereitgestellt hat, verwenden alle neuen Spielesitzungen, die danach erstellt werden, die neue Skriptversion. (Bestehende Spielsitzungen verwenden weiterhin die Originalversion.)

Fangen Sie an, Ihr Spiel mit Echtzeitservern zu integrieren:

- Integration eines Game-Clients für Realtime Server
- Erstellen eines Echtzeit-Skripts

Integration eines Game-Clients für Realtime Server

In diesem Thema wird beschrieben, wie Sie Ihren Spielclient darauf vorbereiten, an von Amazon GameLift gehosteten Spielesitzungen teilnehmen und daran teilnehmen zu können.

Es gibt zwei Gruppen von Aufgaben, die zur Vorbereitung Ihres Spiele-Clients erforderlich sind:

- Einrichten Ihres Spiele-Clients zum Erfassen von Informationen zu bestehenden Spielen, zur Anforderung von Zuordnungen, zum Starten neuer Spielesitzungen und zur Reservierung von Spielesitzungsplätzen für einen Spieler.

- Ermöglichen Sie Ihrem Spielclient, an einer Spielsitzung teilzunehmen, die auf einem Realtime-Server gehostet wird, und Nachrichten auszutauschen.

Finde oder erstelle Spielsitzungen und Spielersitzungen

Richten Sie Ihren Spiel-Client so ein, dass er Spielesitzungen suchen oder starten, FlexMatch-Zuordnungen anfragen und Platz für Spieler in einem Spiel durch die Erstellung von Spielersitzungen reservieren kann. Es hat sich bewährt, einen Backend-Service zu erstellen und ihn zu verwenden, um die direkten Anfragen an den GameLift Amazon-Dienst zu stellen, wenn er durch eine Spiele-Client-Aktion ausgelöst wird. Der Backend-Service leitet dann die relevanten Antworten an den Spielclient zurück.

1. Füge das AWS SDK zu deinem Spielclient hinzu, initialisiere einen GameLift Amazon-Client und konfiguriere ihn so, dass er die Hosting-Ressourcen in deinen Flotten und Warteschlangen nutzt. Das AWS SDK ist in mehreren Sprachen verfügbar. Weitere Informationen finden Sie in den Amazon GameLift SDKsFür kundenspezifische Kundenservices.
2. Fügen Sie Ihrem Backend-Service GameLift Funktionen hinzu. Eine detailliertere Anleitung findest du unter Füge Amazon GameLift zu deinem Spielclient hinzu und FlexMatchMatchmaking hinzufügen. Die beste Vorgehensweise ist es, die Platzierung von Spielsitzungen zu nutzen, um neue Spielsitzungen zu erstellen. Mit dieser Methode kannst du alle Vorteile der GameLift Fähigkeit nutzen, schnell und intelligent neue Spielsitzungen zu platzieren und die Latenzdaten der Spieler zu verwenden, um Spielverzögerungen zu minimieren. Ihr Backend-Service muss mindestens in der Lage sein, neue Spielsitzungen anzufordern und daraufhin Spielsitzungsdaten zu verarbeiten. Sie können außerdem Funktionen zur Suche nach und zum Erhalten von Informationen über bestehende Spielsitzungen hinzufügen und Spielersitzungen anfordern, die effektiv einen Spielerplatz in einer bestehenden Spielsitzung reservieren.
3. Übermitteln Sie Verbindungsinformationen zurück an den Spiel-Client. Der Backend-Service empfängt Objekte für Spielsitzungen und Spielersitzungen als Antwort auf Anfragen an den GameLift Amazon-Dienst. Diese Objekte enthalten Informationen, insbesondere Verbindungsdetails (IP-Adresse und Port) und Spielersitzungs-ID, die der Spiel-Client benötigt, um sich mit der Spielsitzung auf einem Echtzeit-Server zu verbinden.

Stellen Sie eine Verbindung zu Spielen auf Echtzeitservern her

Ermöglichen Sie Ihrem Spielclient, sich direkt mit einer gehosteten Spielsitzung auf einem Echtzeitserver zu verbinden und Nachrichten mit dem Server und anderen Spielern auszutauschen.

1. Hol dir das Realtime Client SDK, erstelle es und füge es zu deinem Game-Client-Projekt hinzu. In der README-Datei finden Sie weitere Informationen zu den SDK-Anforderungen und Anweisungen zum Erstellen der Client-Bibliotheken.
2. Rufen Sie Client() mit einer Client-Konfiguration auf, die den zu verwendenden Client/Server-Verbindungstyp angibt.

   > **Note**
   >
   > Wenn Sie eine Verbindung zu einem Echtzeitserver herstellen, der auf einer gesicherten Flotte mit einem TLS-Zertifikat ausgeführt wird, müssen Sie einen gesicherten Verbindungstyp angeben.

3. Fügen Sie die folgenden Funktionen Ihrem Spiel-Client hinzu. Weitere Informationen finden Sie unter Referenz zur Realtime Server-Client-API (C#).
   - Herstellen und Trennen einer Verbindung zu einem Spiel
     - Connect()
     - Disconnect()
   - Senden von Nachrichten an Zielempfänger
     - SendMessage()
   - Erhalten und Verarbeiten von Nachrichten
     - OnDataReceived()
   - Beitreten zu und Verlassen von Spielergruppen
     - JoinGroup()
     - RequestGroupMembership()
     - LeaveGroup()
4. Richten Sie nach Bedarf Ereignishandler für die Client-Callbacks ein. Siehe Referenz zur Realtime Server-Client-API (C#): Asynchrone Callbacks.

Bei der Arbeit mit Echtzeit-Flotten mit aktivierter TLS-Zertifikatgenerierung wird der Server automatisch mit dem TLS-Zertifikat authentifiziert. TCP- und UDP-Datenverkehr wird während der Übertragung verschlüsselt, um Sicherheit auf Transportebene zu gewährleisten. TCP-Datenverkehr wird mit TLS 1.2 verschlüsselt und UDP-Datenverkehr wird mit DTLS 1.2 verschlüsselt.

Beispiele für Spieleclients

Einfacher Echtzeitclient (C#)

Dieses Beispiel veranschaulicht eine grundlegende Game-Client-Integration mit dem Realtime Client SDK (C#). Wie gezeigt, initialisiert das Beispiel ein Realtime-Client-Objekt, richtet Event-Handler ein und implementiert die clientseitigen Callbacks, stellt eine Verbindung zu einem Realtime-Server her, sendet eine Nachricht und trennt die Verbindung.

```
using System;
using System.Text;
using Aws.GameLift.Realtime;
using Aws.GameLift.Realtime.Event;
using Aws.GameLift.Realtime.Types;

namespace Example
{
    /**
     * An example client that wraps the GameLift Realtime client SDK
     *
     * You can redirect logging from the SDK by setting up the LogHandler as such:
     * ClientLogger.LogHandler = (x) => Console.WriteLine(x);
     *
     */
    class RealTimeClient
    {
        public Aws.GameLift.Realtime.Client Client { get; private set; }

        // An opcode defined by client and your server script that represents a custom
 message type
        private const int MY_TEST_OP_CODE = 10;

        /// Initialize a client for GameLift Realtime and connect to a player session.
        /// <param name="endpoint">The DNS name that is assigned to Realtime server</
param>
        /// <param name="remoteTcpPort">A TCP port for the Realtime server</param>
        /// <param name="listeningUdpPort">A local port for listening to UDP traffic</
param>
        /// <param name="connectionType">Type of connection to establish between client
 and the Realtime server</param>
        /// <param name="playerSessionId">The player session ID that is assigned to the
 game client for a game session </param>
```

```
       /// <param name="connectionPayload">Developer-defined data to be used during
client connection, such as for player authentication</param>
      public RealTimeClient(string endpoint, int remoteTcpPort, int listeningUdpPort,
ConnectionType connectionType,
                    string playerSessionId, byte[] connectionPayload)
       {
           // Create a client configuration to specify a secure or unsecure connection
type
           // Best practice is to set up a secure connection using the connection type
RT_OVER_WSS_DTLS_TLS12.
           ClientConfiguration clientConfiguration = new ClientConfiguration()
     {
               // C# notation to set the field ConnectionType in the new instance of
ClientConfiguration
         ConnectionType = connectionType
     };

           // Create a Realtime client with the client configuration
           Client = new Client(clientConfiguration);

           // Initialize event handlers for the Realtime client
           Client.ConnectionOpen += OnOpenEvent;
           Client.ConnectionClose += OnCloseEvent;
           Client.GroupMembershipUpdated += OnGroupMembershipUpdate;
           Client.DataReceived += OnDataReceived;

           // Create a connection token to authenticate the client with the Realtime
server
           // Player session IDs can be retrieved using AWS SDK for GameLift
           ConnectionToken connectionToken = new ConnectionToken(playerSessionId,
connectionPayload);

           // Initiate a connection with the Realtime server with the given connection
information
           Client.Connect(endpoint, remoteTcpPort, listeningUdpPort, connectionToken);
       }

       public void Disconnect()
       {
           if (Client.Connected)
           {
               Client.Disconnect();
           }
       }
```

```
        public bool IsConnected()
        {
            return Client.Connected;
        }

        /// <summary>
        /// Example of sending to a custom message to the server.
        ///
        /// Server could be replaced by known peer Id etc.
        /// </summary>
        /// <param name="intent">Choice of delivery intent i.e. Reliable, Fast etc. </
param>
        /// <param name="payload">Custom payload to send with message</param>
        public void SendMessage(DeliveryIntent intent, string payload)
        {
            Client.SendMessage(Client.NewMessage(MY_TEST_OP_CODE)
                .WithDeliveryIntent(intent)
                .WithTargetPlayer(Constants.PLAYER_ID_SERVER)
                .WithPayload(StringToBytes(payload)));
        }

        /**
         * Handle connection open events
         */
        public void OnOpenEvent(object sender, EventArgs e)
        {
        }

        /**
         * Handle connection close events
         */
        public void OnCloseEvent(object sender, EventArgs e)
        {
        }

        /**
         * Handle Group membership update events
         */
        public void OnGroupMembershipUpdate(object sender, GroupMembershipEventArgs e)
        {
        }

        /**
```

```
         *  Handle data received from the Realtime server
         */
        public virtual void OnDataReceived(object sender, DataReceivedEventArgs e)
        {
            switch (e.OpCode)
            {
                // handle message based on OpCode
                default:
                    break;
            }
        }

        /**
         * Helper method to simplify task of sending/receiving payloads.
         */
        public static byte[] StringToBytes(string str)
        {
            return Encoding.UTF8.GetBytes(str);
        }

        /**
         * Helper method to simplify task of sending/receiving payloads.
         */
        public static string BytesToString(byte[] bytes)
        {
            return Encoding.UTF8.GetString(bytes);
        }
    }
}
```

Erstellen eines Echtzeit-Skripts

Um Echtzeitserver für Ihr Spiel zu verwenden, müssen Sie ein Skript (in Form von JavaScript Code) bereitstellen, um eine Flotte von Echtzeitservern zu konfigurieren und optional anzupassen. In diesem Thema werden die wichtigsten Schritte beim Erstellen eines Echtzeit-Skripts behandelt. Sobald das Skript fertig ist, laden Sie es in den GameLift Amazon-Service hoch und verwenden Sie es, um eine Flotte zu erstellen (sieheLaden Sie ein Realtime Server-Skript auf Amazon hoch GameLift).

Um ein Skript für die Verwendung mit Echtzeitservern vorzubereiten, fügen Sie Ihrem Echtzeitskript die folgende Funktionalität hinzu.

Den Lebenszyklus der Spielsitzung verwalten (erforderlich)

Ein Realtime-Skript muss mindestens die `Init()` Funktion enthalten, die den Realtime-Server darauf vorbereitet, eine Spielsitzung zu starten. Es wird auch dringend empfohlen, dass Sie auch eine Möglichkeit zum Beenden von Spielsitzungen bereitstellen, um sicherzustellen, dass weiterhin neue Spielsitzungen auf Ihrer Flotte gestartet werden können.

Wenn die `Init()` Callback-Funktion aufgerufen wird, wird ein Realtime-Sitzungsobjekt übergeben, das eine Schnittstelle für den Realtime-Server enthält. Weitere Details zu dieser Schnittstelle finden Sie unter Serverschnittstelle in Echtzeit.

Um eine Spielsitzung ordnungsgemäß zu beenden, muss das Skript auch die Funktion des Realtime-Servers aufrufen. `session.processEnding` Dies erfordert einen Mechanismus, um zu bestimmen, wann eine Sitzung beendet werden soll. Der Beispielcode des Skripts veranschaulicht einen einfachen Mechanismus, der nach Spielerverbindungen sucht und die Beendigung der Spielsitzung auslöst, wenn für eine bestimmte Zeitspanne keine Spieler mit der Sitzung verbunden waren.

Echtzeitserver mit der grundlegendsten Konfiguration — Initialisierung und Beendigung von Serverprozessen — agieren im Wesentlichen als statuslose Relay-Server. Der Echtzeitserver leitet Nachrichten und Spieldaten zwischen den Spielclients weiter, die mit dem Spiel verbunden sind, ergreift jedoch keine unabhängigen Maßnahmen, um Daten zu verarbeiten oder Logik auszuführen. Sie können optional Spielelogik hinzufügen, die durch Spieleereignisse oder andere Mechanismen ausgelöst wird, wie für Ihr Spiel erforderlich.

Serverseitige Spiellogik hinzufügen (optional)

Sie können Ihrem Echtzeit-Skript optional Spiellogik hinzufügen. Beispielsweise können Sie eine oder alle der folgenden Aktionen durchführen. Der Beispielcode des Skripts dient der Veranschaulichung. Siehe Amazon GameLift Realtime Server-Skriptreferenz.

- Ereignisgesteuerte Logik hinzufügen. Implementierung der Callback-Funktionen zur Reaktion auf Client-Server-Ereignisse. Eine vollständige Liste der Callbacks finden Sie unter Script-Callbacks für Echtzeitserver.
- Auslösen einer Logik durch Senden von Nachrichten an den Server. Erstellen Sie eine Reihe von speziellen Operationscodes für Nachrichten, die von Spiel-Clients an den Server gesendet werden, und fügen Sie Funktionen zur Bearbeitung der Nachrichten hinzu. Verwenden Sie den Callback `onMessage` und analysieren Sie den Nachrichteninhalt über die `gameMessage`-Schnittstelle (siehe gameMessage.opcode).

- Aktiviere die Spiellogik, um auf deine anderen AWS Ressourcen zuzugreifen. Details hierzu finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.
- Erlaube der Spiellogik, auf die Flotteninformationen der Instanz zuzugreifen, auf der sie ausgeführt wird. Details hierzu finden Sie unter Flottendaten für eine GameLift Amazon-Instance abrufen.

Beispiel für ein Realtime Server-Skript

Dieses Beispiel veranschaulicht ein grundlegendes Skript, das für die Bereitstellung von Echtzeitservern erforderlich ist, sowie einige benutzerdefinierte Logik. Es enthält die erforderliche `Init()`-Funktion und verwendet einen Timermechanismus, um die Beendigung der Spielsitzung basierend auf der Zeitspanne ohne Spielerverbindungen auszulösen. Es enthält außerdem einige Hooks für eine benutzerdefinierte Logik, einschließlich einiger Callback-Implementierungen.

```
// Example Realtime Server Script
'use strict';

// Example override configuration
const configuration = {
    pingIntervalTime: 30000,
    maxPlayers: 32
};

// Timing mechanism used to trigger end of game session. Defines how long, in
 milliseconds, between each tick in the example tick loop
const tickTime = 1000;

// Defines how to long to wait in Seconds before beginning early termination check in
 the example tick loop
const minimumElapsedTime = 120;

var session;                        // The Realtime server session object
var logger;                         // Log at appropriate level
 via .info(), .warn(), .error(), .debug()
var startTime;                      // Records the time the process started
var activePlayers = 0;              // Records the number of connected players
var onProcessStartedCalled = false; // Record if onProcessStarted has been called

// Example custom op codes for user-defined messages
// Any positive op code number can be defined here. These should match your client
 code.
const OP_CODE_CUSTOM_OP1 = 111;
```

```
const OP_CODE_CUSTOM_OP1_REPLY = 112;
const OP_CODE_PLAYER_ACCEPTED = 113;
const OP_CODE_DISCONNECT_NOTIFICATION = 114;

// Example groups for user-defined groups
// Any positive group number can be defined here. These should match your client code.
// When referring to user-defined groups, "-1" represents all groups, "0" is reserved.
const RED_TEAM_GROUP = 1;
const BLUE_TEAM_GROUP = 2;

// Called when game server is initialized, passed server's object of current session
function init(rtSession) {
    session = rtSession;
    logger = session.getLogger();
}

// On Process Started is called when the process has begun and we need to perform any
// bootstrapping.  This is where the developer should insert any code to prepare
// the process to be able to host a game session, for example load some settings or set
 state
//
// Return true if the process has been appropriately prepared and it is okay to invoke
 the
// GameLift ProcessReady() call.
function onProcessStarted(args) {
    onProcessStartedCalled = true;
    logger.info("Starting process with args: " + args);
    logger.info("Ready to host games...");

    return true;
}

// Called when a new game session is started on the process
function onStartGameSession(gameSession) {
    // Complete any game session set-up

    // Set up an example tick loop to perform server initiated actions
    startTime = getTimeInS();
    tickLoop();
}

// Handle process termination if the process is being terminated by GameLift
// You do not need to call ProcessEnding here
function onProcessTerminate() {
```

```
    // Perform any clean up
}

// Return true if the process is healthy
function onHealthCheck() {
    return true;
}

// On Player Connect is called when a player has passed initial validation
// Return true if player should connect, false to reject
function onPlayerConnect(connectMsg) {
    // Perform any validation needed for connectMsg.payload, connectMsg.peerId
    return true;
}

// Called when a Player is accepted into the game
function onPlayerAccepted(player) {
    // This player was accepted -- let's send them a message
    const msg = session.newTextGameMessage(OP_CODE_PLAYER_ACCEPTED, player.peerId,
                                           "Peer " + player.peerId + " accepted");
    session.sendReliableMessage(msg, player.peerId);
    activePlayers++;
}

// On Player Disconnect is called when a player has left or been forcibly terminated
// Is only called for players that actually connected to the server and not those
 rejected by validation
// This is called before the player is removed from the player list
function onPlayerDisconnect(peerId) {
    // send a message to each remaining player letting them know about the disconnect
    const outMessage = session.newTextGameMessage(OP_CODE_DISCONNECT_NOTIFICATION,
                                                  session.getServerId(),
                                                  "Peer " + peerId + " disconnected");
    session.getPlayers().forEach((player, playerId) => {
        if (playerId != peerId) {
            session.sendReliableMessage(outMessage, playerId);
        }
    });
    activePlayers--;
}

// Handle a message to the server
function onMessage(gameMessage) {
    switch (gameMessage.opCode) {
```

```
      case OP_CODE_CUSTOM_OP1: {
        // do operation 1 with gameMessage.payload for example sendToGroup
        const outMessage = session.newTextGameMessage(OP_CODE_CUSTOM_OP1_REPLY,
 session.getServerId(), gameMessage.payload);
        session.sendGroupMessage(outMessage, RED_TEAM_GROUP);
        break;
      }
    }
}

// Return true if the send should be allowed
function onSendToPlayer(gameMessage) {
    // This example rejects any payloads containing "Reject"
    return (!gameMessage.getPayloadAsText().includes("Reject"));
}

// Return true if the send to group should be allowed
// Use gameMessage.getPayloadAsText() to get the message contents
function onSendToGroup(gameMessage) {
    return true;
}

// Return true if the player is allowed to join the group
function onPlayerJoinGroup(groupId, peerId) {
    return true;
}

// Return true if the player is allowed to leave the group
function onPlayerLeaveGroup(groupId, peerId) {
    return true;
}

// A simple tick loop example
// Checks to see if a minimum amount of time has passed before seeing if the game has
 ended
async function tickLoop() {
    const elapsedTime = getTimeInS() - startTime;
    logger.info("Tick... " + elapsedTime + " activePlayers: " + activePlayers);

    // In Tick loop - see if all players have left early after a minimum period of time
 has passed
    // Call processEnding() to terminate the process and quit
    if ( (activePlayers == 0) && (elapsedTime > minimumElapsedTime)) {
        logger.info("All players disconnected. Ending game");
```

```
        const outcome = await session.processEnding();
        logger.info("Completed process ending with: " + outcome);
        process.exit(0);
    }
    else {
        setTimeout(tickLoop, tickTime);
    }
}

// Calculates the current time in seconds
function getTimeInS() {
    return Math.round(new Date().getTime()/1000);
}

exports.ssExports = {
    configuration: configuration,
    init: init,
    onProcessStarted: onProcessStarted,
    onMessage: onMessage,
    onPlayerConnect: onPlayerConnect,
    onPlayerAccepted: onPlayerAccepted,
    onPlayerDisconnect: onPlayerDisconnect,
    onSendToPlayer: onSendToPlayer,
    onSendToGroup: onSendToGroup,
    onPlayerJoinGroup: onPlayerJoinGroup,
    onPlayerLeaveGroup: onPlayerLeaveGroup,
    onStartGameSession: onStartGameSession,
    onProcessTerminate: onProcessTerminate,
    onHealthCheck: onHealthCheck
};
```

Integrieren von Spielen mit dem Amazon GameLift -Plugin für Unity

Die Themen in diesem Abschnitt beschreiben das Amazon- GameLift Plugin für Unity und wie Sie es verwenden können, um Ihr Multiplayer-Spielprojekt für das Hosting mit Amazon vorzubereiten GameLift. Arbeiten Sie vollständig in Ihrer Unity-Entwicklungsumgebung mit den geführten Workflows des Plugins, um die grundlegenden Anforderungen für das Hosting mit Amazon zu erfüllen GameLift.

Amazon GameLift ist ein vollständig verwalteter Service, mit dem Spieleentwickler dedizierte Spieleserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können. Weitere Informationen zum Amazon GameLift -Hosting finden Sie unter So GameLift funktioniert Amazon.

- Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x, Version 2.0.0, funktioniert mit dem Server-SDK 5.x und unterstützt Amazon GameLift Anywhere.
- Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 4.x, Version 1.0.0, funktioniert mit dem Server-SDK 4.x oder früher. Diese Version verwendet Amazon GameLift Local für Integrationstests.

Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x

Amazon GameLift bietet Tools zur Vorbereitung Ihrer Multiplayer-Spielserver auf die Arbeit mit Amazon GameLift. Das Amazon- GameLift Plugin für Unity erleichtert die Integration von Amazon GameLift in Ihre Unity-Spielprojekte, das Testen Ihrer Integration mit Amazon und GameLift Anywheredie Bereitstellung von Amazon GameLift -Ressourcen für das Cloud-Hosting.

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

Themen

Informationen zum Plugin

Das Plugin für Unity bietet eine optimierte Einstiegserfahrung für die Integration und das Hosten Ihrer Unity-Multiplayer-Spiele mit Amazon GameLift. Sie können die Plugin-Funktionalität und die vorgefertigten Komponenten nutzen, um Ihre Spiele schnell zum Laufen zu bringen.

Das Plugin fügt dem Unity-Editor Tools und Funktionen hinzu. Verwenden Sie die geführten Workflows, um Amazon GameLift in Ihr Spielprojekt zu integrieren, es lokal zu testen und dann den Spieleserver im Amazon GameLift -Cloud-Hosting bereitzustellen.

Verwenden Sie die vordefinierten Hosting-Lösungen des Plugins, um Ihr Spiel bereitzustellen. Richten Sie eine Amazon GameLift Anywhere-Flotte mit Ihrer lokalen Workstation als Host ein.

Wählen Sie für das Cloud-Hosting aus zwei gängigen Bereitstellungsszenarien, die die Spielerlatenz, die Verfügbarkeit von Spielsitzungen und die Kosten auf unterschiedliche Weise ausgleichen. Ein Szenario umfasst einen einfachen FlexMatch Matchmaker und einen Regelsatz. Verwenden Sie diese Szenarien, um eine grundlegende produktionsbereite Hosting-Lösung einzurichten, und optimieren und passen Sie sie dann nach Bedarf an.

Das Plugin enthält die folgenden Komponenten:

- Plugin-Module für den Unity-Editor. Wenn das Plugin installiert ist, erhalten Sie über ein neues Hauptmenüelement Zugriff auf die Amazon- GameLift Funktionalität.
- C#-Bibliotheken für die Amazon- GameLift Service-API mit clientseitiger Funktionalität.
- C#-Bibliotheken für das Amazon GameLift Server SDK (Version 5.x).
- Beispiel für Spielinhalte, einschließlich Komponenten und Szenen, sodass Sie Amazon ausprobieren können, GameLift auch wenn Sie kein aufbaubereites Multiplayer-Spiel haben.
- Lösungskonfigurationen, die als AWS CloudFormation Vorlagen bereitgestellt werden, die das Plugin verwendet, wenn Sie Ihren Spieleserver zum Hosten in der Cloud bereitstellen.

Plugin-Workflow

In den folgenden Schritten wird ein typischer Ansatz für die Integration und Bereitstellung eines Spielprojekts mit dem Amazon GameLift -Plugin für Unity beschrieben. Sie führen diese Schritte aus, indem Sie im Unity-Editor und in Ihrem Spielcode arbeiten.

1. Erstellen Sie ein Benutzerprofil, das mit Ihrem -AWSKonto verknüpft ist und Anmeldeinformationen für einen gültigen -Kontobenutzer mit Berechtigungen zur Verwendung von Amazon bereitstellt GameLift.
2. Fügen Sie Ihrem Spieleprojekt Servercode hinzu, um die Kommunikation zwischen einem laufenden Spieleserver und dem mit dem Amazon- GameLift Service herzustellen.
3. Fügen Sie Ihrem Spielprojekt Client-Code hinzu, mit dem Spielclients Anfragen an Amazon senden können, GameLift um eine Spielsitzung zu starten oder ihr beizutreten und dann eine Verbindung zum Spielserver herzustellen.
4. Verwenden Sie den Anywhere-Workflow, um Ihre lokale Workstation als Anywhere-Host für Ihren Spieleserver einzurichten. Starten Sie Ihren Spieleserver und Ihren Client lokal, stellen Sie eine Verbindung zu einer Spielsitzung her und testen Sie Ihre Integration.
5. Verwenden Sie den EC2-Hosting-Workflow, um Ihren integrierten Spieleserver hochzuladen und eine Cloud-Hosting-Lösung bereitzustellen. Wenn Ihr Spieleserver bereit ist, starten Sie Ihren

Spiele-Client lokal, stellen Sie eine Verbindung zu einer Spielsitzung her und melden Sie sich an und spielen Sie das Spiel.

Wenn Sie im Plugin arbeiten, erstellen und verwenden Sie -AWSRessourcen. Für diese Aktionen können Gebühren für das verwendete AWS Konto anfallen. Wenn Sie noch nicht mit vertraut sindAWS, können -Aktionen durch das AWS kostenlose Kontingent für abgedeckt sein.

Installieren des Plugins für Unity

In diesem Abschnitt wird beschrieben, wie Sie das Plugin zu einem Unity-Projekt hinzufügen. Nach der Installation des Plugins ist die Plugin-Funktionalität verfügbar, wenn Sie das Projekt im Unity-Editor geöffnet haben.

Bevor Sie beginnen

So verwenden Sie das Amazon- GameLift Plugin für Unity:

- Unity für Windows 2022 LTS oder Unity für MacOS
- Amazon GameLift -Plugin für Unity-Download. [Website herunterladen] Der Download enthält zwei Pakete:
  - GameLift eigenständiges Amazon-Plugin für Unity
  - Amazon GameLift -C#-Server-SDK für Unity
- Microsoft Visual Studio 2019 oder höher.
- Ein Multiplayer-Spielprojekt mit C#-Spielcode.
- Die Registrierung des Drittanbieters UnityNuGet. Dieses Tool verwaltet DLLs von Drittanbietern. Weitere Informationen finden Sie im UnityNuGet Github-Repository.

Fügen Sie das Plugin zu Ihrem Spielprojekt hinzu

Führen Sie die folgenden Aufgaben aus und arbeiten Sie im Unity-Editor und in Ihren Spielprojektdateien.

Schritt 1: UnityNuGet Zu Ihrem Spielprojekt hinzufügen

Wenn Sie noch nicht für Ihr Spielprojekt UnityNuGet eingerichtet haben, führen Sie die folgenden Schritte aus, um das Tool mit dem Unity-Paketmanager zu installieren. Alternativ können Sie die NuGet CLI verwenden, um die DLLs manuell herunterzuladen. Weitere Informationen finden Sie im Amazon GameLift -C#-Server-SDK für UnityREADME.

1. Wenn Ihr Projekt im Unity-Editor geöffnet ist, gehen Sie zum Hauptmenü und wählen Sie Bearbeiten, Projekteinstellungen aus. Wählen Sie in den Optionen den Abschnitt Paket-Manager aus und öffnen Sie die Gruppe Scoped Registries.
2. Wählen Sie die Schaltfläche + und geben Sie die folgenden Werte für die UnityNuGet bereichsbezogene Registrierung ein:

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

Für Benutzer der Unity-2021-Version:

Überprüfen Sie nach dem Einrichten von UnityNuGet, ob in der Unity-Konsole `Assembly Version Validation` Fehler angezeigt werden. Diese Fehler treten auf, wenn das Binden von Umleitungen für stark benannte Baugruppen in den NuGet Paketen nicht korrekt in Pfade innerhalb Ihres Unity-Projekts aufgelöst wird. Um dieses Problem zu beheben, konfigurieren Sie die Validierung der Assembly-Version von Unity:

1. Gehen Sie im Unity-Editor zum Hauptmenü und wählen Sie Bearbeiten, Projekteinstellungen und öffnen Sie den Abschnitt Player.
2. Deaktivieren Sie die Option Validierung der Baugruppenversion.

Schritt 2: Hinzufügen der Plugin- und C#-Server-SDK-Pakete

1. Entpacken Sie das Amazon GameLift -Plugin für den Unity-Download, der beide Pakete enthält.
2. Wenn Ihr Projekt im Unity Editor geöffnet ist, gehen Sie zum Hauptmenü und wählen Sie Window, Package Manager aus.
3. Wählen Sie die Schaltfläche +, um ein neues Paket hinzuzufügen. Wählen Sie die Option Paket aus Tarball hinzufügen aus.
4. Suchen Sie unter Pakete auf Datenträger auswählen das Amazon GameLift -C#-Server-SDK-Plugin für Unity-Downloaddateien und wählen Sie die `com.amazonaws.gameliftserver.sdk-<version>.tgz` Datei aus. Wählen Sie Öffnen, um das Plugin zu installieren.
5. Suchen Sie unter Pakete auf Datenträger auswählen das GameLift eigenständige Amazon-Plugin für Unity-Downloaddateien und wählen Sie die Datei aus`com.amazonaws.gamelift-<version>.tgz`. Wählen Sie Öffnen, um das Plugin zu installieren.

6. Stellen Sie sicher, dass das eigenständige Plugin zu Ihrem Projekt hinzugefügt wurde. Kehren Sie zum Unity-Editor-Fenster zurück. Überprüfen Sie das Hauptmenü auf die neue Schaltfläche im Amazon GameLift-Menü.

Schritt 3: Importieren des Beispielspiels (optional)

Das Plugin für Unity enthält eine Reihe von Beispielspielkomponenten, einschließlich Szenen, die Sie Ihrem Spielprojekt hinzufügen können. Das Importieren des Beispielspiels bietet Ihnen einen schnellen Weg zum Testen, Erstellen und Bereitstellen eines einfachen Multiplayer-Spiels mit Amazon GameLift. Das Beispielspiel ist bereits vollständig in Amazon GameLift SDKs integriert, sodass Sie die Integrationsaufgaben überspringen und die verbleibenden Workflow-Aufgaben abschließen können.

Wenn Sie das Beispielspiel verwenden, können Sie in nur wenigen Minuten eine lokal gehostete Amazon GameLift-Anywhere-Flotte einrichten und ihr beitreten. Sie können das Spiel auf Amazon GameLift bereitstellen und in weniger als einer Stunde an einem in der Cloud gehosteten Live-Spiel teilnehmen.

So importieren Sie das Beispielspiel:

1. Wenn Ihr Spielprojekt im Unity Editor geöffnet ist, gehen Sie zum Menü Amazon GameLift und wählen Sie Beispielspiel, Beispielspiel importieren aus.
2. Nachdem die Dateien importiert wurden, wechseln Sie erneut zum Amazon GameLift-Menü und wählen Sie Beispielspiel, Einstellungen initialisieren aus. In diesem Schritt wird Ihr Projekt für die Erstellung des Spielclients und des Servers konfiguriert.

Wenn die Installation abgeschlossen ist, werden Ihrem Spielprojekt zwei neue Szenen hinzugefügt. Sie sehen auch einige zusätzliche Projektressourcen, einschließlich einer GameLiftClientSettings Komponente.

Weitere Informationen zur Benutzeroberfläche und zum Gameplay des Beispiels finden Sie in der Beispiel-Spiellesedatei.

Einrichten eines AWS-Benutzerprofils

Richten Sie nach der Installation des Plugins ein Profil ein und verknüpfen Sie es mit einem gültigen AWS Kontobenutzer. Sie können mehrere Profile verwalten, aber Sie können jeweils nur ein Profil aktiv haben. Wenn Sie im Plugin arbeiten, wählen Sie ein zu verwendendes Profil aus.

Durch die Pflege mehrerer Profile können Sie zwischen verschiedenen Hosting-Szenarien wechseln. Sie können beispielsweise Profile mit denselben AWS Anmeldeinformationen, aber unterschiedlichen AWS Regionen einrichten. Oder Sie können Profile mit unterschiedlichen AWS Konten oder mit unterschiedlichen Benutzern/Berechtigungssätzen einrichten.

> **Note**
>
> Wenn Sie die AWS CLI auf Ihrer Workstation installiert haben und bereits ein Profil konfiguriert ist, kann das Amazon GameLift -Plugin es erkennen und als vorhandenes Profil auflisten. Das Plugin wählt automatisch jedes Profil mit dem Namen aus`[default]`. Sie können ein vorhandenes Profil verwenden oder ein neues erstellen.

So richten Sie Ihr AWS Profil ein

1. Wählen Sie im Hauptmenü Unity-Editor Amazon GameLift und dann AWS Kontoprofile festlegen aus. Diese Aktion öffnet das Plugin-Fenster. Öffnen Sie die Seite AWS Benutzerprofile.
2. Wenn das Plugin ein vorhandenes Profil erkennt, werden Sie nicht aufgefordert, eines zu erstellen. Wählen Sie ein vorhandenes Profil aus oder wählen Sie Ein weiteres Profil hinzufügen, um ein neues Profil zu erstellen.
3. Wenn das Plugin kein vorhandenes Profil erkennt, werden Sie aufgefordert, eines zu erstellen. Sie können ein neues Profil entweder mit einem neuen oder einem vorhandenen AWS Konto erstellen.

   > **Note**
   >
   > Sie müssen die -AWSManagementkonsole verwenden, um ein neues AWS Konto zu erstellen und einen Benutzer mit dem richtigen Berechtigungssatz zu erstellen oder zu aktualisieren.

   Beim Einrichten eines Profils benötigen Sie die folgenden Informationen:

   - Ein AWS-Konto. Wenn Sie ein neues AWS Konto erstellen müssen, folgen Sie den Anweisungen, um das Konto zu erstellen. Weitere Informationen finden Sie unter Erstellen eines -AWSKontos.

- Ein -AWSKontobenutzer mit Berechtigungen zur Nutzung von Amazon GameLift und anderen erforderlichen AWS Services. Richten Sie ein AWS-Konto Anweisungen zum Einrichten eines AWS Identity and Access Management (IAM)-Benutzers mit Amazon- GameLift Berechtigungen finden Sie unter .
- Anmeldeinformationen für Ihren AWS Benutzer. Dieser Benutzer benötigt auch programmgesteuerten Zugriff mit langfristigen Anmeldeinformationen. Diese Anmeldeinformationen bestehen aus einer AWS Zugriffsschlüssel-ID und einem AWS geheimen Schlüssel. Weitere Informationen finden Sie unter Abrufen Ihrer Zugriffsschlüssel.
- Region AWS: Dies ist ein geografischer Standort, an dem Sie Ihre AWS Ressourcen für das Hosting erstellen möchten. Während der Entwicklung empfehlen wir, eine Region in der Nähe Ihres physischen Standorts zu verwenden, um die Latenz zu minimieren. Sehen Sie sich die Liste der unterstützten AWS Regionen an.

4. Wenn Sie ein Profil ausgewählt oder erstellt haben, überprüfen Sie den Bootstrap-Status des Profils und ergreifen Sie bei Bedarf Maßnahmen. Alle Profile müssen gestartet werden, um die Amazon- GameLift Plug-In-Funktionalität nutzen zu können.

So bootstrappen Sie Ihr Profil:

Bootstrapping bezeichnet einen Amazon S3-Bucket zur Verwendung mit dem ausgewählten Benutzerprofil. Amazon S3 ist ein AWS Kernservice für Daten- und Objektspeicherung. Der Bucket, der zum Speichern von Projektkonfigurationen, Build-Artefakten und anderen Abhängigkeiten verwendet wird. Buckets werden nicht zwischen anderen Profilen geteilt.

> **Note**
>
> Bootstrapping erstellt neue AWS Ressourcen und kann Kosten verursachen.

1. Wenn Sie Ihre Profile im Plugin-Fenster AWS Benutzerprofile anzeigen, wählen Sie das Profil aus, das Sie verwenden möchten. Eine Warnmeldung wird angezeigt, wenn das Profil noch nicht gestartet wurde.
2. Wählen Sie im Abschnitt Bootstrap Ihres Profils ein Profil aus der Dropdown-Liste aus und überprüfen Sie den Bootstrap-Status. Wenn der Status angibt, dass kein Bucket vorhanden ist, wählen Sie die Schaltfläche Bootstrap-Profil . Sie können den Bucket-Namen auf einen neuen Bucket-Namen festlegen, einen vorhandenen Bucket eingeben, auf den Sie Zugriff haben, oder den automatisch generierten Namen beibehalten.

3. Warten Sie, bis sich der Bootstrap-Status in „Aktiv“ ändert. Dies kann einige Minuten dauern. Wenn der Status „Aktiv“ lautet, können Sie das Profil verwenden, um mit Plugin-Funktionen zu arbeiten

Einrichten Ihres Spiels für lokale Tests mit Amazon GameLift Anywhere

In diesem Workflow fügen Sie Client- und Server-Spielcode für die Amazon GameLift-Funktionalität hinzu und verwenden das Plugin, um Ihre lokale Workstation als Testspielserver-Host zu kennzeichnen. Wenn Sie die Integrationsaufgaben abgeschlossen haben, verwenden Sie das -Plugin, um Ihre Spielclient- und Serverkomponenten zu erstellen.

So starten Sie den Amazon GameLift -Anywhere-Workflow:

- Wählen Sie im Hauptmenü Unity Editor Amazon GameLift und dann Host with Anywhere aus. Diese Aktion öffnet die Plugin-Seite zum Einrichten Ihres Spiels mit einer @Anywhere-Flotte. Die Seite bietet einen fünfstufigen Prozess zum Integrieren, Erstellen und Starten Ihrer Spielkomponenten.

Festlegen Ihres Profils

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

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

Integrieren Ihres Spiels mit Amazon GameLift

> **Note**
>
> Wenn Sie das Beispielspiel importiert haben, können Sie diesen Schritt überspringen. Die Beispielspielkomponenten verfügen bereits über den erforderlichen Server- und Clientcode.

Für diesen Schritt im Workflow nehmen Sie Aktualisierungen am Client- und Servercode in Ihrem Spielprojekt vor.

- * Spieleserver müssen mit dem Amazon- GameLift Service kommunizieren können, um Aufforderungen zu erhalten, eine Spielsitzung zu starten, Verbindungsinformationen für Spielsitzungen bereitzustellen und den Status zu melden.
- Spielclients müssen in der Lage sein, Informationen über Spielsitzungen abzurufen, Spielsitzungen beizutreten oder zu starten und Verbindungsinformationen abzurufen, um einem Spiel beizutreten.

Integrieren Ihres Servercodes

Wenn Sie Ihr eigenes Spielprojekt mit benutzerdefinierten Szenen verwenden, verwenden Sie den bereitgestellten Beispielcode, um Ihrem Spielprojekt den erforderlichen Servercode hinzuzufügen:

1. Öffnen Sie in Ihren Spielprojektdateien den `Assets/Scripts/Server` Ordner . Wenn es nicht existiert, erstellen Sie es.
2. Gehen Sie zum GitHub Repository aws/amazon-gamelift-plugin-unity und öffnen Sie den Pfad `Samples~/SampleGame/Assets/Scripts/Server`.
3. Suchen Sie die Datei GameLiftServer.cs. und kopieren Sie sie in den Ordner Server Ihres Spielprojekts. Wenn Sie eine ausführbare Serverdatei erstellen, verwenden Sie diese Datei als Build-Ziel.

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

- Initialisiert einen Amazon GameLift -API-Client. Der `InitSDK()` Aufruf mit Serverparametern ist für eine Amazon GameLift -Anywhere-Flotte erforderlich. Diese Einstellungen werden automatisch für die Verwendung im Plugin festgelegt.

- Implementiert die erforderlichen Rückruffunktionen, um auf Anfragen vom Amazon- GameLift Service zu antworten, einschließlich `OnStartGameSession``OnProcessTerminate`, und `onHealthCheck`.
- Ruft `ProcessReady()` mit einem bestimmten Port auf, um den Amazon- GameLift Service zu benachrichtigen, wenn der Serverprozess bereit ist, Spielsitzungen zu hosten.

Wenn Sie den Beispiel-Servercode anpassen möchten, sehen Sie sich die folgenden Ressourcen an:

- Füge Amazon GameLift zu deinem Gameserver hinzu
- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity

Integrieren Ihres Client-Codes

Wenn Sie Ihr eigenes Spielprojekt mit benutzerdefinierten Szenen verwenden, müssen Sie grundlegende Funktionen in Ihren Spielclient integrieren. Sie müssen auch UI-Elemente hinzufügen, damit sich Spieler anmelden und einer Spielsitzung beitreten können. Verwenden Sie die Amazon-GameLift Service-APIs (im AWS -SDK), um Informationen zu Spielsitzungen abzurufen, neue Spielsitzungen zu erstellen oder bestehende Spielsitzungen beizutreten.

Wenn Sie einen Client für lokale Tests mit einer Anywhere-Flotte erstellen, können Sie dem Amazon-GameLift Service direkte Aufrufe hinzufügen. Wenn Sie Ihr Spiel für das Cloud-Hosting entwickeln oder Anywhere-Flotten für das Produktions-Hosting verwenden möchten, müssen Sie einen clientseitigen Backend-Service erstellen, der die gesamte Kommunikation zwischen Spielclients und dem Amazon- GameLift Service übernimmt.

Verwenden Sie die folgenden Ressourcen als Leitfaden, um Amazon GameLift in Ihren Client-Code zu integrieren.

- Integrieren Sie den Client in die - GameLiftCoreApi Klasse im GitHub Repo aws/amazon-gamelift-plugin-unity. Diese Klasse bietet Steuerelemente für die Spielerauthentifizierung und für das Abrufen von Spielsitzungsinformationen.
- Sehen Sie sich Beispiel-Spielintegrationen an, die im GitHub Repository aws/amazon-gamelift-plugin-unity, verfügbar sind`Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`.
- Folgen Sie den Anweisungen unter Hinzufügen von Amazon GameLift zu Ihrem Unity-Spielclient.

Für Spieleclients, die eine Verbindung zu einer Anywhere-Flotte herstellen, benötigt Ihr Spieleclient die folgenden Informationen. Das Plugin aktualisiert Ihr Spielprojekt automatisch, um die Ressourcen zu verwenden, die Ihr im Plugin erstellt.

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

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

Herstellen einer Verbindung mit einer Anywhere-Flotte

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

- Wenn das AWS Konto, das Sie derzeit verwenden, über vorhandene Anywhere-Flotten verfügt, öffnen Sie das Dropdown-Feld Flottenname und wählen Sie eine Flotte aus. In diesem Dropdown-Menü werden nur die Anywhere-Flotten in der AWS Region für das aktuell aktive Benutzerprofil angezeigt.
- Wenn keine Flotten vorhanden sind oder Sie eine neue erstellen möchten, wählen Sie Neue Anywhere-Flotte erstellen und geben Sie einen Flottennamen an.

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

Registrieren einer Datenverarbeitung

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

1. Geben Sie einen Datenverarbeitungsnamen für Ihren lokalen Computer ein. Wenn Sie mehr als eine Rechenleistung in der Flotte hinzufügen, müssen die Namen eindeutig sein.
2. Wählen Sie Rechenleistung registrieren aus. Sie können den Fortschritt dieser Anforderung im Ausgabeprotokoll des Unreal-Editors verfolgen.

Das Plugin registriert Ihre lokale Workstation, wobei die IP-Adresse auf localhost (127.0.0.1) festgelegt ist. Bei dieser Einstellung wird davon ausgegangen, dass Sie Ihren Spielclient und -server auf demselben Computer ausführen.

Als Reaktion auf diese Aktion GameLift überprüft Amazon, ob es eine Verbindung zur Datenverarbeitung herstellen kann, und gibt Informationen über die neu registrierte Datenverarbeitung zurück.

Spiel starten

In diesem Schritt erstellen Sie Ihre Spielkomponenten und starten sie, um das Spiel zu spielen. Führen Sie die folgenden Schritte aus:

1. Konfigurieren Sie Ihren Spiele-Client. In diesem Schritt fordern Sie das Plugin auf, eine `GameLiftClientSettings` Komponente für Ihr Spielprojekt zu aktualisieren. Das Plugin verwendet diese Komponente, um bestimmte Informationen zu speichern, die Ihr Spielclient benötigt, um eine Verbindung zum Amazon- GameLift Service herzustellen.

    a. Wenn Sie das Beispielspiel nicht importiert und initialisiert haben, erstellen Sie eine neue `GameLiftClientSettings` Komponente. Wählen Sie im Hauptmenü Unity Editor die Option Assets, Create, GameLiftClient Settings aus. Wenn Sie mehrere Kopien von GameLiftClientSettings in Ihrem Projekt erstellen, erkennt das Plugin dies automatisch und benachrichtigt Sie, welche Komponente das Plugin aktualisieren wird.

    b. Wählen Sie in Launch Game die Option Configure Client: Apply Anywhere Settings aus. Diese Aktion aktualisiert Ihre Spielclient-Einstellungen, um die soeben eingerichtete Anywhere-Flotte zu verwenden.
2. Erstellen und führen Sie Ihren Spiele-Client aus.

   a. Erstellen Sie eine ausführbare Client-Datei mit dem Standard-Unity-Build-Prozess. Wechseln Sie in Datei, Build-Einstellungen die Plattform zu Windows, Mac, Linux . Wenn Sie das Beispielspiel importiert und die Einstellungen initialisiert haben, werden die Build-Liste und das Build-Ziel automatisch aktualisiert.

   b. Starten Sie eine oder mehrere Instances der neu erstellten ausführbaren Spieleclient-Datei.

3. Starten Sie einen Spieleserver in Ihrer Anywhere-Flotte. Wählen Sie Server: Launch Server im Editor aus. Diese Aufgabe startet einen Live-Server, mit dem Ihr Client eine Verbindung herstellen kann, solange der Unity-Editor geöffnet bleibt.
4. Starten oder treten Sie einer Spielsitzung bei. Verwenden Sie in Ihren Spielclient-Instances die Benutzeroberfläche, um jeden Client einer Spielsitzung beizutreten. Wie Sie dies tun, hängt davon ab, wie Sie dem Client Funktionen hinzugefügt haben.

Wenn Sie den Beispiel-Spielclient verwenden, hat er die folgenden Merkmale:

- Eine Player-Anmeldekomponente. Wenn Sie eine Verbindung zu einem Spieleserver auf einer Anywhere-Flotte herstellen, gibt es keine Spielervalidierung. Sie können beliebige Werte eingeben, um der Spielsitzung beizutreten.
- Eine einfache Benutzeroberfläche für das Beitreten zu einem Spiel. Wenn ein Client versucht, einem Spiel beizutreten, sucht er automatisch nach einer aktiven Spielsitzung mit einem verfügbaren Spielerslot. Wenn keine Spielsitzung verfügbar ist, fordert der Client eine neue Spielsitzung an. Wenn eine Spielsitzung verfügbar ist, fordert der Client auf, der verfügbaren Spielsitzung beizutreten. Wenn Sie Ihr Spiel mit mehreren gleichzeitigen Clients testen, startet der erste Client die Spielsitzung und die verbleibenden Clients treten automatisch der vorhandenen Spielsitzung bei.
- Spielsitzungen mit vier Spielerslots. Sie können bis zu vier Spieleclient-Instances gleichzeitig starten und sie treten derselben Spielsitzung bei.

Starten von einer ausführbaren Serverdatei (optional)

Sie können Ihre ausführbare Spieleserver-Datei zum Testen auf einer Anywhere-Flotte erstellen und starten.

1. Erstellen Sie eine ausführbare Serverdatei mit dem Standard-Unity-Build-Prozess. Wechseln Sie in Datei, Build-Einstellungen zu Dedicated Server und erstellen Sie.

2. Rufen Sie ein kurzfristiges Authentifizierungstoken ab, indem Sie den AWS CLI-Befehl get-compute-auth-token mit Ihrer Anywhere-Flotten-ID und -AWSRegion aufrufen. Die Flotten-ID wird beim Erstellen der Flotte unter Mit einer Anywhere-Flotte verbinden angezeigt. Die AWS Region wird unter Profil festlegen angezeigt, wenn Sie Ihr aktives Profil auswählen.

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

3. Starten Sie die neu erstellte ausführbare Spieleserver-Datei über eine Befehlszeile und übergeben Sie ein gültiges Authentifizierungstoken.

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

Bereitstellen Ihres Spiels für das Cloud-Hosting mit verwalteten EC2-Flotten

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

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

- Wählen Sie im Hauptmenü Unity Editor Amazon GameLift und dann Host with Managed EC2 aus. Dieser Workflow bietet einen sechsstufigen Prozess zum Integrieren, Erstellen, Bereitstellen und Starten Ihrer Spielkomponenten.

Festlegen Ihres Profils

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

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

Integrieren Ihres Spiels mit Amazon GameLift

Für diese Aufgabe nehmen Sie Aktualisierungen am Client- und Servercode in Ihrem Spielprojekt vor.

- Spielserver müssen mit dem Amazon GameLift-Service kommunizieren können, um Aufforderungen zum Starten einer Spielsitzung zu erhalten, Verbindungsinformationen für Spielsitzungen bereitzustellen und den Status zu melden.
- Spielclients müssen in der Lage sein, Informationen über Spielsitzungen abzurufen, Spielsitzungen beizutreten oder zu starten und Verbindungsinformationen abzurufen, um einem Spiel beizutreten.

> **Note**
>
> Wenn Sie das Beispielspiel importiert haben, können Sie diesen Schritt überspringen. Die Beispielspielkomponenten verfügen bereits über den erforderlichen Server- und Clientcode.

Integrieren Ihres Servercodes

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

1. Öffnen Sie in Ihren Spielprojektdateien den `Assets/Scripts/Server` Ordner . Wenn es nicht existiert, erstellen Sie es.
2. Gehen Sie zum GitHub Repository aws/amazon-gamelift-plugin-unity und öffnen Sie den Pfad `Samples~/SampleGame/Assets/Scripts/Server`.
3. Suchen Sie die Datei `GameLiftServer.cs` und kopieren Sie sie in den `Server` Ordner Ihres Spielprojekts. Wenn Sie eine ausführbare Serverdatei erstellen, verwenden Sie diese Datei als Build-Ziel.

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

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

Wenn Sie den Beispiel-Servercode anpassen möchten, sehen Sie sich die folgenden Ressourcen an:

- Füge Amazon GameLift zu deinem Gameserver hinzu
- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity

Integrieren Ihres Client-Codes

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

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

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

Um diese Komponenten zu verwenden, benötigt Ihr Spielclient Funktionen, um Anforderungen an den Backend-Service zu senden, um Folgendes zu tun:

- Erstellen Sie einen Spielerbenutzer im AWS Cognito-Benutzerpool und authentifizieren Sie sich.

- Nehmen Sie an einer Spielsitzung teil und erhalten Sie Verbindungsinformationen.
- Nehmen Sie mit Matchmaking an einem Spiel teil.

Verwenden Sie die folgenden Ressourcen als Leitfaden.

- Integrieren Sie den Client in die - GameLiftCoreApiKlasse im GitHub Repository aws/amazon-gamelift-plugin-unity. Diese Klasse bietet Steuerelemente für die Spielerauthentifizierung und für das Abrufen von Spielsitzungsinformationen.
- Die Beispielintegrationen für Spiele finden Sie unter GitHub repo aws/amazon-gamelift-plugin-unity , `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`.
- Fügen Sie Ihrem Unity- GameLift Spielclient Amazon hinzu.

Bereitstellungsszenario auswählen

In diesem Schritt wählen Sie die Spiele-Hosting-Lösung aus, die Sie zu diesem Zeitpunkt bereitstellen möchten. Sie können mehrere Bereitstellungen Ihres Spiels haben, indem Sie eines der Szenarien verwenden.

- Flotte für eine einzelne Region: Stellt Ihren Spieleserver für eine einzelne Flotte von Hosting-Ressourcen in der AWS Standardregion des aktiven Profils bereit. Dieses Szenario ist ein guter Ausgangspunkt für das Testen Ihrer Serverintegration mit AWS und Server-Build-Konfiguration. Es stellt die folgenden Ressourcen bereit:
  - AWS -Flotte (On-Demand) mit installiertem und ausgeführtem Spieleserver-Build.
  - Amazon Cognito-Benutzerpool und -Client, damit Spieler ein Spiel authentifizieren und starten können.
  - API-Gateway-Genehmiger, der den Benutzerpool mit APIs verknüpft.
  - WebACl zur Drosselung übermäßiger Player-Aufrufe an das API-Gateway.
  - API-Gateway + Lambda-Funktion für Spieler, um einen Spielslot anzufordern. Diese Funktion ruft auf`CreateGameSession()`, wenn keine verfügbar sind.
  - API-Gateway + Lambda-Funktion für Spieler, um Verbindungsinformationen für ihre Spielanfrage abzurufen.
- FlexMatch Flotte: Stellt Ihren Spieleserver für eine Reihe von Flotten bereit und richtet einen FlexMatch Matchmaker mit Regeln ein, um Spielerübereinstimmungen zu erstellen. In diesem Szenario wird kostengünstiges Spot-Hosting mit einer Struktur mit mehreren Flotten und mehreren Standorten verwendet, um eine dauerhafte Verfügbarkeit zu gewährleisten. Dieser Ansatz ist

nützlich, wenn Sie bereit sind, mit der Entwicklung einer Matchmaker-Komponente für Ihre Hosting-Lösung zu beginnen. In diesem Szenario erstellen Sie die grundlegenden Ressourcen für diese Lösung, die Sie später nach Bedarf anpassen können. Es stellt die folgenden Ressourcen bereit:

- FlexMatch Matchmaking-Konfiguration und Matchmaking-Regelsatz zur Annahme von Spieleranfragen und Formularübereinstimmungen.
- Drei AWS Flotten mit installiertem und ausgeführtem Spieleserver-Build an mehreren Standorten. Enthält zwei Spot-Flotten und eine On-Demand-Flotte als Backup.
- AWS Platzierungswarteschlange für Spielsitzungen, die Anfragen nach vorgeschlagenen Übereinstimmungen erfüllt, indem sie die bestmögliche Hosting-Ressource findet (basierend auf Realisierbarkeit, Kosten, Spielerlatenz usw.) und eine Spielsitzung startet.
- Amazon Cognito-Benutzerpool und -Client, damit Spieler ein Spiel authentifizieren und starten können.
- API-Gateway-Genehmiger, der den Benutzerpool mit APIs verknüpft.
- WebACl zur Drosselung übermäßiger Player-Aufrufe an das API-Gateway.
- API-Gateway + Lambda-Funktion für Spieler, um einen Spielslot anzufordern. Diese Funktion ruft auf`StartMatchmaking()`.
- API-Gateway + Lambda-Funktion für Spieler, um Verbindungsinformationen für ihre Spielanfrage abzurufen.
- Amazon-DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Spielsitzungsinformationen.
- SNS-Thema + Lambda-Funktion zur Verarbeitung von GameSessionQueue Ereignissen.

Festlegen von Spielparametern

In diesem Schritt beschreiben Sie Ihr Spiel zum Hochladen in AWS .

- Spielname: Geben Sie einen aussagekräftigen Namen für Ihr Spielprojekt an. Dieser Name wird im Plugin verwendet.
- Flottenname: Geben Sie einen aussagekräftigen Namen für Ihre verwaltete EC2-Flotte an. Amazon GameLift verwendet diesen Namen (zusammen mit der Flotten-ID) beim Auflisten von Ressourcen in der AWS Konsole.
- Build-Name: Geben Sie einen aussagekräftigen Namen für Ihren Server-Build an. AWS verwendet diesen Namen, um auf die Kopie Ihres Server-Builds zu verweisen, die in Amazon hochgeladen GameLift und für Bereitstellungen verwendet wird.

- Startparameter: Geben Sie optionale Anweisungen ein, die beim Starten der ausführbaren Serverdatei auf einer verwalteten EC2-Flotten-Instance ausgeführt werden sollen. Die maximale Länge beträgt 1024 Zeichen.
- Spieleserver-Ordner: Geben Sie den Pfad zu einem lokalen Ordner an, der Ihren Server-Build enthält.
- Spieleserverdatei: Geben Sie den Namen der ausführbaren Serverdatei an.

Bereitstellen eines Szenarios

In diesem Schritt stellen Sie Ihr Spiel auf einer Cloud-Hosting-Lösung bereit, die auf dem von Ihnen ausgewählten Bereitstellungsszenario basiert. Dieser Vorgang kann bis zu 40 Minuten dauern, während Ihren Server-Build AWS validiert, Hosting-Ressourcen bereitstellt, Ihren Spieleserver installiert, Serverprozesse startet und sie für das Hosten von Spielsitzungen bereit macht.

Um mit der Bereitstellung zu beginnen, wählen Sie Bereitstellen aus CloudFormation. Sie können den Status Ihres Spiele-Hostings hier verfolgen. Für detailliertere Informationen können Sie sich bei der -AWSManagementkonsole für anmelden AWS und Ereignisbenachrichtigungen anzeigen. Stellen Sie sicher, dass Sie sich mit demselben Konto, demselben Benutzer und derselben AWS Region wie das aktive Benutzerprofil im Plugin anmelden.

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

Starten des Spieleclients

Wenn Ihre Flotte erfolgreich bereitgestellt wurde, werden jetzt Spieleserver ausgeführt und stehen zum Hosten von Spielsitzungen zur Verfügung. Sie können jetzt Ihren Client erstellen, starten, eine Verbindung herstellen, um der Spielsitzung beizutreten.

1. Konfigurieren Sie Ihren Spiele-Client. In diesem Schritt fordern Sie das Plugin auf, eine `GameLiftClientSettings` Komponente für Ihr Spielprojekt zu aktualisieren. Das Plugin verwendet diese Komponente, um bestimmte Informationen zu speichern, die Ihr Spielclient benötigt, um eine Verbindung zum Amazon- GameLift Service herzustellen.

    a. Wenn Sie das Beispielspiel nicht importiert und initialisiert haben, erstellen Sie eine neue `GameLiftClientSettings` Komponente. Wählen Sie im Hauptmenü Unity Editor die Option Assets, Create, GameLift, Client Settings aus. Wenn Sie mehrere Kopien von

GameLiftClientSettings in Ihrem Projekt erstellen, erkennt das Plugin dies automatisch und benachrichtigt Sie, welche Komponente das Plugin aktualisieren wird.

b. Wählen Sie in Launch Game die Option Configure Client: Apply Managed EC2 Settings aus. Diese Aktion aktualisiert Ihre Spielclient-Einstellungen, um die soeben bereitgestellte verwaltete EC2-Flotte zu verwenden.

2. Erstellen Sie Ihren Spiele-Client. Erstellen Sie eine ausführbare Client-Datei mit dem Standard-Unity-Build-Prozess. Wechseln Sie unter Datei, Build-Einstellungen die Plattform zu Windows, Mac, Linux. Wenn Sie das Beispielspiel importiert und die Einstellungen initialisiert haben, werden die Build-Liste und das Build-Ziel automatisch aktualisiert.
3. Starten Sie die ausführbare Datei für den neu erstellten Spieleclient. Um mit dem Spiel zu beginnen, starten Sie zwei bis vier Client-Instances und verwenden Sie die Benutzeroberfläche in jeder , um einer Spielsitzung beizutreten.

Wenn Sie den Beispiel-Spielclient verwenden, weist er die folgenden Merkmale auf:

- Eine Player-Anmeldekomponente. Wenn Sie eine Verbindung zu einem Spieleserver auf einer Anywhere-Flotte herstellen, gibt es keine Spielervalidierung. Sie können beliebige Werte eingeben, um der Spielsitzung beizutreten.
- Eine einfache Join-Spiel-Benutzeroberfläche. Wenn ein Client versucht, einem Spiel beizutreten, sucht er automatisch nach einer aktiven Spielsitzung mit einem verfügbaren Spielerslot. Wenn keine Spielsitzung verfügbar ist, fordert der Client eine neue Spielsitzung an. Wenn eine Spielsitzung verfügbar ist, fordert der Client auf, der verfügbaren Spielsitzung beizutreten. Wenn Sie Ihr Spiel mit mehreren gleichzeitigen Clients testen, startet der erste Client die Spielsitzung und die verbleibenden Clients treten automatisch der vorhandenen Spielsitzung bei.
- Spielsitzungen mit vier Spielerslots. Sie können bis zu vier Spieleclient-Instances gleichzeitig starten und sie treten derselben Spielesitzung bei.

Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 4.x

> **Note**
>
> Dieses Thema enthält Informationen zu einer früheren Version des Amazon- GameLift Plugins für Unity. Version 1.0.0 (veröffentlicht im Jahr 2021) verwendet das Amazon GameLift Server SDK 4.x oder früher. Eine Dokumentation zur neuesten Version des Plugins, das

Server-SDK 5.x verwendet und Amazon unterstützt GameLift Anywhere, finden Sie unter Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x.

Amazon GameLift bietet Tools zur Vorbereitung Ihrer Multiplayer-Spielserver für die Ausführung auf Amazon GameLift. Das Amazon GameLift -Plugin für Unity erleichtert die Integration von Amazon GameLift in Ihre Unity-Spielprojekte und die Bereitstellung von Amazon- GameLift Ressourcen für das Cloud-Hosting. Verwenden Sie das Plugin für Unity, um auf Amazon- GameLift APIs zuzugreifen und AWS CloudFormation Vorlagen für gängige Gaming-Szenarien bereitzustellen.

Nachdem Sie das Plugin eingerichtet haben, können Sie das Amazon- GameLift Unity-Beispiel auf ausprobieren GitHub.

Themen

Integrieren von Amazon GameLift in ein Unity-Spielserverprojekt

> **Note**
>
> Dieses Thema enthält Informationen zu einer früheren Version des Amazon- GameLift Plugins für Unity. Version 1.0.0 (veröffentlicht im Jahr 2021) verwendet das Amazon GameLift Server SDK 4.x oder früher. Eine Dokumentation zur neuesten Version des Plugins, das Server-SDK 5.x verwendet und Amazon unterstützt GameLift Anywhere, finden Sie unter Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x.

Dieses Thema hilft Ihnen dabei, Ihren benutzerdefinierten Spieleserver für das Hosting auf Amazon vorzubereiten GameLift. Der Spieleserver muss in der Lage sein, Amazon GameLift über seinen Status zu informieren, Spielsitzungen zu starten und zu beenden, wenn er dazu aufgefordert wird,

und andere Aufgaben auszuführen. Weitere Informationen finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Voraussetzungen

Führen Sie vor der Integration Ihres Spieleservers die folgenden Aufgaben aus:

- Einrichten einer IAM-Servicerolle für Amazon GameLift
- Installieren des Plugins für Unity

Einrichten eines neuen Serverprozesses

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Richten Sie die Kommunikation mit Amazon ein GameLift und melden Sie, dass der Serverprozess bereit ist, eine Spielsitzung zu hosten.

1. Initialisieren Sie das Server-SDK, indem Sie aufrufen`InitSDK()`.
2. Um den Server auf die Annahme einer Spielsitzung vorzubereiten, rufen Sie `ProcessReady()` mit den Details zum Verbindungsport und zum Spielsitzungsspeicherort auf. Geben Sie die Namen der Callback-Funktionen an, die der Amazon- GameLift Service aufruft, z. B`OnHealthCheck()`. `OnGameSession()`, `OnGameSessionUpdate()`, `OnProcessTerminate()`, . Es GameLift kann einige Minuten dauern, bis Amazon einen Rückruf bereitstellt.
3. Amazon GameLift aktualisiert den Status des Serverprozesses auf `ACTIVE`.
4. Amazon GameLift ruft `onHealthCheck` regelmäßig auf.

Das folgende Codebeispiel zeigt, wie Sie einen einfachen Serverprozess mit Amazon einrichten GameLift.

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

//processReady
```

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

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

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

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

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

Starten einer Spielsitzung

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Nachdem die Spielinitialisierung abgeschlossen ist, können Sie eine Spielsitzung starten.

1. Implementieren Sie die Callback-Funktion `onStartGameSession`. Amazon GameLift ruft diese Methode auf, um eine neue Spielsitzung auf dem Serverprozess zu starten und Spielerverbindungen zu erhalten.
2. Um eine Spielsitzung zu aktivieren, rufen Sie auf`ActivateGameSession()`. Weitere Informationen zum SDK finden Sie unter Referenz zum GameLift Amazon-Server-SDK (C#): Aktionen.

Das folgende Codebeispiel veranschaulicht, wie Sie eine Spielsitzung mit Amazon starten GameLift.

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

Beenden einer Spielsitzung

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Benachrichtigen Sie Amazon GameLift , wenn eine Spielsitzung endet. Es hat sich bewährt, Serverprozesse nach Abschluss der Spielsitzungen herunterzufahren, um Hosting-Ressourcen zu recyceln und zu aktualisieren.

1. Richten Sie eine Funktion mit dem Namen einon`ProcessTerminate`, um Anfragen von Amazon zu empfangen GameLift und aufzurufen`ProcessEnding()`.
2. Der Prozessstatus ändert sich in `TERMINATED`.

Im folgenden Beispiel wird beschrieben, wie Sie einen Prozess für eine Spielsitzung beenden.

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

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

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

Erstellen von Server-Builds und Hochladen in Amazon GameLift

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Nachdem Sie Ihren Spieleserver in Amazon integriert haben GameLift, laden Sie die Build-Dateien in eine Flotte hoch, damit Amazon sie für das Spiele-Hosting bereitstellen GameLift kann. Weitere Informationen zum Hochladen Ihres Servers in Amazon finden Sie GameLiftunter Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift.

Integrieren von Amazon GameLift in ein Unity-Spielclient-Projekt

> **Note**
>
> Dieses Thema enthält Informationen zu einer früheren Version des Amazon- GameLift Plugins für Unity. Version 1.0.0 (veröffentlicht im Jahr 2021) verwendet das Amazon GameLift Server SDK 4.x oder früher. Eine Dokumentation zur neuesten Version des Plugins, das Server-SDK 5.x verwendet und Amazon unterstützt GameLift Anywhere, finden Sie unter Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x.

Dieses Thema hilft Ihnen, einen Spieleclient einzurichten, um über einen Backend-Service eine Verbindung zu von Amazon GameLift gehosteten Spielsitzungen herzustellen. Verwenden Sie Amazon- GameLift APIs, um Matchmaking zu initiieren, die Platzierung von Spielsitzungen anzufordern und vieles mehr.

Fügen Sie dem Backend-Serviceprojekt Code hinzu, um die Kommunikation mit dem Amazon-GameLift Service zu ermöglichen. Ein Backend-Service übernimmt die gesamte Kommunikation des Spieleclients mit dem GameLift Service. Weitere Informationen zu Backend-Services finden Sie unter Gestalte deinen Spiele-Client-Dienst.

Ein Backend-Server übernimmt die folgenden Spieleclient-Aufgaben:

- Passen Sie die Authentifizierung für Ihre Spieler an.
- Fordern Sie Informationen zu aktiven Spielsitzungen vom Amazon- GameLift Service an.
- Erstellen Sie eine neue Spielsitzung.
- Fügen Sie einen Spieler zu einer vorhandenen Spielsitzung hinzu.
- Entfernen Sie einen Spieler aus einer vorhandenen Spielsitzung.

Themen

Voraussetzungen

Führen Sie die folgenden Aufgaben aus, bevor Sie die Kommunikation des Spieleservers mit dem Amazon- GameLift Client einrichten:

- Richten Sie ein AWS-Konto
- Installieren des Plugins für Unity
- Integrieren von Amazon GameLift in ein Unity-Spielserverprojekt
- GameLiftAmazon-Flotten einrichten

Initialisieren eines Spielclients

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Fügen Sie Code hinzu, um einen Spielclient zu initialisieren. Führen Sie diesen Code beim Start aus. Er ist für andere Amazon- GameLift Funktionen erforderlich.

1. Initialisieren Sie `AmazonGameLiftClient`. Rufen Sie entweder `AmazonGameLiftClient` mit einer Standard-Client-Konfiguration oder einer benutzerdefinierten Konfiguration auf. Weitere Informationen zum Konfigurieren eines Clients finden Sie unter Amazon GameLift auf einem Backend-Service einrichten.
2. Generieren Sie eine eindeutige Spieler-ID für jeden Spieler, um eine Verbindung zu einer Spielsitzung herzustellen. Weitere Informationen finden Sie unter Generieren Sie Spieler-IDs.

   Die folgenden Beispiele zeigen, wie Sie einen Amazon- GameLift Client einrichten.

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


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

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

        CredentialProfile profile = null;
```

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

    }
}
```

Erstellen einer Spielsitzung auf einer bestimmten Flotte

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Fügen Sie Code zum Starten neuer Spielsitzungen auf den von Ihnen bereitgestellten Flotten hinzu und stellen Sie diese Spielern zur Verfügung. Nachdem Amazon die neue Spielsitzung erstellt und ein zurückgegeben GameLift hat`GameSession`, können Sie Spieler hinzufügen.

- Stellen Sie eine Anfrage für eine neue Spielsitzung.
  - Wenn Ihr Spiel Flotten verwendet, rufen Sie `CreateGameSession()` mit einer Flotten- oder Alias-ID, einem Sitzungsnamen und der maximalen Anzahl gleichzeitiger Spieler für das Spiel auf.
  - Wenn Ihr Spiel Warteschlangen verwendet, rufen Sie auf`StartGameSessionPlacement()`.

Das folgende Beispiel zeigt, wie Sie eine Spielsitzung erstellen.

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

```
    cgsreq.MaximumPlayerSessionCount = 4;

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

Spieler zu Spielsitzungen hinzufügen

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Nachdem Amazon die neue Spielsitzung erstellt und ein `GameSession`Objekt zurückgegeben GameLift hat, können Sie Spieler hinzufügen.

1. Reservieren Sie einen Spielerslot in einer Spielsitzung, indem Sie eine neue Spielersitzung erstellen. Verwenden Sie `CreatePlayerSession` oder `CreatePlayerSessions` mit der Spielsitzungs-ID und einer eindeutigen ID für jeden Spieler.
2. Stellen Sie eine Verbindung mit der Spielsitzung her. Rufen Sie das `PlayerSession` Objekt ab, um die Verbindungsinformationen der Spielsitzung abzurufen. Sie können diese Informationen verwenden, um eine direkte Verbindung zum Serverprozess herzustellen:

    a. Verwenden Sie den angegebenen Port und entweder den DNS-Namen oder die IP-Adresse des Serverprozesses.

    b. Verwenden Sie den DNS-Namen und den Port Ihrer Flotten. Der DNS-Name und der Port sind erforderlich, wenn für Ihre Flotten die Generierung von TLS-Zertifikaten aktiviert ist.

    c. Verweisen Sie auf die Player-Sitzungs-ID. Die Spielersitzungs-ID ist erforderlich, wenn Ihr Spieleserver eingehende Spielerverbindungen validiert.

Die folgenden Beispiele zeigen, wie Sie einen Spieler-Spot in einer Spielsitzung reservieren.

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

Der folgende Code veranschaulicht, wie Sie einen Spieler mit der Spielsitzung verbinden.

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

Entfernen eines Spielers aus einer Spielsitzung

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Sie können die Spieler aus der Spielsitzung entfernen, wenn sie das Spiel verlassen.

1. Informieren Sie den Amazon- GameLift Service, dass ein Player die Verbindung zum Serverprozess getrennt hat. Rufen Sie `RemovePlayerSession` mit der Sitzungs-ID des Spielers auf.
2. Stellen Sie sicher, dass `RemovePlayerSession` zurückgibt`Success`. Anschließend GameLift ändert Amazon den Spieler-Slot so, dass er verfügbar ist, was Amazon einem neuen Spieler zuweisen GameLift kann.

Das folgende Beispiel veranschaulicht, wie Sie eine Player-Sitzung entfernen.

```
public void DisconnectPlayer(int playerIdx)
{
    //Receive the player session ID.
    string playerSessionId = playerSessions[playerIdx];
    var outcome = GameLiftServerAPI.RemovePlayerSession(playerSessionId);
    if (outcome.Success)
    {
        Debug.Log (":) PLAYER SESSION REMOVED");
    }
    else
    {
        Debug.Log(":(PLAYER SESSION REMOVE FAILED. RemovePlayerSession()
        returned " + outcome.Error.ToString());
    }
}
```

Installieren und Einrichten des Plugins

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

In diesem Abschnitt wird beschrieben, wie Sie das Amazon- GameLift Plugin für Unity, Version 1.0.0, herunterladen, installieren und einrichten.

Voraussetzungen

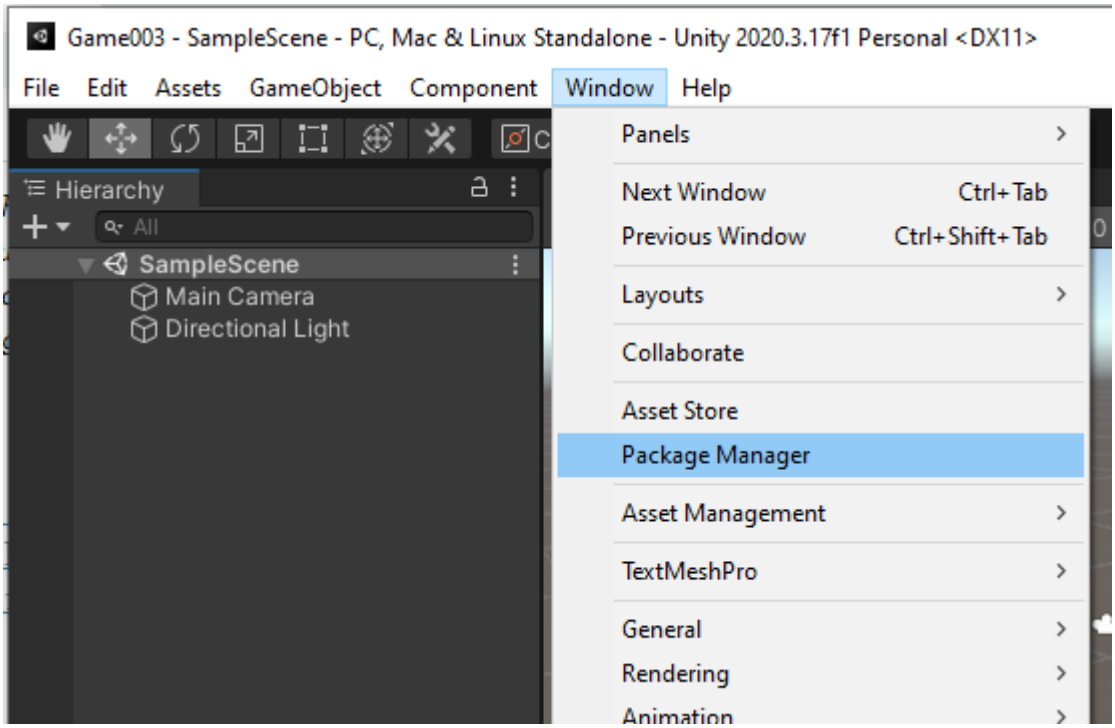
- Unity für Windows 2019.4 LTS, Windows 2020.3 LTS oder Unity für MacOS
- Aktuelle Version von Java
- Aktuelle Version von .NET 4.x

So laden Sie das Plugin für Unity herunter und installieren es

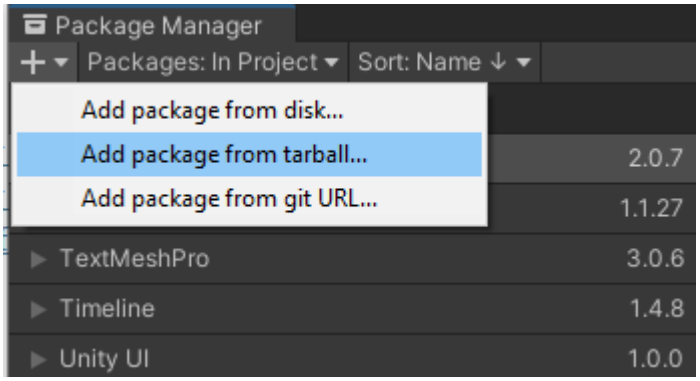
1. Laden Sie das Amazon- GameLift Plugin für Unity herunter. Die neueste Version finden Sie auf der Repository-Seite des Amazon GameLift -Plugins für Unity. Wählen Sie in der neuesten

Version Assets aus und laden Sie dann die `com.amazonaws.gamelift-version.tgz` Datei herunter.

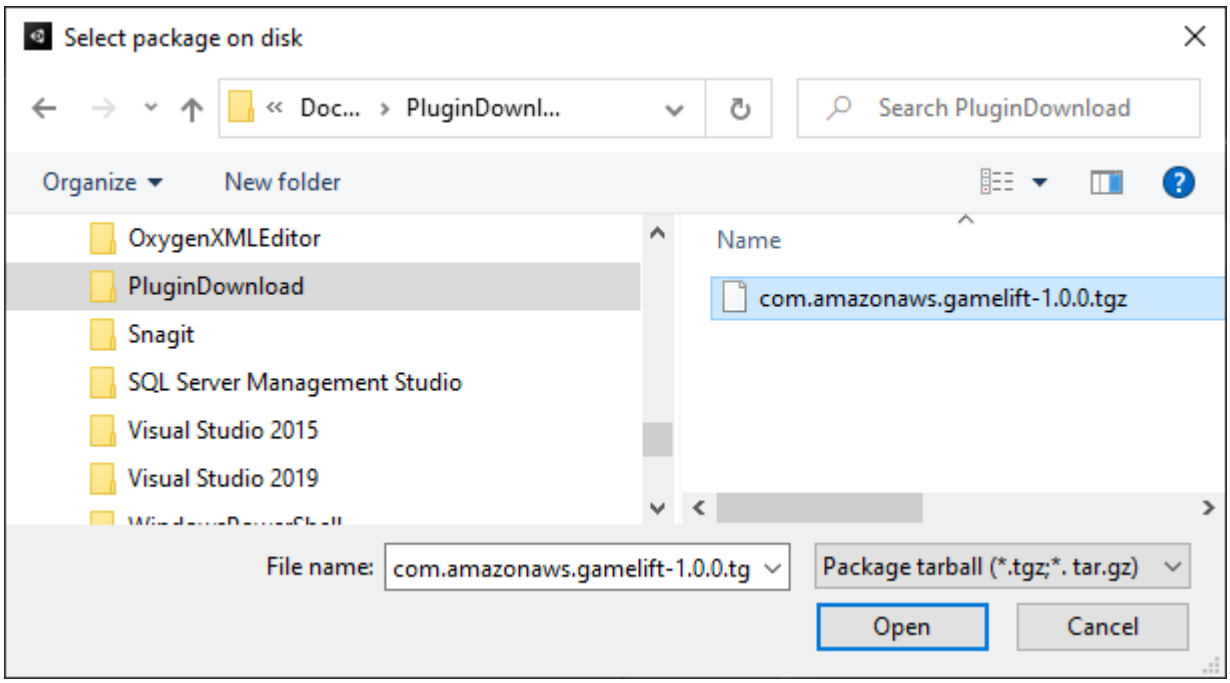
2. Starten Sie Unity und wählen Sie ein Projekt aus.
3. Wählen Sie in der oberen Navigationsleiste unter Fenster die Option Paketmanager aus:



4. Wählen Sie auf der Registerkarte Package Manager + und dann Paket aus Tarball hinzufügen...:



5. Navigieren Sie im Fenster Pakete auf Datenträger auswählen zum `com.amazonaws.gamelift` Ordner , wählen Sie die Datei `com.amazonaws.gamelift-version.tgz` und dann Öffnen aus:



6. Nachdem Unity das Plugin geladen hat, wird Amazon GameLift als neues Element im Menü Unity angezeigt. Die Installation und Neukompilierung von Skripten kann einige Minuten dauern. Die Registerkarte Amazon- GameLift Plugin-Einstellungen wird automatisch geöffnet.

7. Wählen Sie im Bereich SDK die Option .NET 4.x verwenden aus.

   Bei der Konfiguration ändert sich der Status von Nicht konfiguriert zu Konfiguriert.

Testen Sie Ihr Spiel lokal

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Verwenden Sie Amazon GameLift Local, um Amazon GameLift auf Ihrem lokalen Gerät auszuführen. Sie können Amazon GameLift Local verwenden, um Codeänderungen innerhalb von Sekunden ohne Netzwerkverbindung zu überprüfen.

Konfigurieren lokaler Tests

1. Wählen Sie im Fenster Plugin für Unity die Registerkarte Test aus.
2. Wählen Sie im Bereich Test die Option Amazon GameLift Local herunterladen aus. Das Plugin für Unity öffnet ein Browserfenster und lädt die `GameLift_06_03_2021.zip` Datei in Ihren Download-Ordner herunter.

   Der Download enthält das C# Server SDK, .NET-Quelldateien und eine mit Unity kompatible .NET-Komponente.
3. Entpacken Sie die heruntergeladene `GameLift_06_03_2021.zip`-Datei.
4. Wählen Sie im Fenster Amazon- GameLift Plugin-Einstellungen die Option Amazon GameLift Local Path aus, navigieren Sie zum entpackten Ordner, wählen Sie die Datei aus `GameLiftLocal.jar`und wählen Sie dann Öffnen aus.

   Bei der Konfiguration ändert sich der lokale Teststatus von Nicht konfiguriert zu Konfiguriert.
5. Überprüfen Sie den Status der JRE. Wenn der Status Nicht konfiguriert lautet, wählen Sie JRE herunterladen und installieren Sie die empfohlene Java-Version.

   Nachdem Sie die Java-Umgebung installiert und konfiguriert haben, ändert sich der Status in Konfiguriert.

Ausführen Ihres lokalen Spiels

1. Wählen Sie auf der Registerkarte Plugin für Unity die Registerkarte Test aus.
2. Wählen Sie im Bereich Test die Option Lokale Testbenutzeroberfläche öffnen aus.

3. Geben Sie im Fenster Local Testing einen ausführbaren Server-Pfad an. Wählen Sie ... aus, um den Pfad und den ausführbaren Namen Ihrer Serveranwendung auszuwählen.
4. Geben Sie im Fenster Lokale Tests einen Bol Lokaler Port an.
5. Wählen Sie Bereitstellen und Ausführen aus, um den Server bereitzustellen und auszuführen.
6. Um Ihren Spieleserver anzuhalten, wählen Sie Anhalten oder Schließen der Spieleserverfenster.

Bereitstellen eines Szenarios

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Ein Szenario verwendet eine -AWS CloudFormationVorlage, um die Ressourcen zu erstellen, die Sie für die Bereitstellung einer Cloud-Hosting-Lösung für Ihr Spiel benötigen. In diesem Abschnitt werden die von Amazon GameLift bereitgestellten Szenarien und deren Verwendung beschrieben.

Voraussetzungen

Um das Szenario bereitzustellen, benötigen Sie eine IAM-Rolle für den Amazon- GameLift Service. Informationen zum Erstellen einer Rolle für Amazon finden Sie GameLiftunter Richten Sie ein AWS-Konto.

Jedes Szenario erfordert Berechtigungen für die folgenden Ressourcen:

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

Szenarien

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Das Amazon- GameLift Plug-in für Unity umfasst die folgenden Szenarien:

Nur Authentifizierung

In diesem Szenario wird ein Spiele-Backend-Service erstellt, der die Spielerauthentifizierung ohne Spielserver-Fähigkeit durchführt. Die Vorlage erstellt die folgenden Ressourcen in Ihrem Konto:

- Ein Amazon Cognito-Benutzerpool zum Speichern von Player-Authentifizierungsinformationen.
- Ein Amazon API Gateway REST-Endpunkt-gestützter Handler, AWS Lambda der Spiele startet und Spielverbindungsinformationen anzeigt.

Flotte für eine einzelne Region

In diesem Szenario wird ein Spiele-Backend-Service mit einer einzigen Amazon- GameLift Flotte erstellt. Es erstellt die folgenden Ressourcen:

- Ein Amazon Cognito-Benutzerpool für einen Spieler zur Authentifizierung und zum Starten eines Spiels.
- Ein AWS Lambda Handler, um nach einer vorhandenen Spielsitzung mit einem offenen Spieler-Slot auf der Flotte zu suchen. Wenn ein offener Slot nicht gefunden werden kann, wird eine neue Spielsitzung erstellt.

Multiregionale Flotte mit einer Warteschlange und einem benutzerdefinierten Matchmaker

Dieses Szenario bildet Übereinstimmungen mithilfe von Amazon- GameLift Warteschlangen und einem benutzerdefinierten Matchmaker, um die ältesten Spieler im wartenden Pool zu gruppieren. Es erstellt die folgenden Ressourcen:

- Ein Amazon Simple Notification Service-Thema, in dem Amazon Nachrichten GameLift veröffentlicht. Weitere Informationen zu SNS-Themen und -Benachrichtigungen finden Sie unter Richten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein.

- Eine Lambda-Funktion, die von der Nachricht aufgerufen wird, die Platzierungs- und Spielverbindungsdetails angibt.
- Eine Amazon-DynamoDB-Tabelle zum Speichern von Platzierungs- und Spielverbindungsdetails. `GetGameConnection` ruft aus dieser Tabelle gelesen auf und gibt die Verbindungsinformationen an den Spielclient zurück.

Spot-Flotten mit einer Warteschlange und einem benutzerdefinierten Matchmaker

Dieses Szenario bildet Übereinstimmungen mithilfe von Amazon- GameLift Warteschlangen und einem benutzerdefinierten Matchmaker und konfiguriert drei Flotten. Es erstellt die folgenden Ressourcen:

- Zwei Spot-Flotten, die verschiedene Instance-Typen enthalten, um eine Haltbarkeit für die Nichtverfügbarkeit von Spot zu gewährleisten.
- Eine On-Demand-Flotte, die als Backup für die anderen Spot-Flotten fungiert. Weitere Informationen zum Entwerfen Ihrer Flotten finden Sie unter Leitfaden zur GameLift Amazon-Flottenplanung.
- Eine Amazon- GameLift Warteschlange, um die Serververfügbarkeit hoch und kostengünstig zu halten. Weitere Informationen und bewährte Methoden zu Warteschlangen finden Sie unter Entwerfen Sie eine Warteschlange für Spielsitzungen.

FlexMatch

In diesem Szenario wird FlexMatch, ein verwalteter Matchmaking-Service, verwendet, um Spieler zusammenzubringen. Weitere Informationen zu FlexMatchfinden Sie unter Was ist Amazon GameLift FlexMatch. In diesem Szenario werden die folgenden Ressourcen erstellt:

- Eine Lambda-Funktion zum Erstellen eines Matchmaking-Tickets nach Erhalt von `StartGame` Anfragen.
- Eine separate Lambda-Funktion zum Abhören von FlexMatch Übereinstimmungsereignissen.

Um unnötige Gebühren für Ihr zu vermeidenAWS-Konto, entfernen Sie die Ressourcen, die von jedem Szenario erstellt wurden, nachdem Sie sie nicht mehr verwendet haben. Löschen Sie den entsprechenden AWS CloudFormationStack.

Aktualisieren von AWS Anmeldeinformationen

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Das Amazon GameLift -Plugin für Unity benötigt Sicherheitsanmeldeinformationen, um ein Szenario bereitzustellen. Sie können entweder neue Anmeldeinformationen erstellen oder vorhandene Anmeldeinformationen verwenden.

Weitere Informationen zum Konfigurieren von Anmeldeinformationen finden Sie unter Verstehen und Abrufen Ihrer AWS Anmeldeinformationen.

So aktualisieren Sie AWS Anmeldeinformationen

1. Wählen Sie unter Unity im Plugin für Unity die Registerkarte Deploy aus.
2. Wählen Sie im Bereich Bereitstellen AWS die Option Anmeldeinformationen aus.
3. Sie können neue AWS Anmeldeinformationen erstellen oder vorhandene Anmeldeinformationen auswählen.
   - Um Anmeldeinformationen zu erstellen, wählen Sie Neues Anmeldeinformationsprofil erstellen und geben Sie dann den neuen Profilnamen , die AWS Zugriffsschlüssel-ID , den AWS geheimen Schlüssel und anAWS-Region.
   - Um vorhandene Anmeldeinformationen auszuwählen, wählen Sie Profil für vorhandene Anmeldeinformationen auswählen und wählen Sie dann einen Profilnamen und ausAWS-Region.
4. Wählen Sie im Fenster AWS Anmeldeinformationen aktualisieren die Option Anmeldeinformationsprofil aktualisieren aus.

Konto-Bootstrap aktualisieren

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Der Bootstrap-Speicherort ist ein Amazon S3-Bucket, der während der Bereitstellung verwendet wird. Es wird verwendet, um Spieleserver-Assets und andere Abhängigkeiten zu speichern. Das , das AWS-Region Sie für den Bucket auswählen, muss dieselbe Region sein, die Sie für die Szenariobereitstellung verwenden werden.

Weitere Informationen zu Amazon S3-Buckets finden Sie unter Erstellen, Konfigurieren und Arbeiten mit Amazon-Simple-Storage-Service-Buckets.

So aktualisieren Sie den Konto-Bootstrap-Speicherort

1. Wählen Sie unter Unity im Plugin für Unity die Registerkarte Deploy aus.
2. Wählen Sie im Bereich Bereitstellen die Option Konto-Bootstrap aktualisieren aus.
3. Wählen Sie im Fenster Konto-Bootstrapping einen vorhandenen Amazon S3-Bucket aus oder erstellen Sie einen neuen Amazon S3-Bucket:
   - Um einen vorhandenen Bucket auszuwählen, wählen Sie Vorhandenen Amazon S3-Bucket auswählen und Aktualisieren, um Ihre Auswahl zu speichern.
   - Wählen Sie Neuen Amazon S3-Bucket erstellen, um einen neuen Amazon-Simple-Storage-Service-Bucket zu erstellen, und wählen Sie dann eine Richtlinie aus. Die Richtlinie gibt an, wann der Amazon S3-Bucket abläuft. Wählen Sie Erstellen, um den Bucket zu erstellen.

Bereitstellen eines Spielszenarios

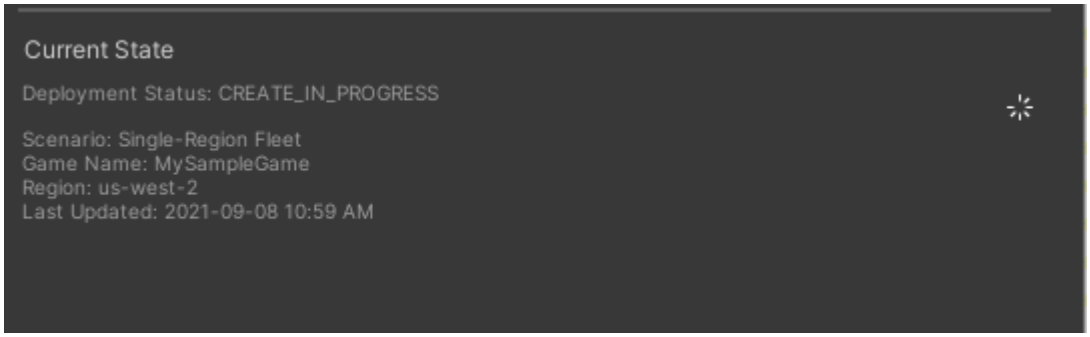
> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Sie können ein Szenario verwenden, um Ihr Spiel mit Amazon zu testen GameLift. Jedes Szenario verwendet eine AWS CloudFormation Vorlage, um einen Stack mit den erforderlichen Ressourcen zu erstellen. Die meisten Szenarien erfordern eine ausführbare Datei und einen Build-Pfad für den Spieleserver. Wenn Sie das Szenario bereitstellen, kopiert Amazon GameLift im Rahmen der Bereitstellung Spielressourcen an den Bootstrap-Standort.

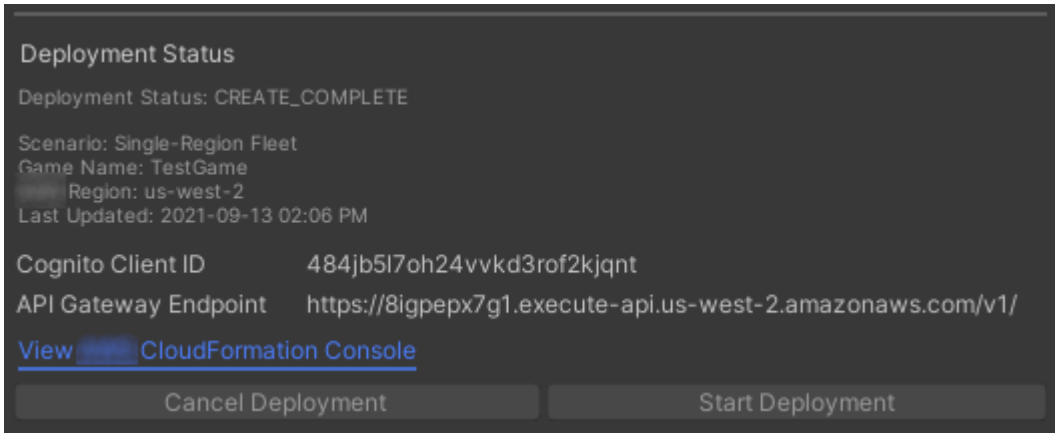
Sie müssen AWS Anmeldeinformationen und einen AWS Konto-Bootstrap konfigurieren, um ein Szenario bereitzustellen.

So stellen Sie ein Szenario bereit

1. Wählen Sie unter Unity im Plugin für Unity die Registerkarte Deploy aus.
2. Wählen Sie im Bereich Bereitstellen die Option Bereitstellungsbenutzeroberfläche öffnen aus.
3. Wählen Sie im Fenster Bereitstellung ein Szenario aus.
4. Geben Sie einen Spielenamen ein. Dieser Wert muss eindeutig sein. Der Spielename ist Teil des AWS CloudFormationStack-Namens, wenn Sie das Szenario bereitstellen.
5. Wählen Sie den Pfad für den Spieleserver-Build-Ordner aus. Der Pfad des Build-Ordners verweist auf den Ordner, der die ausführbare Serverdatei und Abhängigkeiten enthält.
6. Wählen Sie den Dateipfad Game Server Build .exe aus. Der Pfad der ausführbaren Build-Datei verweist auf die ausführbare Spieleserver-Datei.
7. Wählen Sie Bereitstellung starten, um mit der Bereitstellung eines Szenarios zu beginnen. Sie können den Status der Aktualisierung im Bereitstellungsfenster unter Aktueller Status verfolgen. Die Bereitstellung von Scenarios kann bis zu 30 Minuten dauern.



8. Wenn das Szenario die Bereitstellung abgeschlossen hat, wird der aktuelle Status aktualisiert und enthält die Cognito-Client-ID und den API-Gateway-Endpunkt, die Sie kopieren und in das Spiel einfügen können.



9. Um die Spieleinstellungen zu aktualisieren, wählen Sie im Menü Unity die Option Zu Client-Verbindungseinstellungen gehen aus. Dadurch wird auf der rechten Seite des Bildschirms Unity eine Registerkarte Inspector angezeigt.
10. Deaktivieren Sie den lokalen Testmodus .
11. Geben Sie den API Gateway-Endpunkt und die Coginito-Client-ID ein. Wählen Sie dieselbe aus, die AWS-Region Sie für die Szenariobereitstellung verwendet haben. Anschließend können Sie den Spielclient mithilfe der bereitgestellten Szenarioressourcen neu erstellen und ausführen.

Löschen von Ressourcen, die durch das Szenario erstellt wurden

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Um die für das Szenario erstellten Ressourcen zu löschen, löschen Sie den entsprechenden AWS CloudFormationStack.

So löschen Sie Ressourcen, die durch das Szenario erstellt wurden

1. Wählen Sie im Fenster Amazon GameLift -Plugin für Unity Deployment die Option View AWS CloudFormation Console aus, um die AWS CloudFormation Konsole zu öffnen.
2. Wählen Sie in der -AWS CloudFormationKonsole Stacks und dann den Stack aus, der den bei der Bereitstellung angegebenen Spielenamen enthält.
3. Wählen Sie Löschen, um den Stack zu löschen. Das Löschen eines Stacks kann einige Minuten dauern. Nachdem den vom Szenario verwendeten Stack AWS CloudFormation gelöscht hat, ändert sich sein Status in `ROLLBACK_COMPLETE`.

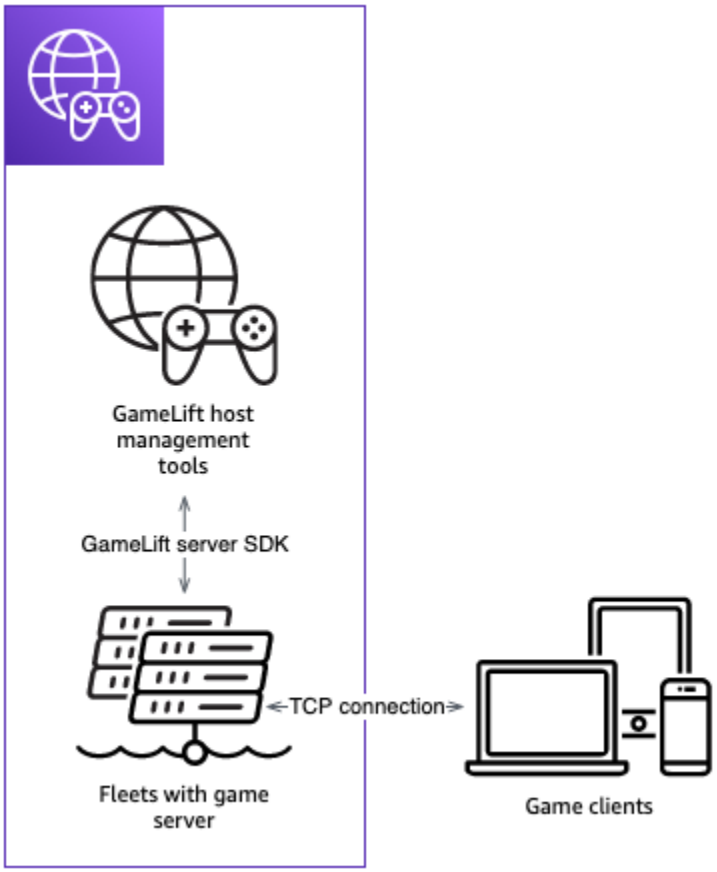
Integrieren von Spielen mit Amazon GameLift in Unity

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Integrieren Sie Ihr Unity-Spiel in Amazon, GameLift indem Sie die folgenden Aufgaben ausführen:

- Integrieren von Amazon GameLift in ein Unity-Spielserverprojekt
- Integrieren von Amazon GameLift in ein Unity-Spielclient-Projekt

Das folgende Diagramm zeigt einen Beispielablauf für die Integration eines Spiels. Im Diagramm wird eine Flotte mit dem Spieleserver auf Amazon bereitgestellt GameLift. Der Spieleclient kommuniziert mit dem Spieleserver, der mit Amazon kommuniziert GameLift.



Importieren und Ausführen eines Beispielspiels

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Das Amazon GameLift -Plugin für Unity enthält ein Beispielspiel, mit dem Sie die Grundlagen der Integration Ihres Spiels in Amazon erkunden können GameLift. In diesem Abschnitt erstellen Sie den Spieleclient und den Spieleserver und testen dann lokal mit Amazon GameLift Local.

Voraussetzungen

- Richten Sie ein AWS-Konto
- Installieren und Einrichten des Plugins

Erstellen und Ausführen des Beispiel-Spielservers

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Richten Sie die Spieleserverdateien des Beispielspiels ein.

1. Wählen Sie unter Unity im Menü Amazon GameLift und dann Beispielspiel importieren aus.
2. Wählen Sie im Fenster Beispielspiel importieren die Option Import ierenaus, um das Spiel, seine Komponenten und Abhängigkeiten zu importieren.
3. Erstellen Sie den Spieleserver. Wählen Sie im Menü Unity die Option Amazon GameLift und anschließend Windows-Beispielserver-Build-Einstellungen anwenden oder MacOS-Beispielserver-Build-Einstellungen anwenden aus. Nachdem Sie die Spielservereinstellungen konfiguriert haben, kompiliert Unity die Komponenten neu.
4. Wählen Sie im Menü Unity die Option File und anschließend Build aus. Wählen Sie Server Build , Build und dann einen Build-Ordner speziell für Serverdateien aus.

   Unity erstellt den Beispiel-Spielserver und platziert die ausführbare Datei und die erforderlichen Komponenten im angegebenen Build-Ordner.

Erstellen und Ausführen des Beispiel-Spiel-Clients

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Richten Sie die Spielclientdateien des Beispielspiels ein.

1. Wählen Sie unter Unity im Menü Amazon GameLift und anschließend Windows-Beispiel-Client-Build-Einstellungen anwenden oder MacOS-Beispiel-Client-Build-Einstellungen anwenden aus. Nachdem die Spielclient-Einstellungen konfiguriert wurden, kompiliert Unity die Komponenten neu.
2. Wählen Sie unter Unity im Menü Zu Client-Einstellungen gehen aus. Dadurch wird auf der rechten Seite des Bildschirms Unity eine Registerkarte Inspector angezeigt. Wählen Sie auf der Registerkarte Amazon GameLift -Client-Einstellungen die Option Lokaler Testmodus aus.
3. Erstellen Sie den Spiele-Client. Wählen Sie im Menü Unity die Option File aus. Vergewissern Sie sich, dass Server Build nicht aktiviert ist, wählen Sie Erstellen und wählen Sie dann einen Build-Ordner speziell für Clientdateien aus.

   Unity erstellt den Beispielspielclient und platziert die ausführbare Datei und die erforderlichen Komponenten im angegebenen Client-Build-Ordner.
4. Sie haben den Spieleserver und den Client nicht erstellt. In den nächsten Schritten führen Sie das Spiel aus und sehen, wie es mit Amazon interagiert GameLift.

Testen Sie das Beispielspiel lokal

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Führen Sie das Beispielspiel aus, das Sie mit Amazon GameLift Local importiert haben.

1. Starten Sie den Spieleserver. Wählen Sie unter Unity im Plugin für Unity die Registerkarte Deploy aus.

2. Wählen Sie im Bereich Test die Option Lokale Testbenutzeroberfläche öffnen aus.
3. Geben Sie im Fenster Lokale Tests einen .exe-Dateipfad für Game Server an. Der Pfad muss den ausführbaren Namen enthalten. Beispiel: `C:/MyGame/GameServer/MyGameServer.exe`
4. Wählen Sie Bereitstellen und Ausführen aus. Das Plugin für Unity startet den Spieleserver und öffnet ein Amazon GameLift Local-Protokollfenster. Die Fenster enthalten Protokollnachrichten, einschließlich Nachrichten, die zwischen dem Spieleserver und Amazon GameLift Local gesendet wurden.
5. Starten Sie den Spielclient. Suchen Sie den Build-Speicherort mit dem Beispiel-Spiel-Client und wählen Sie die ausführbare Datei aus.
6. Geben Sie im Amazon- GameLift Beispielspiel eine E-Mail und ein Passwort ein und wählen Sie dann Anmelden aus. Die E-Mail-Adresse und das Passwort werden nicht validiert oder verwendet.
7. Wählen Sie im Amazon- GameLift Beispielspiel die Option Starten aus. Der Spielclient sucht nach einer Spielsitzung. Wenn eine Sitzung nicht gefunden werden kann, wird eine erstellt. Der Spielclient startet dann die Spielsitzung. Sie können Spielaktivitäten in den Protokollen sehen.

Beispiele für Spieleserver-Protokolle

```
...
2021-09-15T19:55:3495 PID:20728 Log :) GAMELIFT AWAKE
2021-09-15T19:55:3512 PID:20728 Log :) I AM SERVER
2021-09-15T19:55:3514 PID:20728 Log :) GAMELIFT StartServer at port 33430.
2021-09-15T19:55:3514 PID:20728 Log :) SDK VERSION: 4.0.2
2021-09-15T19:55:3556 PID:20728 Log :) SERVER IS IN A GAMELIFT FLEET
2021-09-15T19:55:3577 PID:20728 Log :) PROCESSREADY SUCCESS.
2021-09-15T19:55:3577 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
...
2021-09-15T19:55:3634 PID:20728 Log :) GAMELOGIC AWAKE
2021-09-15T19:55:3635 PID:20728 Log :) GAMELOGIC START
2021-09-15T19:55:3636 PID:20728 Log :) LISTENING ON PORT 33430
2021-09-15T19:55:3636 PID:20728 Log SERVER: Frame: 0 HELLO WORLD!
...
2021-09-15T19:56:2464 PID:20728 Log :) GAMELIFT SESSION REQUESTED
2021-09-15T19:56:2468 PID:20728 Log :) GAME SESSION ACTIVATED
2021-09-15T19:56:3578 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:57:3584 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:58:0334 PID:20728 Log SERVER: Frame: 8695 Connection accepted: playerIdx
 0 joined
```

```
2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerIdx
 1 joined
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerIdx 0 Msg:
 CONNECT: server IP localhost
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 0:CONNECT:
 server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 0
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerIdx 1 Msg:
 CONNECT: server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 1:CONNECT:
 server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 1
```

Beispiele GameLift für lokale Amazon-Protokolle

```
12:55:26,000  INFO || - [SocketIOServer] main - Session store / pubsub factory used:
 MemoryStoreFactory (local session store only)
12:55:28,092  WARN || - [ServerBootstrap] main - Unknown channel option 'SO_LINGER' for
 channel '[id: 0xe23d0a14]'
12:55:28,101  INFO || - [SocketIOServer] nioEventLoopGroup-2-1 - SocketIO server
 started at port: 5757
12:55:28,101  INFO || - [SDKConnection] main - GameLift SDK server (communicates with
 your game server) has started on http://localhost:5757
12:55:28,120  INFO || - [SdkWebSocketServer] WebSocketSelector-20 - WebSocket Server
 started on address localhost/127.0.0.1:5759
12:55:28,166  INFO || - [StandAloneServer] main - GameLift Client server (listens for
 GameLift client APIs) has started on http://localhost:8080
12:55:28,179  INFO || - [StandAloneServer] main - GameLift server sdk http listener has
 started on http://localhost:5758
12:55:35,453  INFO || - [SdkWebSocketServer] WebSocketWorker-12 - onOpen
 socket: /?pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp and handshake /?
pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp
12:55:35,551  INFO || - [HostProcessManager] WebSocketWorker-12 - client connected with
 pID 20728
12:55:35,718  INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
 GameLift API to use: ProcessReady for pId 20728
12:55:35,718  INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
 Received API call for processReady from 20728
12:55:35,738  INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
 onProcessReady data: port: 33430
 12:55:35,739  INFO || - [HostProcessManager] GameLiftSdkHttpHandler-thread-0 -
 Registered new process with pId 20728
```

```
12:55:35,789  INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
 GameLift API to use: ReportHealth for pId 20728
12:55:35,790  INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
 Received API call for ReportHealth from 20728
12:55:35,794  INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
 ReportHealth data: healthStatus: true
 12:56:24,098  INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
 GameLift.DescribeGameSessions
12:56:24,119  INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
 to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,241  INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
 GameLift.CreateGameSession
12:56:24,242  INFO || - [CreateGameSessionDispatcher] Thread-12 - Received API call to
 create game session with input: {"FleetId":"fleet-123","MaximumPlayerSessionCount":4}
12:56:24,265  INFO || - [HostProcessManager] Thread-12 - Reserved process:
 20728 for gameSession: arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d
12:56:24,266  INFO || - [WebSocketInvoker] Thread-12 - StartGameSessionRequest:
 gameSessionId=arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d, fleetId=fleet-123, gameSessionName=null,
 maxPlayers=4, properties=[], ipAddress=127.0.0.1, port=33430, gameSessionData?=false,
 matchmakerData?=false, dnsName=localhost
12:56:24,564  INFO || - [CreateGameSessionDispatcher] Thread-12 - GameSession with
 id: arn:aws:gamelift:local::gamesession/fleet-123/gsess-59f6cc44-4361-42f5-95b5-
fdb5825c0f3d created
12:56:24,585  INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
 GameLift.DescribeGameSessions
12:56:24,585  INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
 to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,660  INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
 GameLift API to use: GameSessionActivate for pId 20728
12:56:24,661  INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0 -
 Received API call for GameSessionActivate from 20728
12:56:24,678  INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0
 - GameSessionActivate data: gameSessionId: "arn:aws:gamelift:local::gamesession/
fleet-123/gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d"
```

Herunterfahren des Serverprozesses

> **Note**
>
> Dieses Thema bezieht sich auf das Amazon GameLift -Plugin für Unity Version 1.0.0, das Server-SDK 4.x oder früher verwendet.

Nachdem Sie mit Ihrem Beispielspiel fertig sind, fahren Sie den Server in Unity herunter.

1. Wählen Sie im Spielclient Beenden oder schließen Sie das Fenster, um den Spielclient zu beenden.
2. Wählen Sie unter Unity im Fenster Local Testing die Option Stop aus oder schließen Sie die Spielserverfenster, um den Server zu stoppen.

Integrieren von Spielen mit dem Amazon GameLift -Plugin für Unreal Engine

Die Themen in diesem Abschnitt beschreiben das Amazon- GameLift Plugin für Unreal Engine (UE) und wie Sie es verwenden können, um Ihr Multiplayer-Spielprojekt für das Hosting mit Amazon vorzubereiten GameLift. Arbeiten Sie vollständig in Ihrer UE-Entwicklungsumgebung mit den geführten Workflows des Plugins, um die grundlegenden Anforderungen für das Hosting mit Amazon zu erfüllen GameLift.

Amazon GameLift ist ein vollständig verwalteter Service, mit dem Spieleentwickler dedizierte Spieleserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können. Weitere Informationen zum Amazon GameLift -Hosting finden Sie unter So GameLift funktioniert Amazon.

Themen

Informationen zum Plugin

Das Plugin fügt dem UE-Editor Amazon- GameLift Tools und -Funktionen hinzu. Die geführten Workflows des Plugins zur Integration von Amazon GameLift in Ihr Spielprojekt, zur Benennung einer Workstation als lokalen Host zum Testen und zur Bereitstellung des Spieleservers im Amazon GameLift -Cloud-Hosting.

Verwenden Sie die vordefinierten Hosting-Lösungen des Plugins, um Ihr Spiel bereitzustellen. Richten Sie eine Amazon GameLift Anywhere-Flotte mit Ihrer lokalen Workstation als Host ein. Wählen Sie für das Cloud-Hosting aus zwei gängigen Bereitstellungsszenarien, die die Spielerlatenz, die Verfügbarkeit von Spielsitzungen und die Kosten auf unterschiedliche Weise ausgleichen. Ein Szenario umfasst einen einfachen FlexMatch Matchmaker und einen Regelsatz. Verwenden Sie diese Lösungen, um schnell mit einer produktionsbereiten Hosting-Struktur zu beginnen, und optimieren und nach Bedarf anpassen.

Das Plugin enthält die folgenden Komponenten:

- Plugin-Module für den UE-Editor. Wenn das Plugin installiert ist, erhalten Sie über eine neue Hauptmenü-Schaltfläche Zugriff auf die Amazon- GameLift Funktionalität.
- C++-Bibliotheken für die Amazon- GameLift Service-API mit clientseitiger Funktionalität.
- Unecht Bibliotheken für das Amazon GameLift Server SDK (Version 5).
- Inhalt zum Testen, einschließlich einer Startup-Spielkarte und zwei Testkarten mit grundlegenden Vorlagen und UI-Elementen zur Verwendung beim Testen einer Serverintegration.
- Bearbeitbare Konfigurationen in Form von AWS CloudFormation Vorlagen, die das Plugin bei der Bereitstellung Ihres Spieleservers für das Hosting verwendet.

Plugin-Workflow

In den folgenden Schritten wird ein typischer Ansatz für die Integration und Bereitstellung eines Spielprojekts mit dem Amazon GameLift -Plugin für Unreal Engine beschrieben. Sie führen diese Schritte aus, indem Sie im UE-Editor und in Ihrem Spielcode arbeiten.

1. Erstellen Sie ein Benutzerprofil, das mit Ihrem -AWSKonto verknüpft ist und Anmeldeinformationen für gültige -Kontobenutzer mit Berechtigungen zur Verwendung von Amazon bereitstellt GameLift.
2. Fügen Sie Ihrem Spieleprojekt Servercode hinzu, um die Kommunikation zwischen einem laufenden Spieleserver und dem mit dem Amazon- GameLift Service herzustellen.

3. Fügen Sie Ihrem Spielprojekt Client-Code hinzu, mit dem Spielclients Anfragen an Amazon senden können, GameLift um neue Spielsitzungen zu starten und dann eine Verbindung zu ihnen herzustellen.
4. Verwenden Sie den Anywhere-Workflow, um Ihre lokale Workstation als Anywhere-Host für Ihren Spieleserver einzurichten. Starten Sie Ihren Spieleserver und Ihren Client lokal über das Plugin, stellen Sie eine Verbindung zu einer Spielsitzung her und testen Sie Ihre Integration.
5. Verwenden Sie den EC2-Hosting-Workflow, um Ihren integrierten Spieleserver hochzuladen und eine Cloud-Hosting-Lösung bereitzustellen. Wenn Ihr Spieleserver bereit ist, starten Sie Ihren Spieleclient lokal über das Plugin, stellen Sie eine Verbindung zu einer Spielsitzung her und spielen Sie das Spiel.

Wenn Sie im Plugin arbeiten, erstellen und verwenden Sie -AWSRessourcen. Für diese Aktionen können Gebühren für das verwendete AWS Konto anfallen. Wenn Sie noch nicht mit vertraut sindAWS, sind diese Aktionen möglicherweise durch das AWS kostenlose Kontingent für abgedeckt.

Installieren des Plugins für Unreal

In diesem Abschnitt werden die ersten Installationsaufgaben beschrieben, um das Plugin zu einem Unreal-Engine-Projekt hinzuzufügen. Die Plugin-Funktionalität ist verfügbar, wenn Sie das Projekt im Unreal-Editor geöffnet haben.

> **Note**
>
> Sie können das Amazon GameLift -Plugin mit einer Standardversion des UE-Editors verwenden, aber Sie müssen eine quellgefertigte Version verwenden, wenn Sie Ihren Spieleserver-Build verpacken.

Bevor Sie beginnen

So verwenden Sie das Amazon- GameLift Plugin für Unreal Engine:

- Amazon GameLift -Plugin für das Unreal Engine-Release-Paket. [Website herunterladen]
- Microsoft Visual Studio 2019 oder höher.
- Eine quellgenerierte Version des Unreal Engine-Editors. Sie benötigen eine quellgefertigte Version, um die Serverkomponenten für ein Multiplayer-Spiel zu verpacken. Weitere Informationen, einschließlich zusätzlicher Voraussetzungen, finden Sie in der Unreal Engine-Dokumentation:

  - Zugriff auf den Quellcode der Unreal Engine auf GitHub Sie benötigen GitHub - und Epic-Games-Konten.
  - Tutorial „Erstellen einer unrealen Engine aus der Quelle“.
- Ein Multiplayer-Spieleprojekt mit C++-Spielecode. Wenn Sie mit einem Blueprint-Projekt arbeiten, finden Sie in der Dokumentation Unreal dazu, wie Sie C++-Quellcode für Ihr Projekt generieren.

Fügen Sie das Plugin zu Ihrem Spielprojekt hinzu

Führen Sie die folgenden Aufgaben aus, um das Plugin zu Ihrem Spielprojekt hinzuzufügen.

Erstellen des Amazon GameLift -C++-Server-SDK

1. Entpacken Sie das Amazon GameLift -Plugin für das Unreal Engine-Release-Paket, um zwei ZIP-Dateien zu extrahieren:

   - `amazon-gamelift-plugin-unreal-<>-sdk-<>.zip`
   - `GameLift-Cpp-ServerSDK-<>.zip.`

   Entpacken Sie diese Dateien.
2. Öffnen Sie den `GameLift-Cpp-ServerSDK-<>` Ordner und führen Sie dann die folgenden Anweisungen für Ihre Plattform aus: Linux oder Microsoft Windows.

Linux

1. Führen Sie die folgenden Befehle aus:

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

   Mit diesen Befehlen wird die `/lib/aws-cpp-sdk-gamelift-server.so` Datei erstellt.
2. Kopieren Sie `/lib/aws-cpp-sdk-gamelift-server.so` in das `amazon-gamelift-plugin-unreal/GameLiftPlugin/Source/GameliftServer/ThirdParty/GameLiftServerSDK/Linux/x86_64-unknown-linux-gnu/` Verzeichnis .

Microsoft Windows

1. Führen Sie die folgenden Befehle aus:

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

   Diese Befehle erstellen die folgenden Binärdateien.

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

2. Kopieren Sie die Dateien in das `amazon-gamelift-plugin-unreal\GameLiftPlugin\Source\GameliftServer\ThirdParty\GameLiftServerSDK\Win64\` Verzeichnis .

Führen Sie die folgenden Aufgaben aus und arbeiten Sie in Ihren Spielprojektdateien.

1. Installieren Sie die Plugin-Dateien.

   a. Suchen Sie den Stammordner Ihres Spieleprojekts, z. B. `... > Unreal Projects/[project-name]/`. Wenn der Plugins-Ordner dort nicht vorhanden ist, erstellen Sie ihn.

   b. Gehen Sie zu dem `amazon-gamelift-plugin-unreal` Ordner, der aus entpackt wurde`amazon-gamelift-plugin-unreal-<>-sdk-<>.zip`. Kopieren Sie den `GameLiftPlugin` Ordner aus dem `gamelift-plugin-unreal` Ordner in den `Plugins` Ordner im Spieleprojektverzeichnis.

2. Fügen Sie das Plugin der **`.uproject`** Datei hinzu.

   a. Öffnen Sie im Stammordner Ihres Spieleprojekts die `.uproject` Datei .

   b. Aktualisieren Sie die Datei, um dem `Plugins` Abschnitt „GameLiftPlugin“ und „WebBrowserWidget“ hinzuzufügen und sie zu aktivieren. Der folgende Code zeigt die aktualisierte `.uproject` Datei für ein Spiel namens „MyGame“.

```
UnrealProjects > MyGame > MyGame.uproject
{
  ...
  "Plugins": [
```

```
    {
      "Name": "ModelingToolsEditorMode",
      "Enabled": true,
      "TargetAllowList": [ "Editor" ]
    },
    {
      "Name": "GameLiftPlugin",
      "Enabled": true
    },
    {
      "Name": "WebBrowserWidget",
      "Enabled": true
    }
  ]
}
```

3. Ändern Sie die UE-Editor-Version für Ihr Projekt.

   Wenn Sie ein Projekt für eine Editorversion erstellt haben und jetzt zu einer anderen Version wechseln möchten (z. B. eine Quellerstellungsversion), müssen Sie das Projekt aktualisieren.

   Wählen Sie im Stammordner Ihres Spieleprojekts die `.uproject` Datei aus und wählen Sie die Option Switch Unreal Engine Version aus. Wählen Sie eine neue Editorversion aus.

4. Erstellen Sie die Projektlösung mit Ihren Updates neu.

   a. Suchen Sie im Projektstammordner nach einer Lösungsdatei (`*.sln`). Wenn keine vorhanden ist, wählen Sie die `.uproject` Datei aus und wählen Sie die Option Visual Studio-Projektdateien generieren aus.

   b. Öffnen Sie die Lösungsdatei und erstellen oder erstellen Sie das Projekt neu.

5. Stellen Sie sicher, dass das Plugin im UE-Editor aktiviert ist.

   > **Note**
   >
   > Wenn Sie den Editor bereits geöffnet haben, müssen Sie den Editor möglicherweise neu starten, bevor er das neue Plugin erkennt.

   a. Öffnen Sie das Projekt in dem von Ihnen ausgewählten UE-Editor.

   b. Überprüfen Sie die Symbolleiste des Haupteditors auf die neue Schaltfläche im Amazon-GameLift Menü [Bild erforderlich].

c. Suchen Sie im Content Browser nach den Amazon- GameLift Plug-In-Assets. Stellen Sie sicher, dass in Ihrer Einstellung Ansichtsoptionen die Option Plugin-Inhalt anzeigen ausgewählt ist.

Einrichten eines AWS-Benutzerprofils

Richten Sie nach der Installation des Plugins ein Profil ein und verknüpfen Sie es mit einem gültigen AWS Kontobenutzer. Sie können mehrere Profile verwalten, aber Sie können jeweils nur ein Profil aktiv haben. Wenn Sie im Plugin arbeiten, wählen Sie ein zu verwendendes Profil aus.

Durch die Pflege mehrerer Profile können Sie zwischen verschiedenen Hosting-Szenarien wechseln. Sie können beispielsweise Profile mit denselben AWS Anmeldeinformationen, aber unterschiedlichen AWS Regionen einrichten. Oder Sie können Profile mit unterschiedlichen AWS Konten oder mit unterschiedlichen Benutzern/Berechtigungssätzen einrichten.

> **Note**
>
> Wenn Sie die AWS CLI auf Ihrer Workstation installiert haben und bereits ein Profil konfiguriert ist, kann das Amazon GameLift -Plugin es erkennen und als vorhandenes Profil auflisten. Das Plugin wählt automatisch jedes Profil mit dem Namen aus`[default]`. Sie können ein vorhandenes Profil verwenden oder ein neues erstellen.

So verwalten Sie Ihre AWS Profile

1. Wählen Sie in der Hauptsymbolleiste Unreal Editor das Menü Amazon GameLift und anschließend AWS Benutzerprofile festlegen aus. Diese Aktion öffnet Projekteinstellungen für das Plugin. Erweitern Sie den Abschnitt AWS Benutzerprofile.
2. Wenn das Plugin kein vorhandenes Profil erkennt, werden Sie aufgefordert, eines zu erstellen. Sie können ein neues Profil entweder mit einem neuen oder einem vorhandenen AWS Konto erstellen.

   > **Note**
   >
   > Sie müssen die -AWSManagementkonsole verwenden, um ein neues AWS Konto zu erstellen und einen Benutzer mit dem richtigen Berechtigungssatz zu erstellen oder zu aktualisieren.

Beim Einrichten eines Profils benötigen Sie die folgenden Informationen:

- Ein AWS-Konto. Wenn Sie ein neues AWS Konto erstellen müssen, folgen Sie den Anweisungen, um das Konto zu erstellen. Weitere Informationen finden Sie unter Erstellen eines -AWSKontos.
- Ein -AWSBenutzer mit Berechtigungen zur Nutzung von Amazon GameLift und anderen erforderlichen AWS Services. Anweisungen zum Einrichten eines AWS Identity and Access Management (IAM)-Benutzers mit Amazon- GameLift Berechtigungen und programmatischem Zugriff mit langfristigen Anmeldeinformationen Richten Sie ein AWS-Konto finden Sie unter .
- Anmeldeinformationen für Ihren AWS Benutzer. Diese Anmeldeinformationen bestehen aus einer AWS Zugriffsschlüssel-ID und einem AWS geheimen Schlüssel. Weitere Informationen finden Sie unter Abrufen Ihrer Zugriffsschlüssel.
- AWS Region. Dies ist ein geografischer Standort, an dem Sie Ihre AWS Ressourcen für das Hosting erstellen möchten. Während der Entwicklung empfehlen wir, eine Region in der Nähe Ihres physischen Standorts zu verwenden. Sehen Sie sich die Liste der unterstützten AWS Regionen an.

3. Wenn das Plugin ein vorhandenes Profil erkennt, werden Sie nicht aufgefordert, eines zu erstellen. Wenn Sie ein Profil aktualisieren oder ein neues erstellen möchten, wählen Sie Profil verwalten.

So bootstrappen Sie Ihr Profil:

Alle Profile müssen gebootet werden, um mit dem Amazon- GameLift Plugin verwendet zu werden. Bootstrapping erstellt einen Amazon S3-Bucket, der für das Profil spezifisch ist. Es wird verwendet, um Projektkonfigurationen, Build-Artefakte und andere Abhängigkeiten zu speichern. Buckets werden nicht zwischen anderen Profilen geteilt.

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

1. Wählen Sie in der Hauptsymbolleiste Unreal Editor das Amazon- GameLift Symbol und dann AWS Benutzerprofile festlegen aus. Diese Aktion öffnet Projekteinstellungen für das Plugin. Erweitern Sie den Abschnitt AWS Benutzerprofile.
2. Wählen Sie im Abschnitt Bootstrap Ihres Profils ein Profil aus der Dropdown-Liste aus und überprüfen Sie den Bootstrap-Status. Wenn der Status angibt, dass kein Bucket vorhanden ist,

wählen Sie die Schaltfläche Bootstrap und Profil erstellen, um einen Amazon S3-Bucket für das ausgewählte Profil zu erstellen.

3. Warten Sie, bis sich der Bootstrap-Status in „Aktiv“ ändert. Dies kann einige Minuten dauern.

Einrichten Ihres Spiels zum Testen mit Amazon GameLift Anywhere

In diesem Workflow fügen Sie Client- und Server-Spielcode für die Amazon GameLift-Funktionalität hinzu und verwenden das Plugin, um Ihre lokale Workstation als Testspielserver-Host zu kennzeichnen. Wenn Sie die Integrationsaufgaben abgeschlossen haben, verwenden Sie das -Plugin, um Ihre Spielclient- und Serverkomponenten zu erstellen.

So starten Sie den Amazon GameLift -Anywhere-Workflow:

- Wählen Sie in der Hauptsymbolleiste Unreal Editor das Menü Amazon GameLift und dann Hosten mit Anywhere aus. Diese Aktion öffnet die Plugin-Seite Deploy Anywhere, die einen sechsstufigen Prozess zum Integrieren, Erstellen und Starten Ihrer Spielkomponenten darstellt.

Schritt 1: Festlegen Ihres Profils

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

1. Wählen Sie ein Profil aus der Dropdown-Liste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum Amazon- GameLift Menü und wählen Sie AWS Benutzerprofile festlegen aus.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie Bootstrap-Profil und warten Sie, bis sich der Status in „Aktiv“ ändert.

Schritt 2: Einrichten Ihres Spielcodes

In diesem Schritt nehmen Sie eine Reihe von Aktualisierungen an Ihrem Client- und Servercode vor, um die Hosting-Funktionalität hinzuzufügen. Wenn Sie noch keine vom Quellcode erstellte Version des Unreal-Editors eingerichtet haben, enthält das Plugin Links zu Anweisungen und Quellcode.

Mit dem -Plugin kann einige Vorteile bei der Integration Ihres Spielcodes nutzen. Sie können eine minimale Integration durchführen, um grundlegende Hosting-Funktionen einzurichten. Sie können auch eine umfassendere benutzerdefinierte Integration durchführen. Die Informationen in diesem Abschnitt beschreiben die minimale Integrationsoption. Verwenden Sie die im Plugin enthaltenen Testkarten, um Ihrem Spielprojekt Client-Amazon- GameLift Funktionalität hinzuzufügen. Verwenden Sie für die Serverintegration das bereitgestellte Codebeispiel, um den Spielmodus Ihres Projekts zu aktualisieren.

Integrieren des Server-Spielmodus

Fügen Sie Ihrem Spiel Servercode hinzu, der die Kommunikation zwischen Ihrem Spieleserver und dem Amazon- GameLift Service ermöglicht. Ihr Spieleserver muss in der Lage sein, auf Anfragen von Amazon zu antworten GameLift, z. B. um eine neue Spielsitzung zu starten und auch den Status über den Zustand des Spieleservers und die Spielerverbindungen zu melden.

1. Öffnen Sie in Ihrem Code-Editor die Lösungsdatei (`.sln`) für Ihr Spielprojekt, die sich normalerweise im Stammordner des Projekts befindet. Zum Beispiel: `GameLiftUnrealApp.sln`.
2. Suchen Sie bei geöffneter Lösung die Projektspielmodus-Header-Datei: `[project-name]GameMode.h` Datei. Zum Beispiel: `GameLiftUnrealAppGameMode.h`.
3. Ändern Sie die Header-Datei so, dass sie dem folgenden Beispielcode entspricht. Ersetzen Sie unbedingt „GameLiftServer“ durch Ihren eigenen Projektnamen. Diese Updates sind spezifisch für den Spieleserver. Wir empfehlen Ihnen, eine Sicherungskopie der ursprünglichen Spielemodusdateien zur Verwendung mit Ihrem Client zu erstellen.

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

#pragma once

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

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

UCLASS(minimalapi)
class AGameLiftServerGameMode : public AGameModeBase
```

```
{
    GENERATED_BODY()

public:
    AGameLiftServerGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

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

4. Öffnen Sie die zugehörige Quelldatei `[project-name]GameMode.cpp` (z. B. `GameLiftUnrealAppGameMode.cpp`). Ändern Sie den Code so, dass er dem folgenden Beispielcode entspricht. Ersetzen Sie unbedingt „GameLiftUnrealApp“ durch Ihren eigenen Projektnamen. Diese Updates sind spezifisch für den Spieleserver. Wir empfehlen Ihnen, eine Sicherungskopie der Originaldatei zur Verwendung mit Ihrem Client zu erstellen.

   Der folgende Beispielcode zeigt, wie Sie die mindestens erforderlichen Elemente für die Serverintegration mit Amazon hinzufügen GameLift:

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

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

#include "GameLiftServerGameMode.h"
```

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

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

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

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

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

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

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

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

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

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

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

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

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

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

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

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

```
            // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
            ServerParametersForAnywhere.m_processId =
                TCHAR_TO_UTF8(
                    *FText::Format(
                        FText::FromString("ProcessId_{0}"),
                        FText::AsNumber(std::time(nullptr))
                    ).ToString()
                );
        }

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

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

        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Web Socket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

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

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

    ProcessParameters = MakeShared<FProcessParameters>();

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

    //OnProcessTerminate callback. GameLift will invoke this callback before
 shutting down an instance hosting this game server.
    //It gives this game server a chance to save its state, communicate with
 services, etc., before being shut down.
    //In this case, we simply tell GameLift we are indeed going to shutdown.
```

```
    ProcessParameters->OnTerminate.BindLambda([=]()
    {
        UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
        GameLiftSdkModule->ProcessEnding();
    });

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

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

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

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

        if (SwitchStr.Split("=", &Key, &Value))
        {
            if (Key.Equals("port"))
            {
                ProcessParameters->port = FCString::Atoi(*Value);
            }
        }
    }

    //Here, the game server tells GameLift where to find game session log files.
    //At the end of a game session, GameLift uploads everything in the specified
    //location and stores it in the cloud for access later.
```

```
    TArray<FString> Logfiles;
    Logfiles.Add(TEXT("GameServerLog/Saved/Logs/GameServerLog.log"));
    ProcessParameters->logParameters = Logfiles;

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

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

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

Integrieren Ihrer Client-Spielkarte

Die Startup-Spielekarte enthält Blueprint-Logik- und Benutzeroberflächenelemente, die bereits Basiscode enthalten, um Spielsitzungen anzufordern und Verbindungsinformationen zu verwenden, um eine Verbindung zu einer Spielsitzung herzustellen. Sie können die Zuordnung unverändert verwenden oder diese nach Bedarf ändern. Verwenden Sie die Startup-Spielkarte mit anderen Spielkomponenten, z. B. dem von Unreal Engine bereitgestellten Vorlagenprojekt für Dritte. Diese Komponenten sind in Content Browser verfügbar. Sie können sie verwenden, um die Bereitstellungsworkflows des Plugins zu testen, oder als Leitfaden, um einen benutzerdefinierten Backend-Service für Ihr Spiel zu erstellen.

Die Startup-Übersicht weist die folgenden Merkmale auf:

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

So verwenden Sie die Startup-Spielekarte

1. Öffnen Sie im UE-Editor die Seite Projekteinstellungen, Karten und Modi und erweitern Sie den Abschnitt Standardkarten.
2. Wählen Sie für Editor Startup Map die Option „StartupMap“ aus der Dropdown-Liste aus. Möglicherweise müssen Sie nach der Datei suchen, die sich in befindet`... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Plugin Content/Maps`.
3. Wählen Sie für Game Default Map das gleiche „StartupMap“ aus der Dropdown-Liste aus.
4. Wählen Sie für Server Default Map die Option „“ ausThirdPersonMap. Dies ist eine Standardkarte, die in Ihrem Spielprojekt enthalten ist. Diese Karte ist für zwei Spieler im Spiel konzipiert.
5. Öffnen Sie das Detailfenster für die Server-Standardzuordnung. Setzen Sie GameMode Override auf „Keine“.
6. Erweitern Sie den Abschnitt Standardmodi und setzen Sie den globalen Standardserver-Spielmodus auf den Spielmodus, den Sie für Ihre Serverintegration aktualisiert haben.

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

Erstellen Ihrer Spielkomponenten

1. Erstellen neuer Server- und Client-Zieldateien

    a. Gehen Sie in Ihrem Spieleprojektordner zum Quellordner und suchen Sie die `Target.cs` Dateien.

b. Kopieren Sie die Datei `[project-name]Editor.Target.cs` in zwei neue Dateien mit dem Namen `[project-name]Client.Target.cs` und `[project-name]Server.Target.cs`.

c. Bearbeiten Sie jede der neuen Dateien, um die Werte für Klassennamen und Zieltyp zu aktualisieren, wie gezeigt:

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

using UnrealBuildTool;
using System.Collections.Generic;

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

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

using UnrealBuildTool;
using System.Collections.Generic;

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

2. Aktualisieren Sie die `-.Build.cs`Datei.

   a. Öffnen Sie die `.Build.cs` Datei für Ihr Projekt. Diese Datei befindet sich unter `UnrealProjects/[project name]/Source/[project name]/[project name].Build.cs`.

   b. Aktualisieren Sie die `ModuleRules` Klasse wie im folgenden Codebeispiel gezeigt.

```
public class MyGame : ModuleRules
{
  public GameLiftUnrealApp(TargetInfo Target)
  {
    PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
 "Engine", "InputCore" });
    bEnableExceptions = true;

    if (Target.Type == TargetRules.TargetType.Server)
    {
      PublicDependencyModuleNames.AddRange(new string[]
 { "GameLiftServerSDK" });
      PublicDefinitions.Add("WITH_GAMELIFT=1");
    }
    else
    {
      PublicDefinitions.Add("WITH_GAMELIFT=0");
    }
  }
}
```

3. Erstellen Sie Ihre Spielprojektlösung neu.
4. Öffnen Sie Ihr Spielprojekt in einer quellgefertigten Version des Unreal Engine-Editors.
5. Gehen Sie für Ihren Client und Ihren Server wie folgt vor:

   a. Wählen Sie ein Ziel aus. Gehen Sie zu Plattformen, Windows und wählen Sie eine der folgenden Optionen aus:

      - Server: `[your-application-name]Server`
      - Client: `[your-application-name]Client`

   b. Starten Sie den Build. Gehen Sie zu Plattform, Windows, Paketprojekt .

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

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

Schritt 3: Herstellen einer Verbindung mit einer Anywhere-Flotte

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

- Wenn das AWS Konto, das Sie derzeit verwenden, über vorhandene Anywhere-Flotten verfügt, öffnen Sie das Dropdown-Feld Flottenname und wählen Sie eine Flotte aus. In diesem Dropdown-Menü werden nur die Anywhere-Flotten in der AWS Region für das aktuell aktive Benutzerprofil angezeigt.
- Wenn keine Flotten vorhanden sind oder Sie eine neue erstellen möchten, wählen Sie Neue Anywhere-Flotte erstellen und geben Sie einen Flottennamen an.

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

Schritt 4: Registrieren Ihrer Workstation

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

1. Geben Sie einen Datenverarbeitungsnamen für Ihren lokalen Computer ein. Wenn Sie mehr als eine Rechenleistung in der Flotte hinzufügen, müssen die Namen eindeutig sein.
2. Geben Sie eine IP-Adresse für Ihren lokalen Computer an. In diesem Feld wird standardmäßig die öffentliche IP-Adresse Ihres Computers verwendet. Sie können localhost (127.0.0.1) auch verwenden, solange Sie Ihren Spielclient und -server auf demselben Computer ausführen.
3. Wählen Sie Rechenleistung registrieren aus. Sie können den Fortschritt dieser Anforderung im Ausgabeprotokoll des Unreal-Editors verfolgen.

Als Reaktion auf diese Aktion GameLift überprüft Amazon, ob es eine Verbindung zur Datenverarbeitung herstellen kann, und gibt Informationen über die neu registrierte

Datenverarbeitung zurück. Außerdem werden die Konsolenargumente erstellt, die Ihre ausführbaren Spieldateien bei der Initialisierung der Kommunikation mit dem Amazon- GameLift Service benötigen.

Schritt 5: Generieren eines Authentifizierungstokens

Spieleserverprozesse, die auf Ihrer Anywhere-Datenverarbeitung ausgeführt werden, benötigen ein Authentifizierungstoken, um den GameLift Service aufzurufen. Das Plugin generiert und speichert automatisch ein Authentifizierungstoken für die Anywhere-Flotte, wenn Sie den Spieleserver über das Plugin starten. Der Authentifizierungstoken-Wert wird als Befehlszeilenargument gespeichert, das Ihr Servercode zur Laufzeit abrufen kann.

In diesem Schritt müssen Sie keine Maßnahmen ergreifen.

Schritt 6: Starten des Spiels

An diesem Punkt haben Sie alle Aufgaben abgeschlossen, die zum Starten und Abspielen Ihres Multiplayer-Spiels auf einer lokalen Workstation mit Amazon erforderlich sind GameLift.

1. Starten Sie Ihren Spieleserver. Der Spieleserver benachrichtigt Amazon GameLift , wenn er bereit ist, Spielsitzungen zu hosten.
2. Starten Sie Ihren Spielclient und verwenden Sie die neue Funktionalität, um eine neue Spielsitzung zu starten. Diese Anfrage wird GameLift über den neuen Backend-Service an Amazon gesendet. Als Reaktion GameLiftruft Amazon den Spieleserver auf, der auf Ihrem lokalen Computer ausgeführt wird, um eine neue Spielsitzung zu starten. Wenn die Spielsitzung bereit ist, Spieler anzunehmen, GameLift stellt Amazon Verbindungsinformationen für den Spielclient bereit, um der Spielsitzung beizutreten.

Bereitstellen Ihres Spiels für das Cloud-Hosting mit verwalteten EC2-Flotten

In diesem Workflow verwenden Sie das Plugin, um Ihr Spiel für das Hosting auf Cloud-basierten Rechenressourcen zu ändern, die von Amazon verwaltet werden GameLift. Sie fügen Client- und Server-Spielcode für die Amazon- GameLift Funktionalität hinzu und laden dann Ihren Server-Build auf den Amazon- GameLift Service hoch, um ihn in den Cloud-basierten Ressourcen bereitzustellen. Wenn dieser Workflow abgeschlossen ist, verfügen Sie über einen funktionierenden Spieleclient, der eine Verbindung zu Ihren Spieleservern in der Cloud herstellen kann.

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

- Wählen Sie in der Hauptsymbolleiste Unreal Editor das GameLift Menü Amazon und dann Host mit verwaltetem EC2 aus. Diese Aktion öffnet die Plugin-Seite Bereitstellen der Amazon EC2-Flotte , die einen sechsstufigen Prozess zum Integrieren, Erstellen, Bereitstellen und Starten Ihrer Spielkomponenten darstellt.

Schritt 1: Festlegen Ihres Profils

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

1. Wählen Sie ein Profil aus der Dropdown-Liste der verfügbaren Profile aus. Wenn Sie noch kein Profil haben oder ein neues erstellen möchten, gehen Sie zum Amazon- GameLift Menü und wählen Sie AWS Benutzerprofile festlegen aus.
2. Wenn der Bootstrap-Status nicht „Aktiv“ lautet, wählen Sie Bootstrap-Profil und warten Sie, bis sich der Status in „Aktiv“ ändert.

Schritt 2: Einrichten Ihres Spielcodes

In diesem Schritt nehmen Sie eine Reihe von Aktualisierungen an Ihrem Client- und Servercode vor, um die Hosting-Funktionalität hinzuzufügen. Wenn Sie noch keine vom Quellcode erstellte Version des Unreal-Editors eingerichtet haben, enthält das Plugin Links zu Anweisungen und Quellcode.

Wenn Sie Ihr Spiel für die Verwendung mit einer Anywhere-Flotte integriert haben, müssen Sie keine Änderungen an Ihrem Spielcode vornehmen. Wenn Sie die Startup-Spielkarte verwenden, funktioniert dies auch mit EC2-Bereitstellungen.

- Einrichten Ihres Spielcodes (beliebig)
- Erstellen Ihrer Spielkomponenten

Nachdem Sie Ihren Spieleserver erstellt haben, führen Sie die folgenden Aufgaben aus, um ihn auf das Hochladen in Amazon vorzubereiten GameLift.

So verpacken Sie Ihren Server-Build für die Cloud-Bereitstellung

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

1. Kopieren Sie das Installationsskript, das im Plugin-Download enthalten ist, in den Stamm des `WindowsServer` Ordners . Suchen Sie nach der Datei `[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat`. Amazon GameLift verwendet diese Datei, um den Server-Build auf jeder EC2-Hosting-Ressource zu installieren.
2. Kopieren Sie die in Ihrer Visual Studio-Installation enthaltene `VC_redist.x64.exe` Datei in den Stamm des `WindowsServer` Ordners . Diese Datei befindet sich üblicherweise unter `C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`.
3. Kopieren Sie die OpenSSL-DLLs für Ihren Spieleserver-Build in den Ordner `WindowsServer/MyGame/Binaries/Win64`. Stellen Sie sicher, dass die DLLs für dieselbe Version gelten, die im Server-Build verwendet wird. Kopieren Sie die folgenden Dateien:
   - `libssl-3-x64.dll`
   - `libcrypto-3-x64.dll`

Schritt 3: Auswählen des Bereitstellungsszenarios

In diesem Schritt wählen Sie die Spiele-Hosting-Lösung aus, die Sie zu diesem Zeitpunkt bereitstellen möchten. Sie können mehrere Bereitstellungen Ihres Spiels haben, indem Sie eines der Szenarien verwenden.

- Flotte für eine einzelne Region: Stellt Ihren Spieleserver für eine einzelne Flotte von Hosting-Ressourcen in der AWS Standardregion des aktiven Profils bereit. Dieses Szenario ist ein guter Ausgangspunkt für das Testen Ihrer Serverintegration mit AWS und Server-Build-Konfiguration. Es stellt die folgenden Ressourcen bereit:
  - AWS -Flotte (On-Demand) mit installiertem und ausgeführtem Spieleserver-Build.
  - Amazon Cognito-Benutzerpool und -Client, damit Spieler ein Spiel authentifizieren und starten können.
  - API-Gateway-Genehmiger, der den Benutzerpool mit APIs verknüpft.
  - WebACl zur Drosselung übermäßiger Player-Aufrufe an das API-Gateway.

  - API-Gateway + Lambda-Funktion für Spieler, um einen Spielslot anzufordern. Diese Funktion ruft auf`CreateGameSession()`, wenn keine verfügbar sind.
  - API-Gateway + Lambda-Funktion für Spieler, um Verbindungsinformationen für ihre Spielanfrage abzurufen.
- FlexMatch -Flotte: Stellt Ihren Spieleserver für eine Reihe von Flotten bereit und richtet einen FlexMatch Matchmaker mit Regeln ein, um Spieler-Matches zu erstellen. In diesem Szenario wird kostengünstiges Spot-Hosting mit einer Struktur mit mehreren Flotten und mehreren Standorten verwendet, um eine dauerhafte Verfügbarkeit zu gewährleisten. Dieser Ansatz ist nützlich, wenn Sie bereit sind, mit der Entwicklung einer Matchmaker-Komponente für Ihre Hosting-Lösung zu beginnen. In diesem Szenario erstellen Sie die grundlegenden Ressourcen für diese Lösung, die Sie später nach Bedarf anpassen können. Es stellt die folgenden Ressourcen bereit:
  - FlexMatch Matchmaking-Konfiguration und Matchmaking-Regelsatz zur Annahme von Spieleranfragen und Formularübereinstimmungen.
  - Drei AWS Flotten mit installiertem und ausgeführtem Spieleserver-Build an mehreren Standorten. Enthält zwei Spot-Flotten und eine On-Demand-Flotte als Backup.
  - AWS Platzierungswarteschlange für Spielsitzungen, die Anfragen nach vorgeschlagenen Übereinstimmungen erfüllt, indem sie die bestmögliche Hosting-Ressource findet (basierend auf der Realisierbarkeit, den Kosten, der Spielerlatenz usw.) und eine Spielsitzung startet.
  - Amazon Cognito-Benutzerpool und -Client, damit Spieler ein Spiel authentifizieren und starten können.
  - API-Gateway-Genehmiger, der den Benutzerpool mit APIs verknüpft.
  - WebACl zur Drosselung übermäßiger Player-Aufrufe an das API-Gateway.
  - API-Gateway + Lambda-Funktion für Spieler, um einen Spielslot anzufordern. Diese Funktion ruft auf`StartMatchmaking()`.
  - API-Gateway + Lambda-Funktion für Spieler, um Verbindungsinformationen für ihre Spielanfrage abzurufen.
  - Amazon-DynamoDB-Tabellen zum Speichern von Matchmaking-Tickets für Spieler und Spielsitzungsinformationen.
  - SNS-Thema + Lambda-Funktion zur Verarbeitung von GameSessionQueue Ereignissen.

Schritt 4: Festlegen von Spielparametern

In diesem Schritt beschreiben Sie Ihr Spiel zum Hochladen in AWS .

- Server-Build-Name: Geben Sie einen aussagekräftigen Namen für Ihren Spieleserver-Build an. AWS verwendet diesen Namen, um auf die Kopie Ihres Server-Builds zu verweisen, die hochgeladen und für Bereitstellungen verwendet wird.
- Server Build OS: Geben Sie das Betriebssystem ein, auf dem Ihr Server ausgeführt werden soll. Dies gibt anAWS, welche Art von Rechenressourcen zum Hosten Ihres Spiels verwendet werden sollen.
- Spieleserver-Ordner: Identifizieren Sie den Pfad zu Ihrem lokalen Server-Build-Ordner.
- Spieleserver-Build: Identifizieren Sie den Pfad zur ausführbaren Spieleserver-Datei.
- Spielclient-Pfad: Identifizieren Sie den Pfad zur ausführbaren Spielclient-Datei.
- Client-Konfigurationsausgabe: Dieses Feld muss auf einen Ordner in Ihrem Client-Build verweisen, der Ihre AWS Konfiguration enthält. Suchen Sie danach am folgenden Speicherort: `[client-build]/[project-name]/Content/CloudFormation`.

Schritt 5: Bereitstellen eines Szenarios

In diesem Schritt stellen Sie Ihr Spiel auf einer Cloud-Hosting-Lösung bereit, die auf dem ausgewählten Bereitstellungsszenario basiert. Dieser Vorgang kann bis zu 40 Minuten dauern, während Ihren Server-Build AWS validiert, Hosting-Ressourcen bereitstellt, Ihren Spieleserver installiert, Serverprozesse startet und sie für das Hosten von Spielsitzungen bereit macht.

Um mit der Bereitstellung zu beginnen, wählen Sie Bereitstellen aus CloudFormation. Sie können den Status Ihres Spiele-Hostings hier verfolgen. Für detailliertere Informationen können Sie sich bei der -AWSManagementkonsole für anmelden AWS und Ereignisbenachrichtigungen anzeigen. Stellen Sie sicher, dass Sie sich mit demselben Konto, demselben Benutzer und derselben AWS Region wie das aktive Benutzerprofil im Plugin anmelden.

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

Schritt 6: Starten des Clients

An diesem Punkt haben Sie alle Aufgaben abgeschlossen, die zum Starten und Abspielen Ihres Multiplayer-Spiels erforderlich sind, das mit Amazon gehostet wird GameLift. Um das Spiel zu spielen, starten Sie eine Instance Ihres Spielclients.

Wenn Sie das Szenario mit einer einzelnen Flotte bereitgestellt haben, können Sie eine einzelne Client-Instance mit einem Player öffnen, die Serverkarte eingeben und sich bewegen. Öffnen Sie

zusätzliche Instances des Spielclients, um der gleichen Server-Spielekarte einen zweiten Spieler hinzuzufügen.

Wenn Sie das FlexMatch Szenario bereitgestellt haben, wartet die Lösung darauf, dass mindestens zwei Clients für die Platzierung von Spielsitzungen in die Warteschlange gestellt werden, bevor die Spieler die Serverkarte betreten können.

Flottendaten für eine GameLift Amazon-Instance abrufen

Es gibt Situationen, in denen Ihr benutzerdefinierter Game-Build oder Ihr Realtime Server-Skript möglicherweise Informationen über die GameLift Amazon-Flotte benötigt. Ihr Spiel-Build oder Ihr Skript könnte beispielsweise Code für Folgendes enthalten:

- Überwachen Sie Aktivitäten auf der Grundlage von Flottendaten.
- Führen Sie Kennzahlen zusammen, um die Aktivitäten anhand von Flottendaten zu verfolgen. (Viele Spiele verwenden diese Daten für LiveOps Aktivitäten.)
- Stellen Sie relevante Daten für benutzerdefinierte Spieledienste bereit, z. B. für Matchmaking, zusätzliche Kapazitätsskalierung oder Tests.

Flotteninformationen sind als JSON-Datei auf jeder Instanz an den folgenden Speicherorten verfügbar:

- Windows: `C:\GameMetadata\gamelift-metadata.json`
- Linux: `/local/gamemetadata/gamelift-metadata.json`

Die `gamelift-metadata.json` Datei enthält die Attribute einer GameLift Amazon-Flottenressource.

Beispiel für eine JSON-Datei:

```
{
    "buildArn":"arn:aws:gamelift:us-west-2:123456789012:build/
build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
    "buildId":"build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
    "fleetArn":"arn:aws:gamelift:us-west-2:123456789012:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "fleetDescription":"Test fleet for Really Fun Game v0.8",
    "fleetId":"fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
```

```
    "fleetName":"ReallyFunGameTestFleet08",
    "fleetType":"ON_DEMAND",
    "instanceRoleArn":"arn:aws:iam::123456789012:role/S3AccessForGameLift",
    "instanceType":"c5.large",
    "serverLaunchPath":"/local/game/reallyfungame.exe"
}
```

FlexMatchMatchmaking hinzufügen

Verwenden Sie Amazon GameLiftFlexMatch, um Ihren von Amazon GameLift gehosteten Spielen eine Matchmaking-Funktion für Spieler hinzuzufügen. Sie können es entweder FlexMatch mit benutzerdefinierten Spielservern oder Echtzeitservern verwenden.

FlexMatch verbindet den Matchmaking-Service mit einer anpassbaren Regel-Engine. Du gestaltest, wie du Spieler zusammenbringst, basierend auf Spielerattributen und Spielmodi, die für dein Spiel Sinn machen. FlexMatchverwaltet das A und O der Bewertung von Spielern, die nach einem Spiel suchen, der Zusammenstellung von Matches mit einer oder mehreren Teams und der Beginn von Spielsitzungen, um die Spiele auszurichten.

Um den vollen FlexMatch Service nutzen zu können, müssen Sie Ihre Hosting-Ressourcen mit Warteschlangen eingerichtet haben. Amazon GameLift verwendet Warteschlangen, um die bestmöglichen Hosting-Standorte für Spiele in verschiedenen Regionen und Computertypen zu finden. Insbesondere können GameLift Amazon-Warteschlangen Latenzdaten verwenden, sofern sie von Spieleclienten bereitgestellt werden, um Spielsitzungen so zu platzieren, dass die Spieler beim Spielen die niedrigstmögliche Latenz haben.

Weitere Informationen dazu, FlexMatch wie Sie detaillierte Hilfe bei der Integration von Matchmaking in Ihre Spiele integrieren können, finden Sie in den folgenden Themen des GameLiftFlexMatchAmazon-Entwicklerleitfadens:

- So GameLift FlexMatch funktioniert Amazon
- FlexMatchIntegrationsschritte

Verwaltung des Hostings mit GameLift Amazon-Containern

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Amazon GameLift bietet einen kompletten Cloud-Hosting-Service zur Unterstützung von containerisierten Lösungen für das Hosting von Spieleservern. Mit GameLift Amazon-Containerflotten können Sie die Vorteile von Containern wie Portabilität, Agilität und Fehlertoleranz nutzen.

Schlüsselfeatures

Die folgenden Funktionen sind für GameLift Amazon-Containerflotten verfügbar.

- Entwickeln Sie eine benutzerdefinierte Container-Architektur mit leichten Containern, um Ihre Spieleserver-Software auf GameLift Amazon-Hosting-Ressourcen auszuführen.
- Binden Sie den Amazon GameLift Agent ein, um den Lebenszyklus der Spieleserverprozesse in Ihren Containern zu verwalten. Der On-Compute-Agent führt Ihre Anweisungen aus, wann und wie Serverprozesse gestartet werden sollen und wie viele Serverprozesse für das Hosting von Spielsitzungen verwaltet werden müssen.
- Passen Sie die von Amazon bereitgestellten Ressourcen GameLift an, um Container-Images mit Ihrer Gameserver-Anwendung zu erstellen. Verwenden Sie die bereitgestellte Docker-Datei, um ein Linux-basiertes Container-Image zu erstellen. Speichern Sie Bilder für Ihre Containerflotten in einem privaten Repository von Amazon Elastic Container Registry (Amazon ECR).
- Sorgen Sie für Spielererlebnisse mit geringer Latenz, indem Sie Ressourcen für Containerflotten in beliebigen AWS-Region oder lokalen Zonen einsetzen, die Amazon GameLift unterstützt. Erstellen Sie Containerflotten mit mehreren Standorten für ein optimiertes Flottenmanagement. Siehe Amazon GameLift -Hosting-Standorte.
- Testen Sie Ihre containerisierten Game-Hosting-Lösungen mit einer GameLift Anywhere Amazon-Flotte. Verwenden Sie Anywhere, um Ihre Lösungsentwicklung lokal zu testen, einschließlich Ihrer Amazon GameLift SDK-Integration und Ihrer Container-Image-Konfigurationen.
- Verfolge die Leistung des Spiele-Hostings mit containerspezifischen Leistungskennzahlen. Überwachen Sie den Zustand Ihrer Flottenressourcen mithilfe von Hardwaremetriken.
- Verwenden Sie die Amazon-Tools zur Platzierung von GameLift Spielsitzungen, einschließlich Warteschlangen und FlexMatch Spielersuche, um Spielern die bestmöglichen Spielsitzungen zuzuordnen, die auf Ihren Containerflotten veranstaltet werden.

- Verwalten Sie Ressourcen für Containerflotten mithilfe von AWS CloudFormation Vorlagen für Amazon GameLift.

Verwendung von Containerflotten während der öffentlichen Vorschau

Die neue Funktion für Containerflotten befindet sich derzeit in der öffentlichen Vorschauversion. Während dieser Phase werden die folgenden GameLift Amazon-Funktionen unterstützt:

- Verwenden Sie Container-Flotten, um Spieleserver zu hosten, die für Linux entwickelt wurden. Containerflotten verwenden `Amazon_Linux_2023` und unterstützen Linux-Container-Images. Windows-Container werden nicht unterstützt.
- Integrieren Sie Spielserverprojekte nur mit Amazon GameLift Server SDK Version 5+. Frühere Versionen werden nicht unterstützt.
- Verwenden Sie einen der Amazon EC2 EC2-On-Demand-Instance-Typen, die Amazon GameLift unterstützt. Spot-Flotten werden derzeit nicht unterstützt.

So funktionieren Container bei Amazon GameLift

Diese Dokumentation bezieht sich auf eine Funktion, die in der öffentlichen Vorschauversion enthalten ist. Änderungen sind vorbehalten.

GameLift Amazon-Containerflotten sind so konzipiert, dass sie Flexibilität bei der Bereitstellung und Skalierung Ihrer containerisierten Anwendungen bieten. Es verwendet den Amazon Elastic Container Service (Amazon ECS), um die Bereitstellung und Ausführung von Aufgaben für Ihre GameLift Amazon-Flotten zu verwalten. Dieses Thema beschreibt die grundlegenden Strukturelemente für den Betrieb von Containern auf einer von Amazon GameLift verwalteten Flotte, veranschaulicht gängige Architekturen und skizziert einige Kernkonzepte.

Komponenten der Containerflotte

Flotte

Eine Containerflotte ist eine Sammlung von Amazon EC2 EC2-Instances, die von Amazon verwaltet werden GameLift und Ihre containerisierten Spieleserver betreiben. Wenn Sie eine Flotte erstellen, konfigurieren Sie, wie Ihre Container-Architektur und Gameserver-Software auf

jeder Flotteninstanz bereitgestellt werden. Sie können eine Containerflotte an einem einzelnen AWS-Region oder an mehreren geografischen Standorten einsetzen. Sie können die GameLift manuellen oder automatischen Skalierungstools von Amazon verwenden, um die Kapazität einer Containerflotte für die Ausrichtung von Spielsitzungen und Spielern zu skalieren.

Instance

Eine Amazon EC2 EC2-Instance ist der virtuelle Server, der Rechenkapazität für Ihr Spiele-Hosting bereitstellt. Bei Amazon GameLift können Sie aus einer Vielzahl von Instance-Typen wählen. Jeder Instance-Typ bietet eine andere Kombination aus CPU-, Arbeitsspeicher-, Speicher- und Netzwerkkapazität.

Wenn Sie eine Containerflotte erstellen, GameLift stellt Amazon Instances auf der Grundlage des von Ihnen ausgewählten Instance-Typs und Ihrer Flottenkonfiguration bereit. Jede bereitgestellte Flotteninstanz ist identisch und führt Ihre containerisierte Gameserver-Software auf die gleiche Weise aus. Die Anzahl der Instanzen in einer Flotte bestimmt die Größe der Flotte und die Hosting-Kapazität für Spiele.

Container-Gruppe

Amazon GameLift verwendet das Konzept einer Containergruppe, um eine Reihe von Containern zu beschreiben und zu verwalten. Eine Container-Gruppe ähnelt einer Container-"Task“ oder einem „Pod“. Innerhalb jeder Containergruppe können Sie definieren, wie Container die verfügbaren CPU- und Speicherressourcen gemeinsam nutzen. Sie können auch Abhängigkeiten zwischen Containern einrichten und den Lebenszyklus der Containergruppe verwalten.

Containergruppen können sich über jede Flotteninstanz hinweg replizieren, um die Ressourcennutzung zu optimieren. Sie können die Replikation verwalten, indem Sie die Planungsstrategie einer Containergruppe wie folgt festlegen:

- Replikat-Containergruppen verwalten die Container, auf denen deine Gameserver-Anwendung und unterstützende Software ausgeführt werden. Alle Containerflotten müssen eine Replikat-Containergruppe definieren. Eine Replikatgruppe kann mehrere Kopien auf jeder Flotteninstanz haben, abhängig von den Anforderungen der Containergruppe und den Ressourcen des verwendeten Instance-Typs. Alle Container in der Replikatgruppe werden automatisch innerhalb einer Instanz gemeinsam skaliert.
- Daemon-Containergruppen, die optional sind, können für die Ausführung von Hintergrunddiensten oder Hilfsprogrammen nützlich sein, z. B. für die Überwachung. Ihre Gameserver-Software hängt nicht direkt von Prozessen in einer Daemon-Gruppe ab. Daemon-Containergruppen werden nicht repliziert — jede Flotteninstanz hat höchstens eine Kopie der

Daemon-Gruppe. Das bedeutet, dass Container in einer Daemon-Gruppe nicht zusammen mit Containern in einer Replikatgruppe auf eine Flotteninstanz skaliert werden können.

Eine Containerflotte muss über eine Replikat-Containergruppe verfügen und kann optional über eine Daemon-Gruppe verfügen.

Container

Der Container ist das grundlegendste Element einer containerbasierten Architektur. Er besteht aus einem Container-Image mit ausführbaren Softwaredateien und abhängigen Dateien. Wenn Sie einen Container für die Verwendung mit Amazon definieren GameLift, konfigurieren Sie, wie die Software im Container ausgeführt wird.

Jede Containergruppe in einer Containerflotte muss über einen Container verfügen, der als „unverzichtbar“ gekennzeichnet ist. Ein unverzichtbarer Container bestimmt den Lebenszyklus einer Containergruppe. Wenn der Essential-Container ausfällt, wird die gesamte Containergruppe neu gestartet.

Zu den Containertypen gehören:

- Der Essential Replica Container enthält alles, was Sie zum Ausführen Ihrer Spieleserverprozesse und zum Hosten von Spielsitzungen für Spieler benötigen. Es umfasst Ihren Spieleserver-Build, der in das GameLift Amazon-Server-SDK integriert ist, und die dazugehörige Software. Dazu gehört auch der Amazon GameLift Agent, der den Lebenszyklus Ihrer Spieleserverprozesse verwaltet. Die Replikatcontainergruppe einer Flotte besteht aus genau einem wichtigen Replikatcontainer.
- Auf nicht unbedingt benötigten Replikat-Containern, auch „Sidecar“ -Container genannt, wird Software zur Unterstützung deiner Gameserver-Anwendung ausgeführt. Mithilfe eines Sidecar-Containers kannst du unterstützende Software parallel zu deinen Spieleservern ausführen und skalieren, sie aber als separate Container verwalten. Wenn dieser Containertyp ausfällt, wird nur der Container selbst neu gestartet. Die Containergruppe ist davon nicht betroffen.
- Daemon-Container führen einen Daemon-Dienst zur Verwaltung von Hintergrundprozessen aus. Ein Daemon-Container wird häufig verwendet, um einen Amazon CloudWatch (CloudWatch) -Agenten auszuführen, um Metriken, Protokolle und Traces für Ihre Container zu sammeln. Daemon-Container können essenziell oder unwichtig sein, je nachdem, wann ein Container-Ausfall zu einem Neustart der Containergruppe führen muss.

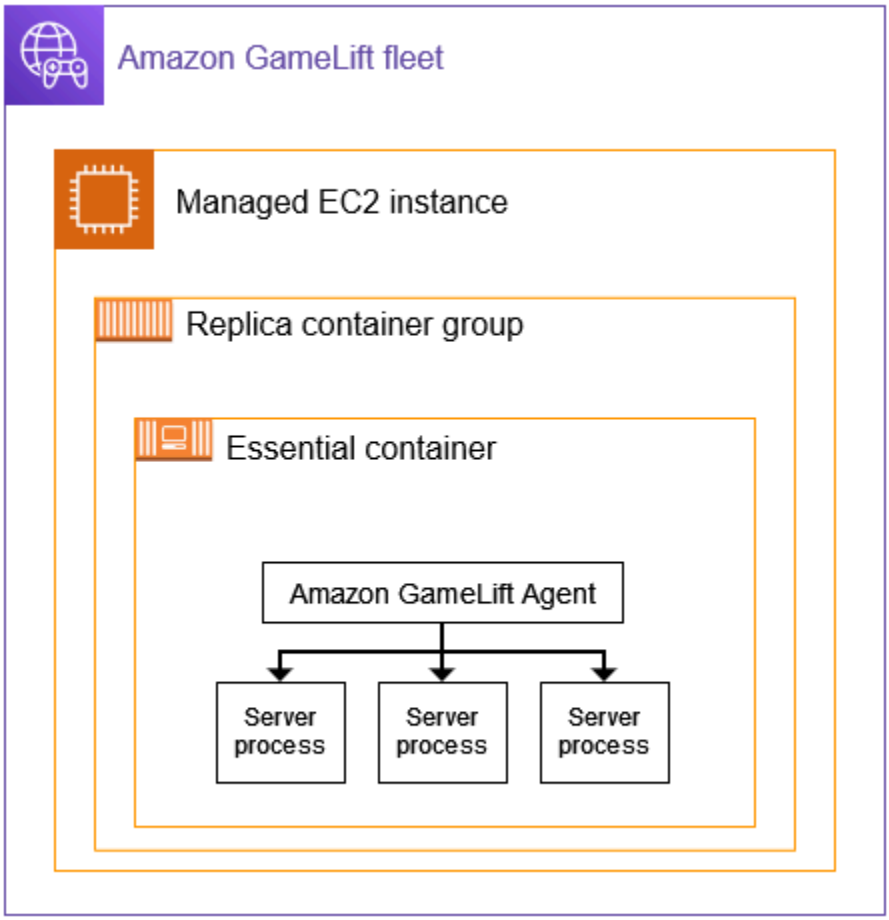
Datenverarbeitung

Eine Compute ist eine Flottenhosting-Ressource, die beim GameLift Amazon-Service registriert ist und mit dem Service kommunizieren kann. In einer Containerflotte ist eine Recheneinheit ein

Container mit einem Prozess, der den Rechenregistrierungsprozess verwaltet. Im unverzichtbaren Replikatcontainer einer Containerflotte registriert der GameLift Amazon-Agent diesen Container automatisch als Rechencontainer.

Gängige Architekturen

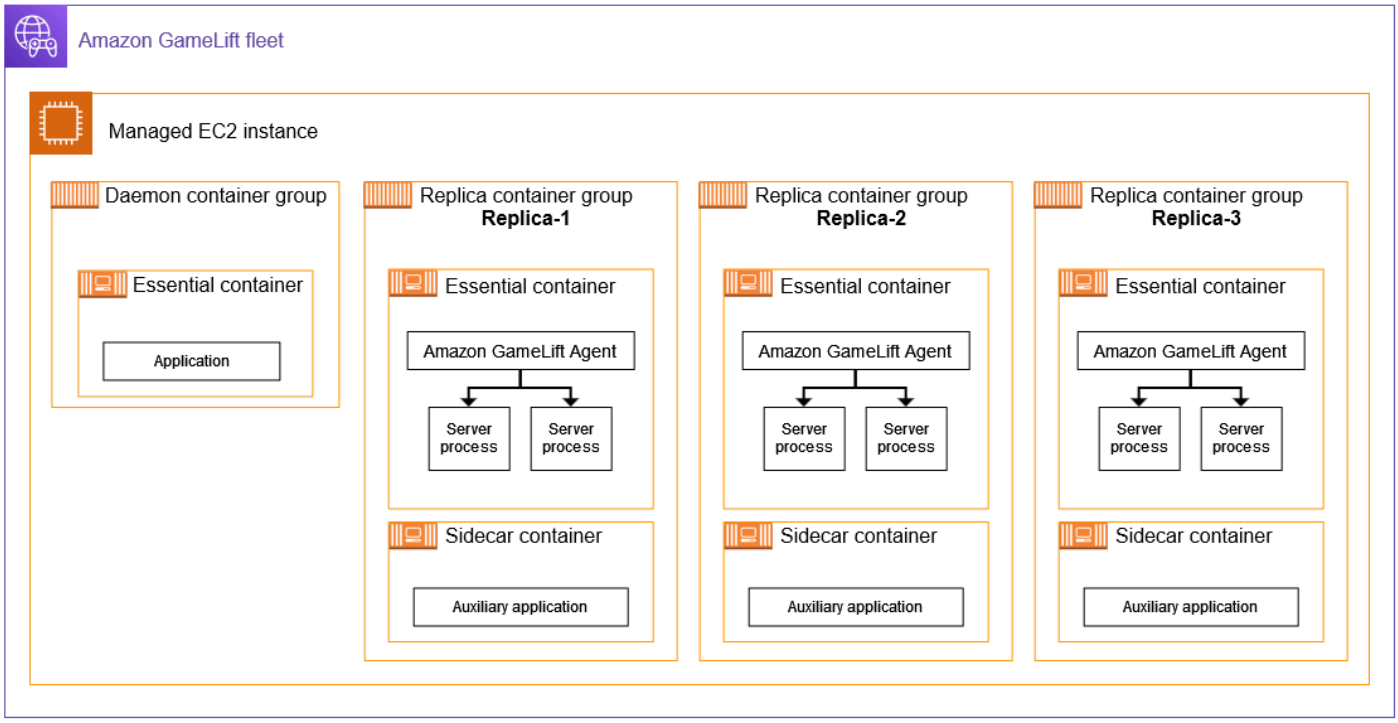
Das folgende Diagramm zeigt die einfachste Struktur der Containerflotte. In dieser Struktur verwaltet jede Instanz in der Flotte eine Kopie der Replikat-Containergruppe. Die Container-Gruppe besteht aus einem einzigen wichtigen Container, auf dem der Amazon GameLift Agent, die Spieleserver-Anwendung und die gesamte unterstützende Software für das Hosten von Spielsitzungen ausgeführt werden. Der Agent implementiert flottenspezifische Anweisungen, um drei Serverprozesse gleichzeitig auszuführen. Da es pro Instanz eine Replikat-Containergruppe gibt, führt jede Flotteninstanz drei Serverprozesse gleichzeitig aus.



Dieses zweite Beispiel veranschaulicht ein komplexeres Design einer Containerflotte. In diesem Beispiel besteht die Flotte aus einer Replikat-Containergruppe mit mehreren Containern und einer

Daemon-Containergruppe mit einem Container. Bei der Flottenkonfiguration werden drei Kopien der Replikat-Container-Gruppe auf jeder Flotteninstanz gespeichert. Die Daemon-Containergruppe wird niemals repliziert.

Jeder Satz von Replikatgruppen-Containern in der hat drei Kopien auf jeder Instanz. In jedem wichtigen Replikatcontainer wird der Agent angewiesen, zwei Serverprozesse gleichzeitig auszuführen. Daher führt jede Flotteninstanz sechs Serverprozesse gleichzeitig aus (zwei Prozesse in jedem der drei Essential Replica-Container).



Schlüsselkonzepte

In diesem Abschnitt wird zusammengefasst, wie Amazon einige grundlegende Container-Konzepte GameLift implementiert. Anweisungen zur Arbeit mit Containerflotten finden Sie in den entsprechenden Themen in diesem Handbuch.

Verpackung von Containergruppen

Bei der Entwicklung Ihrer Containerstruktur für den Einsatz in einer Containerflotte besteht ein gemeinsames Ziel darin, die verfügbare Rechenleistung optimal zu nutzen. Um dieses Ziel zu erreichen, suchen Sie nach der höchsten Anzahl von Replikat-Containergruppen, die Sie auf einer Flotteninstanz platzieren können, ohne die Leistung des Spieleservers zu beeinträchtigen.

Amazon GameLift kann Ihnen dabei helfen. Es berechnet eine maximale Anzahl von Replikatgruppen pro Instance auf der Grundlage der folgenden Informationen:

- Der Instance-Typ der Flotte und die verfügbaren CPU- und Speicherressourcen.
- Die CPU- und Speicheranforderungen, die Sie für alle Container in Ihrer Replikatgruppe festgelegt haben.

  Die CPU- und Speicheranforderungen, die Sie für alle Container in einer Daemon-Gruppe festlegen, falls es eine gibt.
- Ressourcen, die für die Verwaltung von Containern und anderen kritischen Anwendungen auf jeder Instanz reserviert sind.

Wenn Sie eine Containerflotte erstellen, können Sie wählen, ob Sie die berechnete Höchstzahl verwenden möchten, oder Sie können die berechnete Zahl überschreiben, indem Sie eine gewünschte Zahl angeben. Als bewährte Methode sollten Sie mit Ihrer containerisierten Gameserver-Software experimentieren, um den genauen Ressourcenbedarf zu ermitteln. Verwenden Sie diese Daten, um eine optimale Verpackungsstrategie für die Leistung des Spieleservers zu finden.

Spieleserver und der Amazon GameLift Agent

Wenn Sie Ihren Essential Replica-Container erstellen, packen Sie Ihre Spieleserver-Software und den Amazon GameLift Agent zusammen in demselben Container-Image. Dieser On-Compute-Agent steuert den Lebenszyklus der Spieleserver im Container. In jeder Replikat-Container-Gruppe führt der essentielle Replikat-Container den Agenten und alle Spielserverprozesse aus.

Der GameLift Amazon-Agent führt Anweisungen in der Laufzeitkonfiguration der Containerflotte aus. Die Laufzeitkonfiguration identifiziert (1) die ausführbare Datei, die gestartet werden soll, (2) einen optionalen Satz von Startparametern und (3) die Anzahl der Prozesse, die gleichzeitig ausgeführt werden sollen. Eine Laufzeitkonfiguration kann Anweisungen für mehrere verschiedene ausführbare Dateien enthalten. Mindestens eine Anweisung muss für den Spieleserver ausführbar sein. Eine Laufzeitkonfiguration könnte den Agenten beispielsweise anweisen, 10 Prozesse Ihres Gameservers für den Produktionsgebrauch, einen Prozess derselben ausführbaren Datei mit speziellen Startparametern zum Testen und einen Prozess für ein Protokollierungsprogramm bereitzustellen.

Sie können die Laufzeitkonfiguration einer Flotte jederzeit ändern. Der GameLift Amazon-Agent fordert regelmäßig Updates vom Service an. Wenn eine aktualisierte Laufzeitkonfiguration verfügbar

ist, empfängt der Agent sie und beginnt mit der Implementierung der Anweisungen. Zu den Aktionen können das Hinzufügen oder Herunterfahren von Serverprozessen gehören.

Der Amazon GameLift Agent ist eine Open-Source-Version des On-Compute-Agenten, den Amazon für verwaltete EC2-Flotten GameLift verwendet. Dieses Handbuch enthält Anweisungen dazu, wie Sie den Agenten aus dem Quellcode erstellen und in ein Container-Image integrieren. Der Agent erledigt die folgenden Aufgaben:

Verwaltung von Serverprozessen:

- Serverprozesse auf der Grundlage der Laufzeitkonfiguration starten, herunterfahren und ersetzen.
- Fahren Sie Serverprozesse herunter, wenn sie nicht rechtzeitig aktiviert werden.
- Melden Sie Amazon GameLift , wenn ein Serverprozess beendet wird.
- Sendet Flottenereignisse für Serverprozesse aus.

Verwaltung von Containern:

- Fahren Sie Serverprozesse als Reaktion auf Anweisungen von Amazon GameLift herunter.
- Zustand des Containers melden.

Aufgaben beim Hochladen protokollieren:

- Laden Sie Spielsitzungsprotokolle in einen dafür vorgesehenen Amazon S3 S3-Bucket hoch.
- Laden Sie computergestützte Agentenprotokolle in einen dafür vorgesehenen Amazon S3 S3-Bucket hoch.

Skalieren der Flottenkapazität

Die Flottenkapazität gibt die Anzahl der Spielsitzungen an, die die Flotte gleichzeitig veranstalten kann. Sie können die Kapazität auch anhand der Anzahl der Spieler messen, die die Flotte gleichzeitig unterstützen kann.

Um die Hosting-Kapazität einer Flotte zu erhöhen oder zu verringern, fügen Sie Flotteninstanzen hinzu oder entfernen sie. Die Packstrategie einer Containerflotte bestimmt, wie viele Spielsitzungen gleichzeitig auf jeder Flotteninstanz laufen. Diese Zahl gibt Ihnen die Anzahl der Spielsitzungen (und Spielerplätze) an, die Sie hinzufügen oder abziehen, wenn Sie die Flottenkapazität erhöhen oder verringern.

Bei Containerflotten können Sie jede der von Amazon GameLift bereitgestellten Skalierungsmethoden verwenden. Dazu zählen:

- Stellen Sie die Flottenkapazität manuell ein, indem Sie eine bestimmte gewünschte Anzahl von Flotteninstanzen festlegen.
- Richten Sie die automatische Skalierung ein, indem Sie auf einen gewünschten Puffer verfügbarer Instanzen abzielen (Zielverfolgung). Diese Methode verwaltet automatisch eine Reihe ungenutzter Hosting-Ressourcen, sodass neue Spieler immer schnell mit Spielen beginnen können. Wenn die Nachfrage der Spieler steigt oder sinkt, wird die Größe dieses Puffers kontinuierlich angepasst.
- Richten Sie die automatische Skalierung mit benutzerdefinierten Skalierungsregeln ein (erweiterte Funktion).

Client-/Server-Verbindungen für Spiele

Verwaltete EC2-Flotten und Container-Flotten handhaben Verbindungen zwischen Spieleclients und in der Cloud gehosteten Spieleservern auf ähnliche Weise. Wenn Amazon eine neue Spielsitzung GameLift erstellt, übermittelt der Dienst die Verbindungsinformationen der Spielsitzung. Spieleclients verwenden die Informationen, um sich direkt mit dem Spieleserver zu verbinden, der die Spielsitzung hostet. Für alle Arten von Flotten bestehen die Verbindungsinformationen aus einer IP-Adresse und einer Portzuweisung.

Beim Erstellen einer Containerflotte definieren Sie zwei Gruppen von Portbereichen. Zunächst definieren Sie eine Reihe von nach außen gerichteten Verbindungsports, über die Spieleclients eine Verbindung zu einem Spiel herstellen können. Zweitens definierst du eine Reihe von nur internen Container-Ports, die jedem Spieleserverprozess zugewiesen werden, der im Container läuft. Amazon ordnet die internen Container-Ports GameLift dynamisch externen Verbindungsports zu, um Spielern den Zugriff auf Spiele zu ermöglichen. Dieser Ansatz bietet eine zusätzliche Sicherheitsebene, indem er Ihre Spieleserver vor direktem Zugriff auf die Container-Ports schützt.

Wenn Sie Portbereiche für eine Containerflotte definieren, müssen Sie Bereiche mit ausreichend Ports angeben, um alle Serverprozesse zu unterstützen, die gleichzeitig in den Containern auf einer Instance ausgeführt werden.

Für zusätzliche Kontrolle legen Sie auch Eingangsberechtigungen für eine Flotte fest. Eingehende Berechtigungen bestimmen, welche Verbindungsports für eingehenden Datenverkehr geöffnet sind. Sie können die Berechtigungen einer Flotte für eingehenden Datenverkehr jederzeit ändern. Mit eingehenden Berechtigungen können Sie bei Bedarf schnell alle Verbindungsports herunterfahren, einige oder alle öffnen.

Entwicklungs-Roadmap für Amazon-Container GameLift

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Der folgende Workflow fasst die Schritte zusammen, um Ihre Spieleserver auf einer GameLift Amazon-Containerflotte zum Laufen zu bringen.

Schritt 1: Integriere dein Spiel mit Amazon GameLift

Erweitern Sie Ihren Spieleserver um Funktionen, sodass er mit dem GameLift Amazon-Service kommunizieren kann, wenn er in einer Containerflotte eingesetzt wird. Wenn du FlexMatch Matchmaking verwendest, füge diese Funktion deinem Spieleserver und Client hinzu. Weitere Informationen hierzu finden Sie unter Integriere dein Spiel mit Amazon GameLift.

- Holen Sie sich das Amazon GameLift Server SDK (Version 5+) und richten Sie es mit Ihrem Spieleprojekt ein. Das Server-SDK ist in C++, C# und Go verfügbar.
- Ändern Sie den Code Ihres Spieleservers, um die erforderlichen Server-SDK-Funktionen hinzuzufügen.
- Verpacke deinen Gameserver-Build für Linux. Wenn du unter Windows entwickelst, kann dieser Schritt zusätzliche Arbeit erfordern, um eine Linux-Umgebung einzurichten.
- (Optional) Testen Sie Ihre Gameserver-Integration mit einer GameLift Anywhere Amazon-Flotte. Testen Sie, bevor Sie Ihr Container-Image vorbereiten, um Probleme bei Ihrer Integrationsarbeit zu isolieren. Integrieren Sie auch Ihren Spielclient, um die Verbindungen zwischen Spielclient und Server zu testen.

> **Note**
>
> Wenn du unter Windows entwickelst, richte einen separaten Linux-Arbeitsbereich ein oder verwende ein Tool wie das Windows-Subsystem für Linux (WSL). Du benötigst eine Linux-Umgebung, um deinen Gameserver-Build zu testen und auch um deine Container-Images zu erstellen und zu testen.

Schritt 2: Bereite dein Gameserver-Container-Image vor

Erstellen Sie ein Container-Image, das Ihre Spieleserverprozesse ausführt, und speichern Sie es in einem Amazon Elastic Container Registry (Amazon ECR) -Repository zur Verwendung mit Amazon GameLift. Detaillierte Anweisungen finden Sie unter Bereite ein Container-Image mit deiner Gameserver-Software vor.

- Richten Sie ein Arbeitsverzeichnis für Ihr Container-Image mit Ihrem Linux-Spiel-Build, dem Installationsskript und der gesamten unterstützenden Software und Abhängigkeiten ein.
- Holen Sie sich den Amazon GameLift Agent-Quellcode, erstellen Sie ihn und fügen Sie die `jar` Datei Ihrem Arbeitsverzeichnis hinzu.
- Holen Sie sich das Standard-Dockerfile und ändern Sie es, um ein Container-Image mit Ihrer Gameserver-Software zu konfigurieren.
- Erstellen Sie Ihr Container-Image. Führen Sie diesen Schritt in einer Linux-Umgebung aus.
- Erstellen Sie ein privates Amazon ECR-Repository und übertragen Sie Ihr Container-Image dorthin. Erstellen Sie das Repo dort, AWS-Konto AWS-Region wo Sie Ihre Containerflotte einsetzen möchten.
- (optional) Testen Sie Ihre Container-Images mit Ihrer Anywhere Flotte. Sie können eine Laufzeitkonfiguration einrichten, um Anweisungen an den GameLift Amazon-Agenten weiterzuleiten.

Schritt 3: Erstellen Sie Ihre Container und Containergruppen

Entwerfen Sie eine Container-Architektur für das Hosting von Spielen bei Amazon GameLift. Siehe Entwerfen Sie eine GameLift Amazon-Containerflotte und Containergruppendefinitionen für eine GameLift Amazon-Containerflotte erstellen.

- Definieren Sie Ihre Container-Konfigurationen. Für jeden Container definieren Sie Probleme wie Laufzeitprozesse, Speicherzuweisung, Integritätsprüfungen, Netzwerkports usw.
- Verwenden Sie die GameLift Amazon-Konsole oder AWS CLI, um Container-Gruppendefinitionen mit Ihren Container-Konfigurationen zu erstellen. Wenn Sie eine Container-Gruppendefinition erstellen, erstellt GameLift Amazon zu diesem Zeitpunkt einen Snapshot aller Container-Images.

Schritt 4: Stellen Sie Ihren containerisierten Spieleserver auf einer Containerflotte bereit

Verwenden Sie die im vorherigen Schritt erstellten Containergruppen-Definitionen, um eine Container-Flotte zu erstellen und Ihre containerisierte Gameserver-Software bereitzustellen. Siehe Erstellen Sie eine GameLift Amazon-Containerflotte.

- Verwenden Sie die GameLift Amazon-Konsole oder AWS CLI, um eine Containerflotte zu erstellen.
- Verfolgen Sie den Flottenstatus während der Bereitstellung und Aktivierung von Flotteninstanzen. Überprüfen Sie die Ereignisse bei der Flottenerstellung, um sicherzustellen, dass die Flotte erfolgreich an allen Standorten eingesetzt wird.

- Vergewissere dich, dass Spieleclients Spielsitzungen anfordern und daran teilnehmen und das Spiel spielen können. Wenn du Matchmaking eingerichtet hast, teste diese Szenarien.

Schritt 5: Verwalte deine Flotten

Während du dich auf die Nutzung auf Produktionsebene vorbereitest, baue deine Game-Hosting-Lösung aus und verwalte deinen Hosting-Lebenszyklus.

- Erstelle Flotten mit mehreren Standorten und Flotten an anderen Standorten, um deine Spielerbasis AWS-Regionen zu unterstützen.
- Konfiguriere die Platzierung des Game-Hostings mit Warteschlangen oder Spielersuche. FlexMatch Sehen Sie sich diese Ressourcen an:
  - GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten
  - FlexMatch Entwicklerhandbuch
- Richten Sie eine automatische Skalierung ein, um die Flottenkapazität auf der Grundlage der Nachfrage der Spieler nach Spielsitzungen zu verwalten.
- Richten Sie die Überwachung Ihrer Containerflotten ein. Arbeiten Sie mit GameLift Amazon-Metriken, rufen Sie Spielsitzungs- und Container-Protokolle ab und richten Sie den Fernzugriff auf einzelne Container ein.
- Richten Sie ein langfristiges Management von Containerflotten ein. Verwenden Sie Flottenaliase, um den Prozess der Aktualisierung von Containerflotten zu optimieren. Erstellen Sie AWS CloudFormation Vorlagen zur Verwaltung des Flottenlebenszyklus. Sehen Sie sich diese Ressourcen an:
  - Einer GameLift Amazon-Flotte einen Alias hinzufügen
  - Ressourcen verwalten mit AWS CloudFormation

Integriere dein Spiel mit Amazon GameLift

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Bevor Sie mit Ihrer Gameserver-Software ein Container-Image erstellen und es GameLift für Cloud-Hosting auf Amazon bereitstellen können, integrieren Sie zunächst Ihr Spieleprojekt in das GameLift Amazon-Server-SDK und erstellen Sie einen Spieleserver, der unter Linux ausgeführt werden kann. In diesem Thema werden die verschiedenen Integrationstools vorgestellt, die Amazon GameLift bereitstellt.

Gehostete Spieleserver müssen in der Lage sein, mit dem GameLift Amazon-Service zu kommunizieren. Richten Sie die Kommunikation ein, indem Sie das GameLift Amazon-Server-SDK (Version 5+) zu Ihrem Spieleprojekt hinzufügen und den Servercode Ihres Spiels ändern. Amazon GameLift stellt Server-SDK-Ressourcen und Dokumentation zur Unterstützung mehrerer Sprachen und Game-Engines bereit.

Der Integrationsprozess für containerisierte Spieleserver ist praktisch identisch mit der Integration von Spieleservern für das Hosting auf verwalteten EC2- oder Amazon-Flotten. GameLift Anywhere

Tools für die Integration

Amazon GameLift bietet die folgenden Tools und Sprachunterstützung für die Integration:

Für Unreal Engine-Entwickler

Verwenden Sie das Lightweight-Plugin für Unreal. Dieses Plugin enthält die C++-Server-SDK-Bibliotheken mit den erforderlichen GameLift Amazon-Funktionen. Verwenden Sie die Dokumentation, um Ihr Unreal-Spieleprojekt für das Plugin zu konfigurieren und Ihren Spielcode mit den bereitgestellten Codeblöcken zu aktualisieren, um die erforderliche Funktionalität für Ihre Server- und Client-Builds hinzuzufügen.

- Das SDK-Plugin herunterladen
- Leitfaden: Integrieren Sie Ihr Unreal-Projekt mit Amazon GameLift
- Referenzhandbuch: C++ Server SDK 5 für Unreal

Hinweis: Das GameLift eigenständige Amazon-Plugin für Unreal Engine unterstützt die Verwendung von Containerflotten nicht.

Für Unity-Entwickler

Verwenden Sie das Lightweight-Plugin für Unity. Dieses Plugin enthält die C#-Server-SDK-Bibliotheken mit den erforderlichen GameLift Amazon-Funktionen. Verwenden Sie die Dokumentation, um Ihr Unreal-Spieleprojekt für das Plugin zu konfigurieren und Ihren Spielcode mit den bereitgestellten Codeblöcken zu aktualisieren, um die erforderlichen Funktionen für Ihre Server- und Client-Builds hinzuzufügen.

- Das SDK-Plugin herunterladen
- Leitfaden: Integrieren Sie Ihr Unity-Projekt mit Amazon GameLift

- Referenzhandbuch: C# Server SDK 5 für Unity

Hinweis: Das GameLift eigenständige Amazon-Plugin für Unity unterstützt die Verwendung von Containerflotten nicht.

Für Entwickler, die andere Game-Engines verwenden

Folgen Sie diesen allgemeinen Anweisungen zur Server- und Client-Integration:

- Integrieren Sie einen Spieleserver
- Integrieren Sie einen Spielclient

Amazon GameLift bietet Server-SDK 5-Bibliotheken für die folgenden Sprachen an:

- Server-SDK 5 für C++ [SDK-Download] [Referenzhandbuch]
- Server-SDK 5 für C# [SDK-Download] [Referenzhandbuch]
- Server-SDK 5 für Go [SDK-Download] [Referenzhandbuch]

Baue deinen Spieleserver für Linux

GameLift Amazon-Containerflotten unterstützen Spieleserver, die auf einer Linux-Plattform laufen. Hier sind einige Tipps zum Aufbau Ihres Spieleservers für ein Linux-Ziel:

- Wenn Sie Ihr Spiel mit der Unity-Game-Engine entwickeln, bietet der Spieleditor integrierte Unterstützung ohne besondere Anforderungen für die Entwicklung für Linux.
- Wenn Sie Ihr Spiel in C++ entwickeln, müssen Sie die OpenSSL-Bibliotheken für Linux einbeziehen, wenn Sie das GameLift Amazon-Server-SDK SDK for C++ erstellen und wenn Sie Ihren Spieleserver erstellen. Nehmen Sie dieselben Bibliotheken auch in Ihr Gameserver-Container-Image auf.
- Wenn du dein Spiel mit Unreal Engine unter Windows entwickelst, solltest du die folgenden Optionen in Betracht ziehen:
  - Arbeiten Sie mit Unreal Engine, um eine Cross-Compile-Toolchain einzurichten.
  - Richten Sie einen separaten Linux-Arbeitsbereich ein oder verwenden Sie ein Tool wie das Windows-Subsystem für Linux (WSL). Sie können diese Umgebung verwenden, um den Unreal Editor unter Linux auszuführen, um Ihren Spieleserver zu erstellen.

Testen Sie Ihre Integration lokal

Sie können Ihre Spieleintegration lokal mit einer GameLift Anywhere Amazon-Flotte testen. Dieser Ansatz ist eine bewährte Methode, um Probleme zu isolieren, die in direktem Zusammenhang mit der Integration stehen. Eine Anywhere Flotte ist ein nützliches Tool zum Ausführen von Test-Apps und Spielszenarien, z. B. zum Starten/Stoppen von Spielsitzungen und zum Verfolgen von Spielerverbindungen. Mit einer Anywhere Flotte kannst du viel schneller iterativ bauen und testen, was mehr Einblick in die Hosting-Aktivitäten bietet.

Hilfe Testen Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten zur Verwendung einer GameLift Anywhere Amazon-Flotte für Integrationstests finden Sie unter. Der Arbeitsablauf für die Einrichtung einer Testumgebung sieht wie folgt aus:

1. Richten Sie ein lokales Gerät ein, auf dem Linux ausgeführt wird.
2. Richten Sie eine Anywhere Flotte ein. Erstellen Sie einen benutzerdefinierten Standort für Ihr lokales Gerät, erstellen Sie eine Anywhere Flotte und registrieren Sie dann Ihr lokales Gerät als Computer in der Flotte.
3. Besorgen Sie sich ein Authentifizierungstoken für Ihren Spieleserver. Ihr integrierter Serverprozess benötigt ein Token, um sich beim GameLift Amazon-Service zu authentifizieren. Sie können dasselbe Token für mehrere Serverprozesse wiederverwenden, die gleichzeitig ausgeführt werden. Dieser Schritt ist nur erforderlich, wenn Sie eine Anywhere Flotte für Integrationstests verwenden.

   > **Note**
   >
   > Authentifizierungstoken sind temporär und müssen regelmäßig aktualisiert werden. Erwägen Sie, Ihrem Server-Build-Paket ein Skript hinzuzufügen, um ein neues Token anzufordern.

4. Aktualisiere deinen Spieleservercode fürAnywhere. Wenn der Spielserver auf einer Anywhere-Flotte läuft, muss er die Server-SDK-Aktion `InitSdk()` (C++) (C#) (Unreal) mit den folgenden Serverparametern aufrufen. Dieser Schritt ist nur erforderlich, wenn Sie eine Anywhere Flotte für Integrationstests verwenden. Nachdem Sie den Amazon GameLift Agent zu Ihrem Container-Image hinzugefügt haben, verarbeitet er diese Parameter automatisch.

   Es hat sich bewährt, Ihren Servercode so einzurichten, dass er diese Werte aus Umgebungsvariablen oder aus Konsolenargumenten bezieht, die Sie beim Start angeben.

- `webSocketUrl`— Verwenden Sie den Wert von`GameLiftServiceSdkEndpoint`, der vom Aufruf von zurückgegeben wird`register-compute`.
- `processId`— Weist dem Serverprozess eine eindeutige Kennung zu.
- `fleetId`— Die Anywhere-Flottenkennung, die vom Anruf an zurückgegeben wird`create-fleet`.
- `authToken`— Ein gültiges Authentifizierungstoken, das vom Anruf an zurückgegeben wird`get-compute-auth-token`.

5. Richte auf deinem lokalen Computer die Build-Software für deinen Gameserver ein und starte einen Serverprozess.

   Wenn Ihre Serverintegration erfolgreich ist, ruft der Serverprozess die Server-SDK-Aktion `InitSDK()` auf, um eine Verbindung mit dem GameLift Amazon-Service herzustellen, gefolgt von einem Aufruf, um den Service darüber `ProcessReady()` zu informieren, dass er bereit ist, eine Spielsitzung zu veranstalten.

6. Startet eine Spielsitzung. Wenn du deinen Spielclient integriert hast, um eine Spielsitzung anzufordern, kannst du ihn verwenden, um eine neue Spielsitzung anzufordern. Wenn nicht, verwenden Sie den AWS CLI-Befehl create-game-session. Amazon GameLift erstellt ein `GameSession` Objekt und leitet den Prozess ein, um eine neue Spielsitzung zu starten.

   Wenn Ihre Integration funktioniert, GameLift ruft Amazon einen Serverprozess auf Ihrer lokalen Workstation auf, um eine neue Spielsitzung zu starten (mithilfe des `onStartGameSession()` Callbacks). Wenn eine Spielsitzung für Spieler bereit ist, wird der Serverprozess aufgerufen. `ActivateGameSession()` Als Reaktion darauf GameLift aktualisiert Amazon den `GameSession` Status und die Verbindungsinformationen, sodass ein Spielclient eine Verbindung zur Spielsitzung herstellen und das Spiel spielen kann.

Bereite ein Container-Image mit deiner Gameserver-Software vor

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Der Container ist das grundlegendste Element einer GameLift Amazon-Containerflotte. Ihr Container umfasst Ihren Spieleserver zusammen mit seinen Abhängigkeiten wie SDKs, Software, Verzeichnissen und Dateien.

Um in einer Containerflotte zu funktionieren, muss dein Spieleserver unter Linux laufen und in das Server-SDK 5.x integriert sein.

Themen

- Richte dein Arbeitsverzeichnis ein
- Erstelle dein Container-Image
- Übertragen Sie Ihr Container-Image auf Amazon ECR

Richte dein Arbeitsverzeichnis ein

In Ihrem Arbeitsverzeichnis legen Sie alle Dateien ab, die Sie zum Erstellen Ihres Container-Images benötigen, und definieren, wie Amazon es GameLift ausführt.

Um Ihr Container-Arbeitsverzeichnis einzurichten

1. Erstellen Sie das Verzeichnis, in dem Sie mit Ihren GameLift Amazon-Container-Images arbeiten möchten.

   Example

   Beispielsweise:

   ```
   [~/]$ mkdir -p work/glc/gamebuild && cd work && find .
   .
   ./glc
   ./glc/gamebuild
   ```

2. Klonen Sie den GameLift Amazon-Agenten.

   Example

   Beispielsweise:

   ```
   [~/work]$ git clone https://github.com/aws/amazon-gamelift-agent.git
   Cloning into 'amazon-gamelift-agent'...
   ```

3. Erstellen Sie das GameLiftAgent mit Maven.

Example

Beispielsweise:

```
[~/work]$ cd amazon-gamelift-agent
```

Example

```
[~/work/amazon-gamelift-agent]$ mvn clean compile assembly:single && \
mv target ../glc && cd .. && find glc
```

4. Füge einen Spieleserver hinzu, der in das Server-SDK 5.x integriert, erstellt und in eine `.ZIP` Datei gepackt wurde.
5. Kopiere deine `.ZIP` Datei nach. `~/work/glc/gamebuild/`

Wenn du keinen SDK 5.x-Spieleserver hast, kannst du unser SimpleServerBeispielspiel herunterladen und verwenden, um es mit einer Containerflotte zu versuchen.

Example

```
[~/work]$ curl -o glc/gamebuild/SimpleServer.zip \
'https://ws-assets-prod-iad-r-iad-ed304a55c2ca1aee.s3.us-
east-1.amazonaws.com/086bb355-4fdc-4e63-8ca7-af7cfc45d4f2/
AmazonGameLiftSampleServerBinary.zip' &&
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
Dload  Upload   Total   Spent    Left  Speed
100 5140k  100 5140k    0     0  12.3M      0 --:--:-- --:--:-- --:--:-- 12.3M
glc
glc/target
glc/target/GameLiftAgent-1.0.jar
glc/gamebuild
glc/gamebuild/SimpleServer.zip
```

Erstelle dein Container-Image

Ihr Dockerfile spezifiziert die Umgebung, Software und Anweisungen zum Erstellen Ihres Containers.

Um Ihr Dockerfile zu erstellen

1. Gehe in das Unterverzeichnis`glc`.

   Example

   ```
   [~/work]$ cd glc && find

   .
   ./target
   ./target/GameLiftAgent-1.0.jar
   ./gamebuild
   ```

2. Erstellen und öffnen Sie ein neues Dockerfile.

   Example

   Beispielsweise:

   ```
   [~/work/glc]$ nano Dockerfile
   ```

3. Kopieren Sie aus einer der folgenden Vorlagen und fügen Sie den Inhalt dann in Ihr Dockerfile ein.

Dockerfile-Vorlage für deinen Spieleserver

Diese Vorlage enthält die Mindestanweisungen, die ein Container benötigt, um in einer GameLift Amazon-Flotte verwendet zu werden. Ändern Sie den Inhalt nach Bedarf für Ihren Spieleserver.

```
# Base image
# ----------
  # Add the base image that you want to use over here,
  # Make sure to use an image with the same architecture as the
  # Instance type you are planning to use on your fleets.
  # We require JDK to be installed in the base image, so that
  # it can be used to run the &AGS; Agent
FROM public.ecr.aws/amazoncorretto/amazoncorretto:17-amd64
  #
# Game build directory
# --------------------
  # Add your game build to gamebuild directory and add the zip file name in the
 'GAME_BUILD_ZIP' env variable below.
```

```
  # The game build provided over here needs to be integrated with gamelift server sdk.
  # This template assumes that the game build is in a zip format.
ENV GAME_BUILD_ZIP="<ADD_GAME_BUILD_ZIP_FILE_NAME>" \
  #
# Default directory
# -----------------
  # Default directory, the value provided here should be where the game executable
 exists.
  # Provide this same value as your launch path in RuntimeConfiguration when creating a
 fleet.
  # Ref: https://docs.aws.amazon.com/gamelift/latest/apireference/
API_ServerProcess.html
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME BUILD>" \
HOME_DIR="/local/game" \
  #
  # Registered compute in anywhere fleet (not used in container fleets)
  # ---------------------------------------------------------------------
  # Add the name for the registered compute in an anywhere fleet.
  # This environment variable is required only for anywhere fleets, but not for
 container fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
  GAMELIFT_COMPUTE_NAME="<ADD_COMPUTE_NAME>" \
  #
# Default Gamelift Agent jar
# --------------------------
GAMELIFT_AGENT_EXEC="GameLiftAgent-1.0.jar" \
  #
  # This env variable defines the name of the S3 bucket that stores the GameLift Agent
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
GAMELIFT_AGENT_LOGS_BUCKET_NAME="<ADD NAME OF GAMELIFT AGENT LOGS S3 BUCKET>" \
  #
# -----------
  # This env variable defines the name of the S3 bucket that stores the game session
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
```

```
# -----------
GAME_SESSION_LOGS_BUCKET_NAME="<ADD NAME OF GAME SESSION LOGS S3 BUCKET>" \
#
# -----------
GAMELIFT_AGENT_LOGS_PATH="/local/game/agentlogs/" \
  #
# NOT USED in container fleets - USED in Anywhere fleets
# ------------------------------------------------------------
  # Specifiy the type of compute resource used to host the game servers.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
COMPUTE_TYPE="ANYWHERE" \
  #
  # Specify the credential to be used for creating the client.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
CREDENTIAL_PROVIDER="environment-variable"

USER root

    # intall dependencies as necessary
    RUN yum install -y sudo \
                        unzip \
                        git \
                        shadow-utils \
                        iputils \
                        tar \
                        gcc \
                        make \
                        openssl-devel \
                        zlib-devel \
                        vim \
                        net-tools \
                        nc \
                        procps

    # Set up the ground for 'gamescale' user
    RUN groupadd -r gamescale -g 500 && \
```

```
      useradd -u 500 -r -g gamescale -m -s /sbin/nologin -c "Gamescale user" gamescale
&& \
      echo "gamescale ALL=(ALL) NOPASSWD: ALL" | (EDITOR="tee -a" visudo) && \
      mkdir -p $HOME_DIR && \
      mkdir $HOME_DIR/mono && \
      chown -R gamescale:gamescale $HOME_DIR

   WORKDIR $HOME_DIR

   # extract game build as necessary
   COPY ./gamebuild/$GAME_BUILD_ZIP .
   RUN unzip ./$GAME_BUILD_ZIP -d ./

   # copy Gamelift Agent jar
   COPY ./gameliftAgent/$GAMELIFT_AGENT_EXEC ./

   # Add permissions to game build and gamelift agent jar
   RUN chmod +x ./$GAME_EXECUTABLE
   RUN chmod +x ./$GAMELIFT_AGENT_EXEC

   # Check if java is installed on the image, if not then the Agent will not be able
to run
   RUN java --version

   USER gamescale

   ENV PATH="$PATH:$HOME_DIR/bin:$JAVA_HOME"

   # Change directory to bin
   WORKDIR $HOME_DIR

   # check path before starting the container
   RUN echo $PATH

   # Create logs directory for GameLift Agent & server processes
   RUN mkdir logs
   RUN mkdir agentlogs

   # Start the GameLift Agent
   ENTRYPOINT sleep 90 && java -jar $GAMELIFT_AGENT_EXEC -ip "192.168.1.1" -gslb
"$GAME_SESSION_LOGS_BUCKET_NAME" -galb "$GAMELIFT_AGENT_LOGS_BUCKET_NAME" -galp
"$GAMELIFT_AGENT_LOGS_PATH" -glc environment-variable
```

Dockerfile für das Beispiel **`SimpleServer`**

```
# Base image
# ----------
  # Add the base image that you want to use over here,
  # Make sure to use an image with the same architecture as the
  # Instance type you are planning to use on your fleets.
  # We require JDK to be installed in the base image, so that
  # it can be used to run the &AGS; Agent
FROM public.ecr.aws/amazoncorretto/amazoncorretto:17-amd64
  #
# Game build directory
# --------------------
  # Add your game build to gamebuild directory and add the zip file name in the
 'GAME_BUILD_ZIP' env variable below.
  # The game build provided over here needs to be integrated with gamelift server sdk.
  # This template assumes that the game build is in a zip format.
ENV GAME_BUILD_ZIP="SimpleServer.zip" \
  #
# Default directory
# -----------------
  # Default directory, the value provided here should be where the game executable
 exists.
  # Provide this same value as your launch path in RuntimeConfiguration when creating a
 fleet.
  # Ref: https://docs.aws.amazon.com/gamelift/latest/apireference/
API_ServerProcess.html
GAME_EXECUTABLE="GameLiftSampleServer" \
HOME_DIR="/local/game" \
  #
  # Registered compute in anywhere fleet (not used in container fleets)
  # ------------------------------------------------------------------
  # Add the name for the registered compute in an anywhere fleet.
  # This environment variable is required only for anywhere fleets, but not for
 container fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
  GAMELIFT_COMPUTE_NAME="<ADD_COMPUTE_NAME>" \
  #
# Default Gamelift Agent jar
# --------------------------
GAMELIFT_AGENT_EXEC="GameLiftAgent-1.0.jar" \
  #
  # This env variable defines the name of the S3 bucket that stores the GameLift Agent
 logs.
```

```
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
GAMELIFT_AGENT_LOGS_BUCKET_NAME="<ADD NAME OF GAMELIFT AGENT LOGS S3 BUCKET>" \
  #
# -----------
  # This env variable defines the name of the S3 bucket that stores the game session
 logs.
  # This S3 bucket should exist in the customer AWS account.
  # In order to allow GameLift agent to upload logs to this s3 bucket, customers would
 need to
  # include s3:PutObject permission in the IAM role provided as instanceRoleArn during
 CreateFleet operation.
# -----------
GAME_SESSION_LOGS_BUCKET_NAME="<ADD NAME OF GAME SESSION LOGS S3 BUCKET>" \
#
# -----------
GAMELIFT_AGENT_LOGS_PATH="/local/game/agentlogs/" \
  #
# NOT USED in container fleets - USED in Anywhere fleets
# ------------------------------------------------------------
  # Specifiy the type of compute resource used to host the game servers.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
COMPUTE_TYPE="ANYWHERE" \
  #
  # Specify the credential to be used for creating the client.
  # This env variable is required only for anywhere fleets, but not for container
 fleets.
  # If it is set for container fleets, it will be overridden by Gamelift.
#
# -----------
CREDENTIAL_PROVIDER="environment-variable"

USER root

    # intall dependencies as necessary
    RUN yum install -y sudo \
                        unzip \
```

```
                        git \
                        shadow-utils \
                        iputils \
                        tar \
                        gcc \
                        make \
                        openssl-devel \
                        zlib-devel \
                        vim \
                        net-tools \
                        nc \
                        procps

   # Set up the ground for 'gamescale' user
   RUN groupadd -r gamescale -g 500 && \
     useradd -u 500 -r -g gamescale -m -s /sbin/nologin -c "Gamescale user" gamescale
&& \
     echo "gamescale ALL=(ALL) NOPASSWD: ALL" | (EDITOR="tee -a" visudo) && \
     mkdir -p $HOME_DIR && \
     mkdir $HOME_DIR/mono && \
     chown -R gamescale:gamescale $HOME_DIR

   WORKDIR $HOME_DIR

   # extract game build as necessary
   COPY ./gamebuild/$GAME_BUILD_ZIP .
   RUN unzip ./$GAME_BUILD_ZIP -d ./

   # copy Gamelift Agent jar
   COPY ./target/$GAMELIFT_AGENT_EXEC ./

   # Add permissions to game build and gamelift agent jar
   RUN chmod +x ./$GAME_EXECUTABLE
   RUN chmod +x ./$GAMELIFT_AGENT_EXEC

   # Check if java is installed on the image, if not then the Agent will not be able
to run
   RUN java --version

   USER gamescale

   ENV PATH "$PATH:$HOME_DIR/bin:$JAVA_HOME"

   # Change directory to bin
```

```
    WORKDIR $HOME_DIR

    # check path before starting the container
    RUN echo $PATH

    # Create logs directory for GameLift Agent & server processes
    RUN mkdir logs
    RUN mkdir agentlogs

    # Start the GameLift Agent
    ENTRYPOINT sleep 90 && java -jar $GAMELIFT_AGENT_EXEC -ip "192.168.1.1" -gslb
"$GAME_SESSION_LOGS_BUCKET_NAME" -galb "$GAMELIFT_AGENT_LOGS_BUCKET_NAME" -galp
"$GAMELIFT_AGENT_LOGS_PATH" -glc environment-variable
```

> **Note**
>
> Hinweis: Einige der Umgebungsvariablen in der Dockerfile können durch die überschrieben werden. ContainerDefinition

Um Ihr Container-Image zu erstellen

1. Erstellen Sie Ihr Container-Image.

   Wenn Sie Ihren eigenen SDK 5.x-Server verwenden

   Sie können einen beliebigen lokalen Repository-Namen angeben.

   Example

   ```
   [~/work/glc]$ docker build -t <local repository name>:<optional tag> .
   ```

   Wenn Sie unser **SimpleServer** Beispiel verwenden

   Example

   ```
   [~/work/glc]$ docker build -t simple-server:version-1 .
   Successfully built 0123456789012
   Successfully tagged simple-server:version-1
   ```

> **Note**
>
> In den folgenden Beispielen verwenden wir *simpler-server* als REPOSITORY Anfangswert und version-1 als Wert. TAG

2. Sehen Sie sich die Liste der Bilder an und notieren Sie sich die Werte REPOSITORY und IMAGE ID. Sie benötigen sie in einem der folgenden Verfahren.

   Example

```
[~/work/glc]$ docker images
REPOSITORY       TAG          IMAGE ID        CREATED           SIZE
simple-server    version-1    0123456789012    14 minutes ago    1.24GB
```

Übertragen Sie Ihr Container-Image auf Amazon ECR

Laden Sie Ihr Container-Image in ein privates Repository in Amazon ECR hoch. Wenn Sie eine Container-Gruppendefinition erstellen, verweisen Sie auf diesen Repository-Speicherort, GameLift sodass Amazon einen Snapshot Ihres Container-Images erstellen und ihn bei der Bereitstellung einer Containerflotte verwenden kann.

> **Note**
>
> Wenn Sie noch kein privates Amazon ECR-Repository haben, erstellen Sie eines.

Um Ihre Amazon ECR-Anmeldeinformationen zu erhalten

- Bevor Sie Ihr Container-Image auf Amazon ECR übertragen können, müssen Sie Ihre AWS Anmeldeinformationen in temporärer Form abrufen und sie Docker zur Verfügung stellen. Besorgen Sie sich Ihre Amazon ECR-Anmeldeinformationen, damit sich Docker anmelden kann.

  Example

```
[~/work/glc]$ aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
WARNING! Your password will be stored unencrypted in
```

```
/home/user-name/.docker/config.json.
Configure a credential helper to remove this warning.
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Um Ihr Container-Image an Amazon ECR zu übertragen

1. Kopieren Sie die URI des privaten Amazon ECR-Repositorys, das Sie verwenden möchten.
2. Wenden Sie ein Amazon ECR-Tag auf Ihr Container-Image an.

   Example

   ```
   [~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository
    URI>:<optional tag>
   ```
3. Übertragen Sie Ihr Container-Image auf Amazon ECR

   Example

   ```
   [~/work/glc]$ docker image push  <Amazon ECR private repository URI>
   ```

Entwerfen Sie eine GameLift Amazon-Containerflotte

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

In diesen Themen werden die wichtigsten Entscheidungen vorgestellt, die Sie beim Aufbau einer GameLift Amazon-Containerflotte treffen werden. Ihre Entscheidungen wirken sich darauf aus, wie Sie die Einstellungen für Container, Containergruppen und Flotten konfigurieren.

Themen

- Entwerfen Sie die Containerstruktur Ihrer Flotte
- Legen Sie Ressourcenlimits fest
- Benennen Sie wichtige Container
- Netzwerkverbindungen konfigurieren
- Richten Sie Zustandsprüfungen für Container ein

- Legen Sie Container-Abhängigkeiten fest
- Konfigurieren Sie eine Containerflotte

Entwerfen Sie die Containerstruktur Ihrer Flotte

Identifizieren Sie als ersten Schritt die Software und die Ressourcen, die zum Hosten Ihres Spieleservers benötigt werden, einschließlich der folgenden:

- Deine Gameserver-Anwendung. Die Anwendung muss in die GameLift Amazon-Funktionalität für das Hosting integriert sein, einschließlich des Server-SDK Version 5+. Siehe Integriere dein Spiel mit Amazon GameLift.
- Der GameLift Amazon-Agent. Dieser On-Compute-Agent unterhält die Kommunikation mit dem GameLift Amazon-Service und verwaltet den Lebenszyklus aller Spieleserverprozesse. Weitere Details finden Sie unter Spieleserver und der Amazon GameLift Agent.
- Zusätzliche Software und Ressourcen nach Bedarf. Dies kann Software beinhalten, die zum Ausführen deiner Gameserver-Anwendungen benötigt wird. Für die Protokollierung und Überwachung, Sicherheit, Inhaltsbereitstellung und Datensynchronisierung wird gängige unterstützende Software verwendet.

Entscheiden Sie als Nächstes, wie Sie Ihre Software und Ressourcen für eine GameLift Amazon-Containerflotte strukturieren möchten. Amazon GameLift verwendet Containergruppen, um Container zu organisieren. Eine Flotte hat immer eine Replikat-Containergruppe und kann optional über eine Daemon-Containerflotte verfügen. Weitere Details finden Sie unter Komponenten der Containerflotte.

- Beginnen Sie mit dem Entwerfen Ihrer Replikat-Containergruppe. Berücksichtigen Sie die folgenden Hinweise:
  - Bündeln Sie Ihre Gameserver-Anwendung und den Amazon GameLift Agent in demselben Container. Machen Sie diesen Container zum einzigen wichtigen Container der Replikatgruppe.
  - Organisieren Sie alle andere Software für Ihren Spieleserver in Containern. Sie können sich dafür entscheiden, alles in einem einzigen Container in der Replikatgruppe abzulegen. Sie können sich auch dafür entscheiden, einen oder mehrere Sidecar-Container zu erstellen. Zu den Gründen für die Verwendung von Beiwagen gehören:
    - Um eine Start-/Abschaltsequenz für einzelne Software einzurichten. Sie können dies erreichen, indem Sie Software in separaten Containern platzieren und Abhängigkeiten zwischen diesen einrichten.

    - Um containerspezifische Grenzwerte für die Speicher- und CPU-Auslastung festzulegen.
    - Um unterschiedliche Container-Konfigurationseinstellungen für jeden Container anzugeben, z. B. einen Startbefehl, einen Einstiegspunkt, ein Arbeitsverzeichnis, Umgebungsvariablen oder Integritätsprüfungen.
- Entscheiden Sie, ob Sie eine Daemon-Containergruppe für Ihre Flotte benötigen. Berücksichtigen Sie dabei Folgendes:
    - Daemon-Container werden in der Regel zur Ausführung von Hintergrund- oder Überwachungsprozessen verwendet.
    - Container in einer Daemon-Gruppe werden nicht auf einer Flotteninstanz repliziert. Das bedeutet, dass Container in einer Daemon-Gruppe nicht zusammen mit der Replikat-Containergruppe skaliert werden.
    - Eine Daemon-Gruppe kann mehrere Container haben. Sie können jeden Container in einer Daemon-Gruppe als essenziell kennzeichnen.

Legen Sie Ressourcenlimits fest

Ermitteln Sie für jede Containergruppe, wie viel Arbeitsspeicher und CPU die Gruppe benötigt, um ihre Software auszuführen. Amazon GameLift stützt sich auf diese Informationen, um die Ressourcen für die Containergruppe zu verwalten. Es verwendet diese Informationen auch, um zu berechnen, wie viele Replikat-Containergruppen ein Flottenimage aufnehmen kann. Sie können auch Grenzwerte für einzelne Container festlegen.

Legen Sie optionale Grenzwerte für Container fest

Durch die Festlegung containerspezifischer Ressourcenlimits können Sie besser kontrollieren, wie einzelne Container die Ressourcen der Gruppe nutzen können. Wenn Sie keine containerspezifischen Grenzwerte festlegen, teilen sich alle Container in der Gruppe die Gruppenressourcen. Die gemeinsame Nutzung bietet mehr Flexibilität, um Ressourcen dort einzusetzen, wo sie benötigt werden. Es erhöht auch das Potenzial, dass Prozesse miteinander konkurrieren und zum Ausfall von Containern führen.

Legen Sie für jeden Container eine der folgenden `ContainerDefinition` Eigenschaften fest.

- `SoftLimit`(Speicher) — Reservieren Sie eine Mindestmenge an Speicher für die ausschließliche Verwendung des Containers. Dem Container steht immer die reservierte Menge zur Verfügung. Dieses Minimum kann jederzeit überschritten werden, sofern zusätzliche Ressourcen verfügbar sind.

- `HardLimit`(Speicher) — Legen Sie ein maximales Speicherlimit für den Container fest. Wenn der Container dieses Limit überschreitet, führt dies zu einem Neustart.
- `CpuLimit` — Reservieren Sie eine Mindestmenge an CPU-Ressourcen für die ausschließliche Verwendung des Containers. Dem Container steht immer die reservierte Menge zur Verfügung. Dieses Minimum kann jederzeit überschritten werden, sofern zusätzliche Ressourcen verfügbar sind. (1024 CPU-Einheiten entsprechen 1 vCPU.)

Legen Sie die Gesamtressourcenlimits für eine Containergruppe fest

Teilen Sie Amazon mit, GameLift wie viel Arbeitsspeicher und CPU-Ressourcen jede Containergruppe benötigt. Ziel ist es, genügend Ressourcen zuzuweisen, um die Leistung des Spieleservers zu optimieren. Amazon GameLift verwendet diese Grenzwerte, um zu berechnen, wie Replikat-Containergruppen auf einer Flotteninstanz gepackt werden. Sie werden sie auch verwenden, wenn Sie einen Instance-Typ für eine Containerflotte auswählen.

Berechnet den gesamten Arbeitsspeicher und die CPU, die für alle Prozesse in jedem Container in einer Gruppe benötigt werden. Berücksichtigen Sie dabei Folgendes:

- Welche Prozesse werden in allen Containern der Containergruppe ausgeführt? Addieren Sie die für diese Prozesse benötigten Ressourcen.
- Wie viele Gameserver-Prozesse möchten Sie in jeder Containergruppe gleichzeitig ausführen? Sie legen diesen Wert als Teil der Laufzeitkonfiguration einer Flotte fest, aber Sie müssen hier genügend Speicher für sie einplanen (siehe Optimieren Sie Ihre Laufzeitkonfiguration).

Legen Sie auf der Grundlage Ihrer Schätzung der Anforderungen an Containergruppen die folgenden `ContainerGroupDefinition` Eigenschaften fest:

- `TotalMemoryLimit`— Legen Sie ein maximales Speicherlimit für die Containergruppe fest. Alle Container in der Gruppe teilen sich den zugewiesenen Speicher. Wenn Sie Grenzwerte für einzelne Container festlegen, muss das Gesamtspeicherlimit wie folgt lauten:
  - gleich oder größer als die Summe aller Soft-Memory-Limits für Container
  - gleich oder größer als das höchste Hard-Memory-Limit für einen Container in der Gruppe
- `TotalCpuLimit` — Legen Sie ein maximales CPU-Limit für die Containergruppe fest. Alle Container in der Gruppe teilen sich die zugewiesenen CPU-Ressourcen. Wenn Sie Grenzwerte für einzelne Container festlegen, muss das gesamte CPU-Limit wie folgt lauten:
  - gleich oder größer als die Summe aller Container-CPU-Grenzwerte. Es hat sich bewährt, diesen Wert auf das Doppelte der Summe der Container-CPU-Grenzwerte festzulegen.

Beispielszenario

Nehmen wir an, wir definieren eine Replikatcontainergruppe mit den folgenden drei Containern:

- Container A ist unser unverzichtbarer Replikat-Container. Es führt Spielserverprozesse und den Amazon GameLift Agent aus. Wir schätzen den Ressourcenbedarf für einen Spieleserver auf 512 MiB und 1024 CPU. Wir planen, den Container 10 Serverprozesse ausführen zu lassen. Da auf diesem Container unsere kritischste Software ausgeführt wird, haben wir eine weiche Speicherreserve von 6144 MiB und kein Hard-Memory-Limit oder CPU-Reservelimit festgelegt.
- In Container B wird unterstützende Software mit geschätzten Ressourcenanforderungen von 1024 MiB und 1536 CPUs ausgeführt. Wir haben ein Soft-Memory-Reserve-Limit von 1024 MiB, ein Hard-Memory-Limit von 2048 MiB und ein CPU-Reservelimit von 1024 CPU festgelegt.
- Container C führt unkritische Protokollierungs- und andere Überwachungsprogramme aus. Wir haben ein Festplattenlimit von 512 MiB und ein CPU-Reservelimit von 512 CPU festgelegt.

Anhand dieser Informationen haben wir die folgenden Gesamtgrenzen für die Containergruppe festgelegt:

- Gesamtspeicherlimit: 7680 MiB. Dieser Wert überschreitet (1) die Summe der Soft-Memory-Limits (6144+1024 MiB) und (2) die höchste Hard-Memory-Grenze (1024 MiB).
- Gesamtes CPU-Limit: 13312 CPU. Dieser Wert überschreitet die Summe des CPU-Limits (1024+512 CPU).

Benennen Sie wichtige Container

Geben Sie für jeden Behälter den Behälter als unverzichtbar oder nicht unbedingt erforderlich an. Alle Containergruppen müssen mindestens einen Behälter für wichtige Zwecke haben. Der Essential-Container erledigt die wichtige Arbeit der Container-Gruppe, z. B. das Hosten Ihrer Spieleserver. Es wird erwartet, dass der Essential-Container immer läuft. Wenn dies fehlschlägt, wird die gesamte Containergruppe neu gestartet.

- Die Replikatcontainergruppe Ihrer Flotte kann genau einen wichtigen Container enthalten. In diesem Container laufen der Amazon GameLift Agent und die von ihm verwalteten Spieleserverprozesse.
- Wenn Ihre Flotte über eine Daemon-Containergruppe verfügt, können Sie mehrere wichtige Container festlegen. Machen Sie einen Daemon-Container unverzichtbar, wenn Sie möchten, dass ein Container-Fehler einen Neustart der Containergruppe veranlasst.

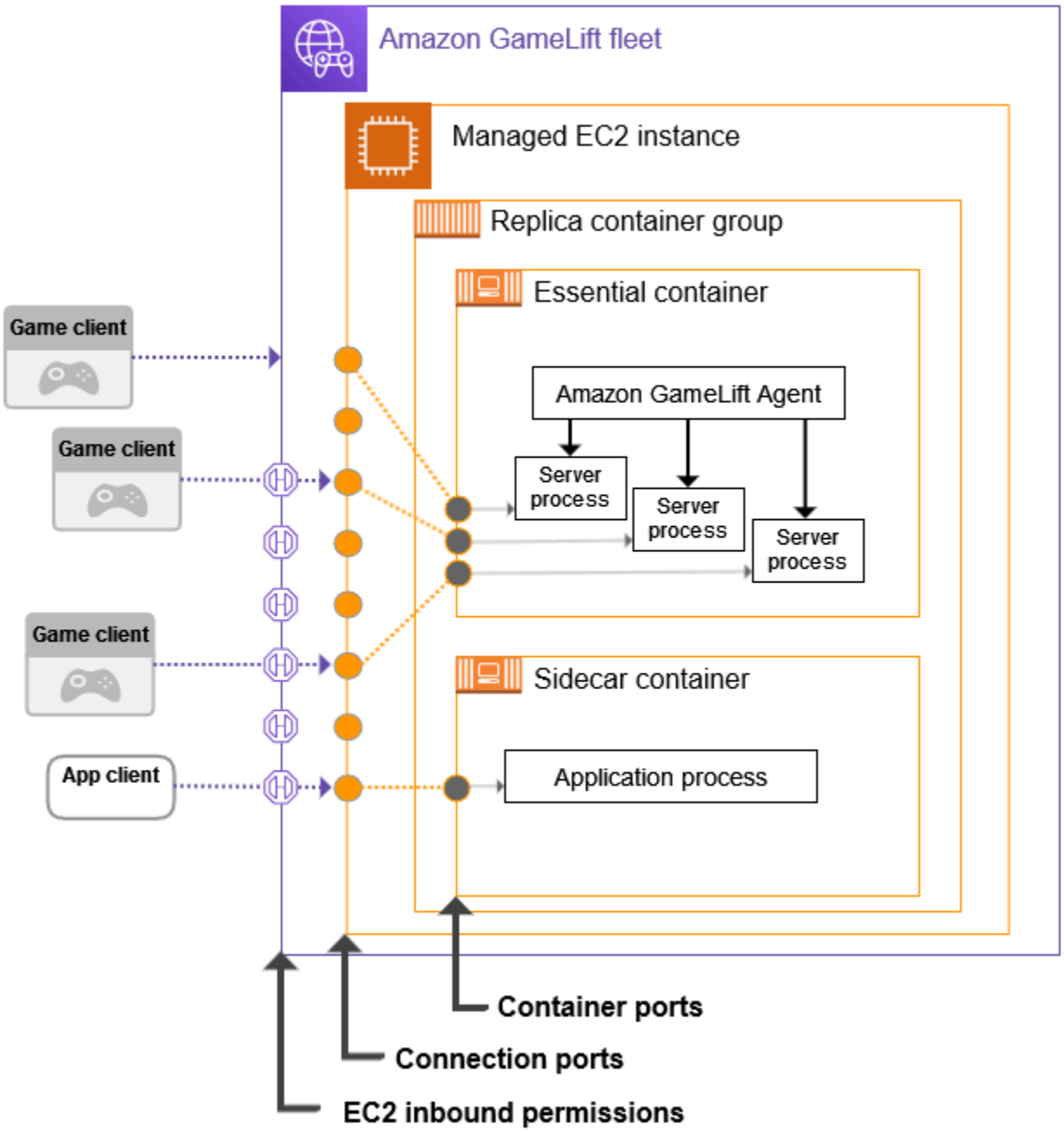
Setzen Sie `Essential` die `ContainerDefinition` Eigenschaft für jeden Container entweder auf true oder false.

Netzwerkverbindungen konfigurieren

Sie können einen Netzwerkzugriff einrichten, damit externer Verkehr eine Verbindung zu jedem Container in einer Containerflotte herstellen kann. Sie müssen beispielsweise Netzwerkverbindungen zu dem Container herstellen, auf dem Ihre Spieleserverprozesse ausgeführt werden, damit Spieleclients Ihrem Spiel beitreten und es spielen können. Spieleclients stellen über Ports und IP-Adressen eine Verbindung zu Spieleservern her.

In einer Containerflotte besteht keine direkte Verbindung zwischen einem Client und einem Server. Intern hört ein Prozess in einem Container auf einem Container-Port ab. Extern stellt der eingehende Verkehr über einen Verbindungsport eine Verbindung zu einer Flotteninstanz her. Amazon GameLift verwaltet die Zuordnungen zwischen internen Container-Ports und nach außen gerichteten Verbindungsports, sodass eingehender Datenverkehr an den richtigen Prozess auf der Instance weitergeleitet wird.

Amazon GameLift bietet eine zusätzliche Kontrollebene für Ihre Netzwerkverbindungen. Jede Containerflotte verfügt über eine Einstellung für Eingangsberechtigungen, mit der Sie den Zugriff auf jeden nach außen gerichteten Verbindungsport kontrollieren können. Sie können die Portkonfigurationen einer vorhandenen Flotte nicht ändern, aber Sie können den Zugriff nach Bedarf zulassen oder einschränken, indem Sie die Eingangsberechtigungen anpassen. Sie können beispielsweise die Berechtigungen für alle Verbindungsports entfernen, um den gesamten Zugriff auf die Container der Flotte zu unterbinden.



Legen Sie die Portbereiche für Container fest

Konfigurieren Sie eine Containerdefinition mit ausreichend Container-Ports für jeden Prozess, der externen Zugriff benötigt. Einige Container benötigen keine Ports. Andere müssen über genügend Ports verfügen, um jedem Prozess, der einen benötigt, einen zuzuweisen.

Ihre essenzielle Replica-Containergruppe, die Ihre Spieleserver betreibt, benötigt einen Port für jeden gleichzeitig laufenden Spieleserver-Prozess (wie in der Flotte konfiguriert). `RuntimeConfiguration` Der Spieleserver-Prozess überwacht den zugewiesenen Port und meldet ihn an Amazon GameLift.

Wenn Sie eine Container-Gruppendefinition erstellen, definieren Sie einen Container-Port-Bereich für jeden Container, der Netzwerkzugriff benötigt (siehe ContainerDefinitionInput: PortConfiguration). Stellen Sie sicher, dass der Bereich groß genug ist, um jedem Prozess, der einen benötigt, einen Port zuzuweisen. Prozessen müssen in der Portkonfiguration des Containers Portnummern zugewiesen werden.

Legen Sie die Portbereiche für die Verbindung fest

Konfigurieren Sie Ihre Containerflotte mit einer Reihe von Verbindungsanschlüssen. Verbindungsports bieten externen Zugriff auf die Flotteninstanzen, auf denen Ihre Container laufen. Amazon GameLift weist Verbindungsports zu und ordnet sie nach Bedarf Container-Ports zu.

Wenn Sie eine Containerflotte erstellen, definieren Sie einen Verbindungsportbereich (siehe ContainerGroupsConfiguration: ConnectionPortRange). Stellen Sie sicher, dass der Bereich über genügend Ports verfügt, um jedem Containerhafen in einer Flotteninstanz zugeordnet zu werden. Verwenden Sie die folgende Formel, um die mindestens erforderlichen Verbindungsports zu berechnen:

```
[Total number of container ports defined for containers in the replica container group] * [Number of replica container groups per instance] + [Total number of container ports defined for containers in the daemon container group]
```

Es hat sich bewährt, die Mindestanzahl an Verbindungsanschlüssen zu verdoppeln.

> **Note**
>
> Die Anzahl der Verbindungsports kann möglicherweise die Anzahl der Replikat-Containergruppen pro Instanz einschränken. Wenn eine Flotte nur über genügend Verbindungsports für eine Replikat-Container-Gruppe pro Instance verfügt, GameLift stellt Amazon nur eine Replikat-Container-Gruppe bereit, auch wenn die Instances über genügend Rechenleistung für mehrere Replikat-Container-Gruppen verfügen.

Legen Sie Berechtigungen für eingehende Nachrichten fest

Eingehende Berechtigungen steuern den externen Zugriff auf eine Containerflotte, indem sie angeben, welche Verbindungsports für eingehenden Verkehr geöffnet werden sollen. Sie können diese Einstellung verwenden, um den Netzwerkzugriff einer Flotte nach Bedarf ein- und auszuschalten.

Wenn Sie eine Containerflotte erstellen, definieren Sie eine Reihe von Berechtigungen für eingehenden Datenverkehr (CreateFleetsehe:EC2). InboundPermissions Legen Sie die Port-Eigenschaften für eingehende Zugriffe so fest, dass sie einige oder alle Werte in den Verbindungs-Port-Einstellungen der Flotte enthalten. Rufen Sie an, um die Zugriffsberechtigungen für eingehende Sendungen für eine bestehende Containerflotte zu ändern. UpdateFleetPortSettings

Beispielszenario

Dieses Beispiel zeigt, wie alle drei Netzwerkverbindungseigenschaften festgelegt werden.

- Die Replikatcontainergruppe unserer Flotte besteht aus einem Container, in dem die Spielserverprozesse ausgeführt werden. Die Laufzeitbestätigung weist den Container an, 10 Gameserver-Prozesse gleichzeitig auszuführen.

  In der Definition der Replikat-Container-Gruppe legen wir den `PortConfiguration` Parameter für diesen Container wie folgt fest:

  ```
  "PortConfiguration": {
    "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP"} ]
    }
  ```

- Unsere Flotte hat auch eine Daemon-Containergruppe mit einem Container. Es hat 1 Prozess, der Netzwerkzugriff benötigt. In der Definition der Daemon-Container-Gruppe legen wir den `PortConfiguration` Parameter für diesen Container wie folgt fest:

  ```
  "PortConfiguration": {
    "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ]  }
  ```

- Unsere Flotte ist mit 3 Replikat-Containergruppen pro Flotteninstanz konfiguriert. Anhand dieser Informationen können wir die Formel verwenden, um die Anzahl der benötigten Verbindungsanschlüsse zu berechnen:
  - Minimum: 31 Ports [10 Replikat-Container-Ports * 3 Replikat-Containergruppen pro Instanz + 1 Daemon-Container-Port]

- Bewährtes Verfahren: 62 Anschlüsse [mindestens 2 Anschlüsse]

Bei der Erstellung der Containerflotte haben wir den `ConnectionPortRange` Parameter `ContainerGroupsConfiguration` wie folgt festgelegt:

```
"ConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- Wir möchten den Zugriff auf alle verfügbaren Verbindungsports ermöglichen. Bei der Erstellung der Containerflotte haben wir den `EC2InboundPermissions` Parameter wie folgt festgelegt:

```
"EC2InboundPermissions": [
  {"FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":
 "TCP"} ]
```

Richten Sie Zustandsprüfungen für Container ein

Ein Container wird automatisch neu gestartet, wenn ein Terminalfehler auftritt, und er wird nicht mehr ausgeführt. Wenn der Container unverzichtbar ist, wird die gesamte Containergruppe neu gestartet.

Sie können zusätzliche benutzerdefinierte Kriterien definieren, um den Zustand des Containers zu messen, und diese Kriterien mit einer Integritätsprüfung testen. Um eine Container-Integritätsprüfung einzurichten, können Sie sie in einem Docker-Container-Image oder in Ihrer Container-Definition definieren. Wenn Sie in der Container-Definition eine Integritätsprüfung einrichten, überschreibt diese alle Einstellungen im Container-Image.

Legen Sie optionale Zustandsprüfungen basierend auf dem Containertyp wie folgt fest:

- Konfigurieren Sie für einen Container mit essenziellen Replikaten keine Integritätsprüfungen. Der GameLift Amazon-Agent kümmert sich automatisch um die Zustandsberichte für diesen Container.
- Für Replikat-Container, die nicht unbedingt erforderlich sind, und für alle Daemon-Container können Sie optional Parameter für die Integritätsprüfung festlegen.

Legen Sie die folgenden `ContainerDefinition` Eigenschaften für eine Container-Zustandsprüfung fest:

- `Command`— Stellen Sie einen Befehl bereit, der einen Aspekt des Zustands des Containers überprüft. Sie entscheiden, nach welchen Kriterien der Gesundheitszustand gemessen werden soll. Der Befehl muss zu einem Ausgangswert von 1 (fehlerhaft) oder 0 (fehlerhaft) führen.

- `StartPeriod`— Geben Sie eine anfängliche Verzögerung an, bis Fehler bei der Integritätsprüfung gezählt werden. Diese Verzögerung gibt dem Container Zeit, seine Prozesse zu booten.
- `Interval`— Entscheiden Sie, wie oft der Health Check-Befehl ausgeführt werden soll. Wie schnell möchten Sie einen Container-Fehler erkennen und beheben?
- `Timeout`— Entscheiden Sie, wie lange Sie auf Erfolg oder Misserfolg warten möchten, bevor Sie den Befehl zur Integritätsprüfung erneut ausführen. Wie lange sollte es dauern, bis der Befehl zur Integritätsprüfung abgeschlossen ist?
- `Retries`— Wie oft sollte der Health-Check-Befehl wiederholt werden, bevor ein Fehler registriert wird?

Legen Sie Container-Abhängigkeiten fest

Innerhalb jeder Containergruppe können Sie Abhängigkeiten zwischen Containern auf der Grundlage des Containerstatus festlegen. Eine Abhängigkeit wirkt sich darauf aus, wann der abhängige Container je nach Status eines anderen Containers gestartet oder heruntergefahren werden kann.

Ein wichtiger Anwendungsfall für Abhängigkeiten ist die Erstellung von Start- und Shutdown-Sequenzen für die Container-Gruppe.

Sie möchten beispielsweise, dass Container A zuerst gestartet und erfolgreich abgeschlossen wird, bevor Container B und C gestartet werden. Um dies zu erreichen, erstellen Sie zunächst eine Abhängigkeit für Container B von Container A mit der Bedingung, dass Container A erfolgreich abgeschlossen werden muss. Erstellen Sie dann eine Abhängigkeit für Container C von Container A mit derselben Bedingung. Die Startsequenzen werden beim Herunterfahren in umgekehrter Reihenfolge ausgeführt.

Konfigurieren Sie eine Containerflotte

Wenn Sie eine Containerflotte erstellen, sollten Sie die folgenden Entscheidungspunkte berücksichtigen. Die meisten dieser Punkte hängen von Ihrer Container-Architektur und -Konfiguration ab.

Entscheiden Sie, wo Sie Ihre Flotte einsetzen möchten

> Im Allgemeinen möchten Sie Ihre Flotten geografisch in der Nähe Ihrer Spieler einsetzen, um die Latenz zu minimieren. Sie können Ihre Containerflotte in jedem Container einsetzen AWS-Region , den Amazon GameLift unterstützt. Wenn du denselben Spielserver an weiteren geografischen Standorten einsetzen möchtest, kannst du der Flotte entfernte Standorte

hinzufügen, einschließlich AWS-Regionen Local Zones. Bei einer Flotte mit mehreren Standorten kannst du die Kapazität an jedem Flottenstandort unabhängig anpassen. Weitere Informationen zu unterstützten Flottenstandorten finden Sie unterAmazon GameLift -Hosting-Standorte.

Wählen Sie einen Instance-Typ und eine Instance-Größe für Ihre Flotte

Amazon GameLift unterstützt eine Vielzahl von Amazon EC2 EC2-Instance-Typen, die alle für die Verwendung mit einer Containerflotte verfügbar sind. Die Verfügbarkeit und der Preis des Instance-Typs variieren je nach Standort. Eine nach Standort gefilterte Liste der unterstützten Instance-Typen finden Sie in der GameLift Amazon-Konsole (unter Ressourcen, Instance und Servicequotas).

Bei der Auswahl eines Instance-Typs sollten Sie zunächst die Instance-Familie berücksichtigen. Instance-Familien bieten verschiedene Kombinationen von CPU-, Arbeitsspeicher-, Speicher- und Netzwerkfunktionen. Erfahren Sie mehr über EC2-Instance-Familien. Innerhalb jeder Familie haben Sie eine Reihe von Instance-Größen zur Auswahl. Beachten Sie bei der Auswahl einer Instance-Größe die folgenden Aspekte:

- Was ist die Mindestinstanzgröße, die Ihren Workload unterstützen kann? Verwenden Sie diese Informationen, um zu kleine Instance-Typen zu eliminieren.
- Welche Instance-Typgrößen eignen sich gut für Ihre Container-Architektur? Idealerweise sollten Sie eine Größe wählen, die mehrere Kopien Ihrer Replikat-Containergruppe aufnehmen kann und dabei nur wenig Speicherplatz verschwendet.
- Welche Skalierungsgranularität ist für Ihr Spiel sinnvoll? Beim Skalieren der Flottenkapazität müssen Instanzen hinzugefügt oder entfernt werden, und jede Instanz steht für die Fähigkeit, eine bestimmte Anzahl von Spielsitzungen zu hosten. Überlegen Sie, wie viel Kapazität Sie mit jeder Instanz hinzufügen oder entfernen möchten. Wenn die Nachfrage der Spieler von Minute zu Minute um Tausende schwankt, kann es sinnvoll sein, sehr große Instanzen zu verwenden, die Hunderte oder Tausende von Spielsitzungen hosten können. Im Gegensatz dazu bevorzugen Sie möglicherweise eine detailliertere Skalierungssteuerung bei kleineren Instance-Typen.
- Gibt es je nach Größe Kosteneinsparungen? Möglicherweise stellen Sie fest, dass die Kosten für bestimmte Instance-Typen je nach Standort je nach Verfügbarkeit variieren.

Optimieren Sie Ihre Laufzeitkonfiguration

Die Laufzeitkonfiguration einer Flotte besteht aus einer Reihe von Anweisungen zum Ausführen von Serverprozessen für das Hosting von Spielsitzungen. Diese Anweisungen werden vom GameLift Amazon-Agenten in jeder Replikat-Containergruppe der Flotte implementiert.

Die Laufzeitkonfiguration einer Flotte bestimmt, wie viele Serverprozesse gleichzeitig in jeder Replikat-Container-Gruppe ausgeführt werden. Diese Einstellung wirkt sich darauf aus, wie Sie die Ressourcenlimits Ihrer Containergruppe berechnen und wie Sie einen Instance-Typ für Ihre Flotte auswählen. Sie müssen diese drei Elemente bei der Gestaltung Ihrer Flotte ausbalancieren.

Weitere Informationen zur Verwendung von Laufzeitkonfigurationen finden Sie unterManagen Sie, wie Amazon Spieleserver GameLift startet.

Legen Sie weitere optionale Flotteneinstellungen fest

Bei der Konfiguration einer Containerflotte können Sie die folgenden optionalen Funktionen verwenden:

- Richte deine Spieleserver so ein, dass sie auf andere AWS Ressourcen zugreifen können. Siehe Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.
- Schütze Spielsitzungen mit aktiven Spielern davor, während eines Scale-Down-Events vorzeitig beendet zu werden.
- Beschränken Sie die Anzahl der Spielsitzungen, die eine Person innerhalb eines begrenzten Zeitraums auf der Flotte erstellen kann.

Containergruppendefinitionen für eine GameLift Amazon-Containerflotte erstellen

Diese Dokumentation bezieht sich auf eine Funktion, die in der öffentlichen Vorschauversion enthalten ist. Änderungen sind vorbehalten.

Eine Container-Gruppendefinition beschreibt, wie Sie Ihre containerisierten Gameserver-Anwendungen in einer Containerflotte bereitstellen. Es ist ein Blueprint, der die Gruppe von Containern identifiziert, die auf der Flotte ausgeführt werden sollen, und wie sie ausgeführt werden. Wenn Sie eine Containerflotte erstellen, geben Sie die Containergruppendefinitionen an, die für die Flotte bereitgestellt werden sollen. Weitere Informationen zu Containergruppen finden Sie unterKomponenten der Containerflotte.

Bevor Sie beginnen

Führen Sie die folgenden Schritte aus:

- Entwerfen Sie eine Container-Architektur für das Hosten Ihrer Spieleserver. Siehe Entwerfen Sie eine GameLift Amazon-Containerflotte.

- Planen Sie die Containerdefinitionen, die in die Containergruppe aufgenommen werden sollen. Wenn Sie die AWS CLI verwenden, erstellen Sie Ihre Container-Definition in einer JSON-Datei.
- Übertragen Sie die endgültigen Container-Images in eine Amazon Elastic Container Registry (Amazon ECR) -Registry in derselben Region, AWS-Region in der Sie die Container-Gruppe erstellen möchten. Amazon GameLift speichert zu dem Zeitpunkt, zu dem Sie die Container-Gruppendefinition erstellen, einen Snapshot jedes Images und verwendet die Kopie bei der Bereitstellung in einer Containerflotte. Siehe Bereite ein Container-Image mit deiner Gameserver-Software vor.
- Stellen Sie sicher, dass Ihr AWS Benutzer über IAM-Berechtigungen für den Zugriff auf das Amazon ECR-Repository verfügt. Siehe Benutzerberechtigungen für Amazon verwalten GameLift. Sie benötigen mindestens Berechtigungen für die folgenden Aktionen:
  - `ecr:DescribeImages`
  - `ecr:BatchGetImage`
  - `ecr:GetDownloadUrlForLayer`

Klonen Sie eine Container-Gruppendefinition

Sie können die GameLift Amazon-Konsole verwenden, um eine bestehende Container-Gruppendefinition zu klonen.

Um eine Container-Gruppe zu klonen

1. Gehen Sie in der GameLift Amazon-Konsole zum linken Navigationsbereich und wählen Sie Containergruppen aus.
2. Wählen Sie auf der Listenseite der Containergruppen die bestehende Container-Gruppe aus, die Sie klonen möchten. Nachdem Sie eine Container-Gruppe ausgewählt haben, ist die Schaltfläche Klonen aktiv.
3. Klicken auf Clone. Diese Aktion öffnet den Assistenten zum Erstellen von Containergruppen mit vordefinierten Einstellungen.
4. Geben Sie einen neuen Namen für die geklonte Containergruppe ein. Die Containergruppe in derselben Region muss eindeutige Namen haben.
5. Gehen Sie die Seiten mit der Containergruppe und der Container-Definition durch, überprüfen Sie die neue Container-Gruppe und erstellen Sie sie.

Erstellen Sie eine Replikat-Container-Gruppendefinition

Eine Replikat-Container-Gruppe verwaltet Ihre Spieleserver-Software. Eine Replikat-Container-Gruppe hat mindestens einen Container, auf dem der Amazon GameLift Agent und Ihre Spieleserver-Prozesse ausgeführt werden. Die Gruppe verfügt möglicherweise über zusätzliche „Sidecar“ - Container, um unterstützende Software auszuführen.

In diesem Thema wird beschrieben, wie Sie mit der GameLift Amazon-Konsole oder den AWS CLI-Tools eine Container-Gruppendefinition erstellen. Ausführlichere Informationen zum Einrichten von Container-Gruppenkonfigurationen finden Sie unterEntwerfen Sie eine GameLift Amazon-Containerflotte.

Console

Wählen Sie in der GameLift Amazon-Konsole den AWS-Region Ort aus, an dem Sie die Container-Gruppe erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie Container-Gruppen. Wählen Sie auf der Seite Container-Gruppen die Option Container-Gruppe erstellen aus.

Schritt 1: Definieren Sie Gruppendetails.

1. Geben Sie einen Definitionsnamen für eine Containergruppe ein. Dieser Name muss für die Region AWS-Konto und eindeutig sein. In der Konsole werden Gruppendefinitionen nach Namen aufgelistet, sodass es hilfreich sein kann, aussagekräftige Bezeichnungen zuzuweisen.
2. Wählen Sie die Strategie zur Planung von Replikaten aus.
3. Geben Sie unter Gesamtspeicherlimit den maximalen Arbeitsspeicher ein, der der Containergruppe zur Verfügung steht. Hilfe zur Berechnung dieses Werts finden Sie unterLegen Sie Ressourcenlimits fest.
4. Geben Sie unter Gesamt-CPU-Limit die maximale Rechenleistung ein, die der Containergruppe zur Verfügung steht. Hilfe zur Berechnung dieses Werts finden Sie unterLegen Sie Ressourcenlimits fest.

Schritt 2: Fügen Sie Containerdefinitionen hinzu.

Definiere den Container mit deiner Gameserver-Anwendung und dem Amazon GameLift Agent. Dies ist Ihr unverzichtbarer Replikat-Container.

1. Geben Sie einen Namen für die Containerdefinition an. Jeder für die Gruppe definierte Container muss einen eindeutigen Namenswert haben.
2. Identifizieren Sie den Amazon ECR-Image-URI des Container-Images. Geben Sie eines der folgenden Formate ein:

   - Nur Bild-URI: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]`
   - Bild-URI + Digest: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]@[digest]`
   - Bild-URI + Tag: `[AWS-Konto].dkr.ecr.[AWS-Region].amazonaws.com/[repository ID]:[tag]`
3. Für Essential-Container wird Yes automatisch für die erste Container-Definition ausgewählt. Wenn Sie eine weitere Containerdefinition hinzufügen, können Sie diese Einstellung für jede Definition ein- oder ausschalten. Weitere Details finden Sie unter Benennen Sie wichtige Container.
4. Legen Sie einen oder mehrere interne Container-Portbereiche fest. Dieser Container hostet deine Spieleserver. Definieren Sie also einen Bereich mit genügend Ports, damit jeder Serverprozess in der Containergruppe ausgeführt werden kann. Weitere Details finden Sie unter Netzwerkverbindungen konfigurieren.
5. Mit den optionalen Einstellungen Overrides und Environment variables kannst du Werte angeben, die beim Start an den Container übergeben werden. Die Werte, die Sie hier festlegen, haben Vorrang vor allen Einstellungen, die sich bereits im Container-Image befinden.
6. Legen Sie optionale Container-Limits fest, um die Ressourcenzuweisung für diesen Container zu verwalten. Weitere Details finden Sie unter Legen Sie Ressourcenlimits fest.
7. Definieren Sie nach Bedarf zusätzliche Container, die nicht unbedingt erforderlich sind:

   - Geben Sie einen Namen für die Container-Definition und einen ECR-Image-URI an. Container, die nicht unbedingt benötigt werden, dürfen den Amazon GameLift Agent nicht ausführen.
   - Legen Sie nur dann einen internen Container-Port-Bereich fest, wenn die Container über Prozesse verfügen, die Netzwerkzugriff benötigen.
   - Richten Sie optional einen Gesundheitscheck für den Container ein. Wenn ein Container, der nicht unbedingt benötigt wird, eine Integritätsprüfung nicht besteht, wird nur der fehlerhafte Container neu gestartet.

- Legen Sie optional nach Bedarf Überschreibungen, Umgebungsvariablen und Grenzwerte für die Ressourcenzuweisung fest.

Schritt 3: Konfigurieren Sie Abhängigkeiten.

Wenn Ihre Container-Gruppendefinition mehr als einen Container enthält, können Sie Abhängigkeiten zwischen ihnen definieren. Verwenden Sie Abhängigkeiten, um Start- und Shutdown-Sequenzen auf der Grundlage der Container-Bedingungen einzurichten. Weitere Details finden Sie unter Legen Sie Container-Abhängigkeiten fest.

1. Identifizieren Sie den Container-Namen, für den Sie eine Abhängigkeit hinzufügen möchten. Dieser Container wird erst gestartet, wenn die Abhängigkeitsbedingung erfüllt ist.
2. Identifizieren Sie den Namen und die Bedingung des Abhängigkeitscontainers. Dieser Container muss die Bedingung erfüllen, bevor der abhängige Container gestartet werden kann.
3. Legen Sie nach Bedarf zusätzliche Abhängigkeiten fest. Sie können mehrere Abhängigkeiten für jeden Container erstellen. Vermeiden Sie die Schaffung von zirkulären Abhängigkeiten.

Schritt 4: Überprüfen und erstellen.

1. Überprüfen Sie alle Ihre Definitionseinstellungen für Containergruppen. Sie können die Konfiguration einer Container-Gruppendefinition nicht ändern, nachdem sie erstellt wurde. Verwenden Sie Bearbeiten, um Änderungen an einem beliebigen Abschnitt vorzunehmen, einschließlich jeder Ihrer Containerdefinitionen für die Gruppe.
2. Wenn Sie mit der Überprüfung fertig sind, wählen Sie Erstellen.

   Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Detailseite für die neue Container-Gruppen-Definitionsressource an. Anfänglich lautet der Status`COPYING`, da Amazon GameLift damit beginnt, Schnappschüsse aller Container-Images für die Gruppe zu erstellen. Wenn diese Phase abgeschlossen ist, ändert sich der Status der Container-Gruppendefinition in`READY`. Eine Containergruppendefinition muss den `READY` Status haben, bevor Sie damit eine Containerflotte erstellen können.

AWS CLI

Wenn Sie die AWS CLI verwenden, um eine Container-Gruppendefinition zu erstellen, verwalten Sie Ihre Container-Definitionskonfigurationen in einer separaten `JSON` Datei. Sie können in Ihrem

CLI-Befehl auf die Datei verweisen. Schemabeispiele finden Sie unterErstellen Sie eine JSON Container-Definitionsdatei.

Erstellen Sie eine Container-Gruppendefinition

Verwenden Sie den `create-container-group-definition` CLI-Befehl, um eine neue Container-Gruppendefinition zu erstellen. Weitere Informationen zu diesem Befehl finden Sie create-container-group-definitionin der AWS CLI-Befehlsreferenz.

Example : Containergruppe replizieren

Dieses Beispiel veranschaulicht eine Anforderung für eine Replikat-Container-Gruppendefinition. Die Befehlsstruktur für die Erstellung von Replikat- und Daemon-Gruppendefinitionen ist im Wesentlichen identisch. Spezifische Details für jeden Gruppentyp sind in den einzelnen Containerdefinitionen beschrieben.

In diesem Beispiel wird davon ausgegangen, dass Sie eine JSON-Datei mit den Containerdefinitionen für diese Gruppe erstellt haben.

```
aws gamelift create-container-group-definition \
    --name MyAdventureGameContainerGroup \
    --operating-system AMAZON_LINUX_2023 \
    --scheduling-strategy REPLICA \
    --total-memory-limit 4096 \
    --total-cpu-limit 1024 \
    --container-definitions file://SimpleServer.json
```

Erstellen Sie eine **JSON** Container-Definitionsdatei

Wenn Sie eine Container-Gruppendefinition erstellen, definieren Sie auch die Container für die Gruppe. Eine Container-Definition spezifiziert das Amazon ECR-Repository, in dem das Container-Image gespeichert ist, sowie optionale Konfigurationen für Netzwerkports, Grenzwerte für die CPU- und Speicherauslastung und andere Einstellungen. Wir empfehlen, eine einzige JSON Datei mit den Konfigurationen für alle Container in einer Containergruppe zu erstellen. Die Pflege einer Datei ist nützlich, um diese kritischen Konfigurationen zu speichern, gemeinsam zu nutzen und Versionen nachzuverfolgen. Wenn Sie die AWS CLI verwenden, um Ihre Container-Gruppendefinitionen zu erstellen, können Sie im Befehl auf die Datei verweisen.

Um eine Container-Definition zu erstellen

1. Erstellen und öffnen Sie eine neue `.JSON` Datei. Beispielsweise:

   ```
   [~/work/glc]$ vim SimpleServer.json
   ```

2. Erstellen Sie für jeden Container der Gruppe eine separate Containerdefinition. Kopieren Sie den folgenden Beispielinhalt und ändern Sie ihn nach Bedarf für Ihre Container. Einzelheiten zur Syntax einer Container-Definition finden Sie ContainerDefinitionInputin der Amazon GameLift API-Referenz.
3. Speichern Sie die Datei lokal, damit Sie in einem AWS CLI-Befehl darauf verweisen können.

Beispiel: Definition eines unverzichtbaren Replikat-Containers

Example

In diesem Beispiel wird der grundlegende Container für Ihre Replikat-Container-Gruppe beschrieben. Der unverzichtbare Replikatcontainer enthält Ihre Gameserver-Anwendung, den Amazon GameLift Agent, und kann auch andere unterstützende Software für Ihr Spiele-Hosting enthalten. Die Definition muss einen Namen, eine Image-URI und eine Portkonfiguration enthalten. In diesem Beispiel werden auch einige containerspezifische Ressourcenlimits festgelegt.

```
[
  {
    "ContainerName": "SimpleServer",
    "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-containers:complex-
server",
    "Essential": true,
    "Cpu": 256,
    "MemoryLimits": {
      "HardLimit": 128
  },
     "PortConfiguration": {
           "ContainerPortRanges": [
              {
                 "FromPort": 2000,
                 "Protocol": "TCP",
                 "ToPort": 2100
              }
           ]
      }
```

```
    }
]
```

Erstellen Sie eine GameLift Amazon-Containerflotte

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Wenn Sie Ihre Containergruppen-Definitionen erstellt haben, verwenden Sie die GameLift Amazon-Konsole oder die AWS Command Line Interface (AWS CLI), um eine Containerflotte zu erstellen.

Nachdem Sie eine neue Flotte erstellt haben, durchläuft der Status der Flotte mehrere Phasen, in denen Amazon Ihre Containergruppen auf jeder Flotteninstanz GameLift bereitstellt und die Spieleserver startet. Wenn die Flotte den Status erreicht`ACTIVE`, ist sie bereit, Spielsitzungen abzuhalten. Hilfe bei Problemen mit der Flottenerstellung finden Sie unter Probleme mit der GameLift Amazon-Flotte beheben.

Console

Wählen Sie in der GameLift Amazon-Konsole den AWS-Region Ort aus, an dem Sie die Flotte erstellen möchten. Die Definitionen der Containergruppen müssen sich in derselben Region befinden, in der Sie die Flotte erstellen möchten.

Öffnen Sie die linke Navigationsleiste der Konsole und wählen Sie Fleets aus. Wählen Sie auf der Seite Flotten die Option Flotte erstellen aus.

Schritt 1: Wählen Sie den Berechnungstyp

- Wählen Sie den Compute-Typ „Container“.

Schritt 2: Definieren Sie die Flottendetails

1. Geben Sie im Abschnitt Flottendetails einen Flottennamen und eine Beschreibung ein.
2. Identifizieren Sie im Abschnitt Containergruppendetails die Containergruppen, die für die Flotte eingesetzt werden sollen. Sie müssen eine Replikat-Containergruppe hinzufügen. Sie können optional eine Daemon-Containergruppe hinzufügen. Jede Gruppe muss den Status `READY` haben.
3. Stellen Sie den Verbindungs-Portbereich für die Flotte ein. Weitere Details finden Sie unter Netzwerkverbindungen konfigurieren.

4. Geben Sie optional die gewünschten Replikate pro bereitzustellender Instanz an. Sie können eine gewünschte Anzahl angeben oder Amazon die maximal mögliche Anzahl GameLift berechnen lassen. Wenn Sie eine gewünschte Anzahl angeben, die größer als das berechnete Maximum ist, schlägt die Flottenerstellung fehl. Sie können diese Einstellung nicht ändern, nachdem die Flotte erstellt wurde. Weitere Informationen zum Packen von Replikatcontainergruppen finden Sie unterSchlüsselkonzepte.
5. (Optional) Unter Zusätzliche Details:

   a. Geben Sie zum Beispiel „Instance-Rolle“ eine IAM-Rolle an, die Anwendungen in Ihrem Spiel-Build autorisiert, auf andere AWS Ressourcen in Ihrem Konto zuzugreifen. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten. Um eine Flotte mit einer Instance-Rolle zu erstellen, muss Ihr Konto über die `PassRole` IAM-Berechtigung verfügen. Weitere Informationen finden Sie unter Beispiele für IAM-Genehmigungen für Amazon GameLift.

   b. Geben Sie unter Metrikgruppe den Namen einer neuen oder vorhandenen Flotten-Metrikgruppe ein. Sie können die Kennzahlen für mehrere Flotten aggregieren, indem Sie sie derselben Metrikgruppe hinzufügen.

Schritt 3: Definieren Sie die Instanzdetails

1. Wählen Sie unter Instanzbereitstellung einen oder mehrere Remote-Standorte aus, an denen Instanzen bereitgestellt werden sollen. Die Heimatregion wird automatisch ausgewählt (dies ist die Region, in der Sie die Flotte erstellen). Wenn Sie zusätzliche Standorte auswählen, werden Flotteninstanzen auch an diesen Standorten bereitgestellt.

   > **Important**
   >
   > Um Regionen zu verwenden, die nicht standardmäßig aktiviert sind, aktivieren Sie sie in Ihrem AWS-Konto.
   >
   > - Flotten mit Regionen, die nicht aktiviert sind und die du vor dem 28. Februar 2022 erstellt hast, sind davon nicht betroffen.
   > - Um neue Flotten mit mehreren Standorten zu erstellen oder bestehende Flotten mit mehreren Standorten zu aktualisieren, aktivieren Sie zunächst alle Regionen, die Sie verwenden möchten.

Weitere Informationen zu Regionen, die standardmäßig nicht aktiviert sind, und zu deren Aktivierung finden Sie unter Verwaltung in der. AWS-RegionenAllgemeine AWS-Referenz

2. Wählen Sie eine Instanzkonfiguration für die Flotte aus. Die Konsole berechnet automatisch die mindestens erforderliche vCPU und den erforderlichen Arbeitsspeicher (basierend auf den Gesamtlimits, die Sie für jede Containergruppe festlegen). Sie filtert die vollständige Liste der verfügbaren Instance-Typen auf der Grundlage der Ressourcenanforderungen und der von Ihnen eingegebenen Standorte. Sie können bei Bedarf weitere Filter hinzufügen.

   Weitere Informationen zur Auswahl eines Instance-Typs finden Sie unterKonfigurieren Sie eine Containerflotte. Die Größe des ausgewählten Instance-Typs wirkt sich darauf aus, wie Replikat-Containergruppen auf jede Flotteninstanz gepackt werden. Je nach Ihrer Wahl sollten Sie in Erwägung ziehen, Ihre Einstellung für die gewünschten Replikate pro Instance zu überprüfen.

Schritt 4: Laufzeit konfigurieren

Die Laufzeitkonfiguration bestimmt, wie Spielserver-Prozesse gestartet und ausgeführt werden. Diese Anweisungen werden an den GameLift Amazon-Agenten weitergeleitet, der sie in jeder Replikat-Container-Gruppe auf die gleiche Weise implementiert. Sie können die Laufzeitkonfiguration einer Flotte aktualisieren, indem Sie anrufen UpdateRuntimeConfiguration.

1. Geben Sie unter Startpfad den Pfad zu einer ausführbaren Datei des Spiels ein.
2. (Optional) Geben Sie unter Startparameter Informationen ein, die als Befehlszeilenparameter an die ausführbare Datei Ihres Spiels übergeben werden sollen.
3. Geben Sie die Anzahl der gleichzeitigen Prozesse an, die in jeder Replikat-Container-Gruppe weiterhin ausgeführt werden sollen. Überprüfen Sie die GameLift Amazon-Kontingente für die Anzahl der Serverprozesse pro Instance. Grenzwerte zu gleichzeitigen Serverprozessen pro Instance gelten für die Summe aller gleichzeitig ausgeführten Prozesse für alle Konfigurationen. Wenn Sie die Flotte so konfigurieren, dass sie das Limit überschreitet, kann die Flotte nicht aktiviert werden.
4. Lege optionale Limits für gleichzeitige Aktivierungen von Spielsitzungen fest. Mit diesen Einstellungen kannst du die Menge an Ressourcen begrenzen, die beim Starten einer neuen

Spielsitzung verbraucht werden. Aktivierungen von Spielsitzungen können sich auf die Leistung vorhandener Spielsitzungen auswirken.

5. Stellen Sie die EC2-Port-Einstellungen so ein, dass externer Datenverkehr auf Prozesse zugreifen kann, die auf der Flotte ausgeführt werden. Geben Sie einige oder alle Verbindungs-Portnummern an, die für die Flotte definiert sind. Du musst diese Ports nicht festlegen, wenn du die Flotte erstellst, aber ohne sie kann kein Traffic eine Verbindung zu deinen Spieleservern herstellen. Um die Porteinstellungen einer Flotte später zu aktualisieren, rufe UpdateFleetPortSettings
6. Konfigurieren Sie unter Ressourceneinstellungen für Spielsitzungen die folgenden optionalen Funktionen:

   a. Aktiviere oder deaktiviere die Game-Scaling-Schutzrichtlinie. Wenn der Schutz aktiviert ist, GameLift wird Amazon Instances während eines Scale-Down-Events nicht herunterfahren, wenn sie eine aktive Spielsitzung veranstalten.

   b. Lege ein maximales Limit für die Erstellung von Ressourcen fest, um die Anzahl der Spielsitzungen zu begrenzen, die ein Spieler während eines bestimmten Zeitraums erstellen kann.

Schritt 5: Tags konfigurieren

- (Optional) Fügen Sie dem Build Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben. Wählen Sie Weiter, um mit der Überprüfung der Flottenerstellung fortzufahren.

Schritt 6: Überprüfen und erstellen.

- Überprüfen Sie Ihre Flottenkonfigurationseinstellungen.

  Sie können die Metadaten und die Konfiguration der Flotte jederzeit aktualisieren, unabhängig vom Flottenstatus. Weitere Informationen finden Sie unter Verwalten Sie Ihre GameLift Amazon-Flotten. Sie können die Flottenkapazität aktualisieren, nachdem die Flotte den Status AKTIV erreicht hat. Weitere Informationen finden Sie unter Skalierung der GameLift Amazon-Hosting-Kapazität. Sie können auch entfernte Standorte hinzufügen oder entfernen.

  Wenn Sie mit der Überprüfung fertig sind, wählen Sie Erstellen.

Wenn Ihre Anfrage erfolgreich ist, zeigt die Konsole die Detailseite für die neue Flottenressource an. Anfänglich lautet der Status`NEW`, da Amazon den Flottenerstellungsprozess GameLift startet. Sie können den Status der neuen Flotte auf der Seite Fleets (Flotten) verfolgen. Eine Flotte ist bereit, Spielsitzungen abzuhalten, wenn sie den Status erreicht hat`ACTIVE`.

AWS CLI

Um eine Containerflotte mit dem zu erstellen AWS CLI, öffnen Sie ein Befehlszeilenfenster und verwenden Sie den `create-fleet` Befehl. Weitere Informationen zu diesem Befehl finden Sie `create-fleet`in der AWS CLI Befehlsreferenz.

Mit der unten abgebildeten `create-fleet` Beispielanforderung wird eine neue Containerflotte mit den folgenden Merkmalen erstellt:

- Das ContainerGroupsConfiguration spezifiziert eine einzelne Replikat-Container-Gruppendefinition:`MegaFrogRaceServer.NA.v2`. Drei Kopien der Replikatgruppe werden für jede Flotteninstanz bereitgestellt. Für jede Instanz stehen 30 Verbindungsports für den Zugriff auf Prozesse auf der Instanz zur Verfügung.
- Die Flotte verwendet c5.large On-Demand-Instances.
- Sie stellt Containergruppen an den folgenden Orten bereit:
  - us-west-2 (Heimatregion)
  - ca-central-1 (entfernter Standort)
- Jede Replikat-Container-Gruppe auf einer Instance führt 5 Spielserverprozesse gleichzeitig aus, sodass jede Instance bis zu 15 Spielsitzungen gleichzeitig hosten kann.
- In jeder Replikat-Container-Gruppe GameLift erlaubt Amazon die gleichzeitige Aktivierung von zwei neuen Spielsitzungen. Außerdem werden alle aktivierenden Spielsitzungen beendet, wenn sie nicht innerhalb von 300 Sekunden bereit sind, Spieler zu empfangen.
- Bei allen auf diesen Instances gehosteten Spielsitzungen ist der Schutz für Spielsitzungen aktiviert.
- Einzelne Spieler können innerhalb von 15 Minuten drei neue Spielsitzungen erstellen.

```
aws gamelift create-fleet \
    --name SampleFleet123 \
    --description "The sample test fleet" \
```

```
    --compute-type "CONTAINER" \
    --container-groups-configuration
 "ContainerGroupDefinitionNames=['MegaFrogRaceServer.NA.v2'],
 DesiredReplicaContainerGroupPerInstance=3,
 ConnectionPortRange={FromPort=1010,ToPort=1040}" \
    --ec2-instance-type c5.large \
    --region us-west-2 \
    --locations "Location=ca-central-1" \
    --fleet-type ON_DEMAND \
    --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
 MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=/local/game/
MegaFrogRace/server.exe,ConcurrentExecutions=5}]" \
    --new-game-session-protection-policy "FullProtection" \
    --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
 PolicyPeriodInMinutes=15" \
    --ec2-inbound-permissions
 "FromPort=1010,ToPort=1040,IpRange=0.0.0.0/0,Protocol=UDP" \
```

Wenn die Anfrage „Flotte erstellen“ erfolgreich ist, GameLift gibt Amazon eine Reihe von Flottenattributen zurück, die die von Ihnen angeforderten Konfigurationseinstellungen und eine neue Flotten-ID enthalten. Amazon setzt GameLift dann den Flottenstatus und den Standortstatus auf Neu und leitet den Flottenaktivierungsprozess ein. Sie können den Status der Flotte nachverfolgen und andere Informationen zu der Flotte über die folgenden CLI-Befehle anzeigen:

- describe-fleet-events
- describe-fleet-attributes
- describe-fleet-capacity
- describe-fleet-port-settings
- describe-fleet-utilization
- describe-runtime-configuration
- describe-fleet-location-attributes
- describe-fleet-location-capacity
- describe-fleet-location-utilization

Mit diesen Befehlen können Sie die Kapazität der Flotte und andere Konfigurationseinstellungen nach Bedarf ändern:

- update-fleet-attributes

- update-fleet-capacity
- update-fleet-port-settings
- update-runtime-configuration
- create-fleet-locations
- delete-fleet-locations

Verwalte deine GameLift Amazon-Containerflotten

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Wenn Sie Informationen über Ihre Containerflotte abrufen oder Änderungen vornehmen möchten, können Sie Ihre Containerflotte mit den folgenden Aktionen verwalten.

Anzeigen der -Ressourcen

Im Folgenden finden Sie einige Möglichkeiten, wie Sie Informationen über die Ressourcen in Ihrer Containerflotte abrufen können.

- DescribeCompute- Gibt Details zu einem Container zurück, der als Compute registriert ist.
- DescribeContainerGroupDefinition- Gibt Details zu einer Container-Gruppendefinition zurück. Diese Ressource beschreibt, wie die Gruppe und ihre Container konfiguriert sind.
- DescribeFleetAttributes- Ruft Flottenattribute ab, zu denen der Verbindungsportbereich und andere Attribute gehören.
- DescribeFleetCapacity- Ruft die Anzahl der Replikat-Containergruppen in der Flotte und deren Status ab.
- DescribeRuntimeConfiguration- Beschreibt Serverprozesse, die in jeder Replikatcontainergruppe ausgeführt werden.
- GetComputeAccess— Ermöglicht den Fernzugriff auf eine Instanz, die die Containergruppe hostet.
- GetComputeAuthToken- Fordert von Amazon ein Authentifizierungstoken GameLift für eine Rechenressource in einer Containerflotte an.
- ListCompute— Listet Containergruppen auf, die als Computes registriert sind.
- ListContainerGroupDefinitions- Listet die Definitionen der Containergruppen auf.

- ListFleets- Listet Flotten auf, die eine bestimmte Containergruppe verwenden.

Aktualisieren von Ressourcen

Im Folgenden finden Sie einige Möglichkeiten, Ressourcen für Containerflotten zu erstellen und zu ändern.

- CreateContainerGroupDefinition- Erstellt eine Container-Gruppendefinition.
- CreateFleet- Erzeugt eine Containerflotte, wenn auf gesetzt `ComputeType` ist`CONTAINER`.
- RegisterCompute- Registriert Berechnungen mit einer Containerflotte.
- UpdateFleetAttributes- Aktualisiert die veränderbaren Attribute einer Flotte, wie z. B. die Anywhere-Flottenkonfigurationsoptionen.
- UpdateFleetCapacity- Aktualisiert die Kapazitätseinstellungen für eine verwaltete EC2-Flotte oder Containerflotte.
- UpdateRuntimeConfiguration- Aktualisiert die Laufzeitkonfiguration, die beschreibt, welche Serverprozesse in jeder als Compute registrierten Replikat-Containergruppe ausgeführt werden sollen.

Löschen von Ressourcen

Im Folgenden finden Sie einige Möglichkeiten, wie Sie Ressourcen für Containerflotten entfernen können.

- DeleteContainerGroupDefinition— Löscht eine Container-Gruppendefinition.
- DeleteFleet- Löscht eine Flotte.
- DeregisterCompute- Entfernt eine Rechenressource aus einer Containerflotte.

Skalierung der GameLift Amazon-Containerflotten

Diese Dokumentation bezieht sich auf eine Funktion, die sich in der öffentlichen Vorschauversion befindet. Änderungen sind vorbehalten.

Eine der schwierigsten Aufgaben beim Hosten von Spielen ist die Skalierung der Kapazität, um der Nachfrage der Spieler gerecht zu werden, ohne Geld für Ressourcen zu verschwenden, die Sie nicht

benötigen. In einer Containerflotte skalieren Sie Ihre Flottenkapazität, indem Sie Flotteninstanzen hinzufügen oder entfernen.

Wenn Sie eine neue Flotte erstellen, GameLift legt Amazon die gewünschte Kapazität der Flotte auf eine Instance fest und stellt eine Instance in der Heimatregion der Flotte bereit. Für eine Flotte mit mehreren Standorten GameLift stellt Amazon eine Instance in der Heimatregion und an jedem Remote-Standort bereit. Sobald der Flottenstatus erreicht ist`ACTIVE`, können Sie die gewünschte Kapazität erhöhen, um nach oben zu skalieren, oder die gewünschte Kapazität verringern, um nach unten zu skalieren.

Sie können die GameLift Skalierungsfunktionen von Amazon verwenden, um die Kapazität manuell zu ändern oder die automatische Skalierung auf der Grundlage der Spielernachfrage einzurichten:

- Richten Sie die automatische Skalierung mit Zielverfolgung ein. Siehe Zielgerichtete automatische Skalierung.
- Ändern Sie die Kapazität Ihrer Flotte manuell. Siehe Manuelles Festlegen der Kapazität für eine GameLift Amazon-Flotte.

Denken Sie bei der Skalierung einer Containerflotte darüber nach, wie sich das Hinzufügen oder Entfernen von Instanzen auf die Kapazität der Flotte auswirkt, Spielsitzungen und Spieler zu veranstalten.

- Spielsitzungen pro Instanz
  - Jeder Spieleserverprozess, der auf einer Instanz ausgeführt wird, entspricht der Kapazität, eine Spielsitzung zu hosten.
  - Verwenden Sie diese Formel, um die Anzahl der Spielsitzungen zu berechnen, die gleichzeitig auf einer Container-Flotteninstanz ausgeführt werden:

    ```
    [Game sessions per instance] = [# of processes per replica container group] * [# of
     replica container groups per instance]
    ```

    - Rufen Sie für Prozesse pro Replikat-Containergruppe die Anzahl der gleichzeitigen Ausführungen für Spieleserverprozesse auf DescribeRuntimeConfigurationund zählen Sie sie.
    - Rufen Sie für Replikat-Containergruppen pro Instanz den Wert DescribeFleetAttributesauf. `DesiredReplicaContainerGroupPerInstance` Wenn dieser Wert nicht festgelegt ist, verwenden Sie den `MaxReplicaContainerGroupsPerInstance` Wert.
- Spieler pro Instanz

- Sie entscheiden, wie viele Spielerplätze Sie in jeder Spielsitzung zulassen möchten. Je nachdem, wie Ihre Hosting-Lösung die Platzierung von Spielsitzungen handhabt, können Sie in Ihrer Matchmaking-Konfiguration oder in Ihren Aufrufen zum Starten einer Spielsitzung Spieler pro Spielsitzung definieren.
- Verwenden Sie diese Formel, um die Anzahl der Spieler zu berechnen, die Ihr Spiel gleichzeitig auf einer Container-Flotteninstanz spielen können:

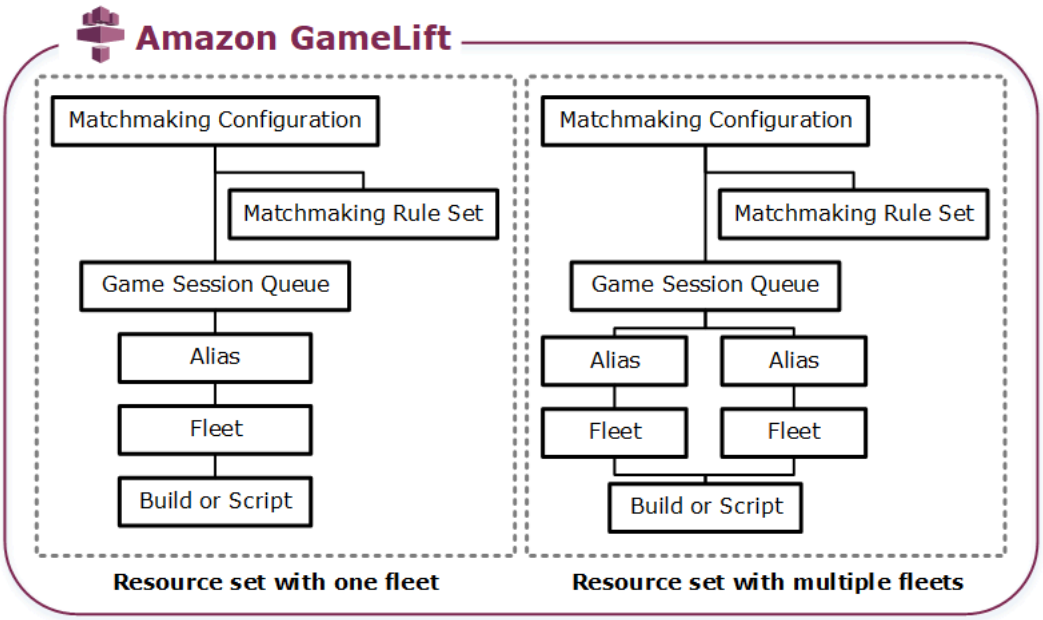
```
[Players per instance] = [# of game sessions per instance] * [# of player slots per
 game session]
```

Um die aktuelle Gesamtkapazität einer Containerflotte zu ermitteln, rufen Sie DescribeFleetCapacityDescribeFleetLocation Capacity auf, um die Anzahl der Replikat-Containergruppen in der Flotte abzurufen. Aktive Gruppen sind Gruppen, die derzeit Spielsitzungen veranstalten. Inaktive Gruppen sind bereit, eine neue Spielsitzung zu veranstalten. Multiplizieren Sie diese Werte mit der Anzahl der Serverprozesse pro Replikat-Containergruppe.

Verwaltung der GameLift Amazon-Hosting-Ressourcen

Dieser Abschnitt enthält detaillierte Informationen zur Einrichtung von von Amazon GameLift verwalteten Ressourcen zum Betrieb Ihrer Spieleserver und zum Hosten von Spielesitzungen für Spieler. Sie müssen Ressourcen konfigurieren und bereitstellen, die Kapazität skalieren, um den Anforderungen der Spieler gerecht zu werden, und verfügbare Ressourcen für die Ausrichtung von Spielsitzungen ausfindig machen.

Das folgende Diagramm zeigt, wie GameLift Amazon-Ressourcenobjekte miteinander in Beziehung stehen. Verwende einen Build oder ein Skript, um eine Flotte zu erstellen, einer Flotte einen Alias zu geben und Flotten mithilfe ihres Alias zu einer Warteschlange für Spielsitzungen hinzuzufügen. Verwenden Sie für Spiele, die FlexMatch Matchmaking verwenden, die Warteschlange für Spielsitzungen und einen Matchmaking-Regelsatz, um eine Matchmaking-Konfiguration zu erstellen.



Spieleservercode

- Build — Ihre maßgeschneiderte Gameserver-Software, die auf Amazon läuft GameLift und Spielesitzungen für Ihre Spieler hostet. Ein Game-Build stellt den Satz von Dateien dar, auf denen Ihr Spieleserver auf einem bestimmten Betriebssystem ausgeführt wird und die Sie in Amazon integrieren müssenGameLift. Laden Sie Game-Build-Dateien GameLift auf Amazon hochAWS-Regionen, wo Sie Flotten einrichten möchten. Weitere Informationen finden Sie unter Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift.
- Script — Deine Konfiguration und benutzerdefinierte Spiellogik für die Verwendung mit Echtzeitservern. Konfiguriere Echtzeitserver für deine Spielclients, indem du mithilfe von

benutzerdefinierter Spiellogik ein Skript JavaScript erstellst, und füge benutzerdefinierte Spiellogik hinzu, um Spielsitzungen für deine Spieler zu hosten. Weitere Informationen finden Sie unter Laden Sie ein Realtime Server-Skript auf Amazon hoch GameLift.

Flotte

Eine Sammlung von Rechenressourcen, die Ihre Spielserver betreiben und Spielsitzungen für Ihre Spieler hosten. Informationen darüber, wo Sie Flotten einsetzen können, finden Sie unterAmazon GameLift -Hosting-Standorte. Hinweise zur Erstellung von Flotten finden Sie unterGameLiftAmazon-Flotten einrichten.

Alias

Eine abstrakte Kennung für eine Flotte, mit der du die Flotte, mit der deine Spieler verbunden sind, jederzeit ändern kannst. Weitere Informationen finden Sie unter Einer GameLift Amazon-Flotte einen Alias hinzufügen.

Warteschlange für Spielsitzungen

Ein Mechanismus zur Platzierung von Spielsitzungen, der Anfragen für neue Spielsitzungen entgegennimmt und nach verfügbaren Spielservern sucht, um die neuen Sessions zu hosten. Weitere Informationen zu Warteschlangen für Spielesitzungen finden Sie unter. GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten

Upload von Builds und Skripten auf Amazon GameLift

Bevor Sie Ihre Multiplayer-Gameserver für das Hosting bei Amazon bereitstellenGameLift, müssen Sie Ihre Gameserverdateien hochladen. Die Themen in diesem Abschnitt enthalten Anleitungen zur Vorbereitung und zum Hochladen benutzerdefinierter Gameserver-Build-Dateien oder Server-Skriptdateien für Realtime Server.

Themen

- Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift
- Laden Sie ein Realtime Server-Skript auf Amazon hoch GameLift

Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift

Nachdem Sie Ihren Spieleserver in Amazon integriert haben GameLift, laden Sie die Build-Dateien auf Amazon hoch GameLift. In diesem Thema erfahren Sie, wie Sie die Build-Dateien Ihres Spiels packen, ein optionales Build-Installationsskript erstellen und die Dateien dann mit dem AWS Command Line Interface(AWS CLI) oder einem AWS SDK hochladen.

Themen

Packen Ihrer Spielserver-Build-Dateien

Bevor Sie Ihren konfigurierten Spieleserver auf Amazon hochladen GameLift, packen Sie die Build-Dateien des Spiels in ein Build-Verzeichnis. Dieses Verzeichnis muss alle für die Ausführung Ihrer Spieleserver erforderlichen Komponenten enthalten und Spielsitzungen hosten, einschließlich der folgenden:

- Gameserver-Binärdateien — Die Binärdateien, die für den Betrieb des Spieleservers erforderlich sind. Ein Build kann Binärdateien für mehrere Spieleserver enthalten, die für die Ausführung auf derselben Plattform erstellt wurden. Eine Liste der unterstützten Plattformen finden Sie unterEntwicklungsunterstützung mit Amazon GameLift.
- Abhängigkeiten — Alle abhängigen Dateien, die die ausführbaren Dateien deines Gameservers zum Ausführen benötigen. Beispiele sind u. a. Assets, Konfigurationsdateien und abhängige Bibliotheken.

> **Note**
>
> Fügen Sie bei Spiele-Builds, die mit dem GameLift Amazon-Server-SDK SDK for C++ erstellt wurden (einschließlich solcher, die mit dem Unreal-Plugin erstellt wurden), die OpenSSL-DLL für dieselbe Version von OpenSSL hinzu, mit der Sie das Server-SDK erstellt haben. Weitere Informationen finden Sie in der README-Datei des Server-SDK.

- Installationsskript (optional) — Eine Skriptdatei zur Bearbeitung von Aufgaben zur Installation Ihres Game-Builds auf GameLift Amazon-Hosting-Servern. Platzieren Sie diese Datei im Stammverzeichnis des Build-Verzeichnisses. Amazon GameLift führt das Installationsskript im Rahmen der Flottenerstellung aus.

Sie können jede Anwendung in Ihrem Build einrichten, einschließlich Ihres Installationsskripts, um sicher auf Ihre Ressourcen in anderen AWS Diensten zuzugreifen. Informationen dazu, wie Sie dies tun können, finden Sie unterKommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Nachdem du deine Build-Dateien gepackt hast, stelle sicher, dass dein Spieleserver auf einer Neuinstallation deines Zielbetriebssystems ausgeführt werden kann. Dadurch wird überprüft, ob du alle erforderlichen Abhängigkeiten in dein Paket aufgenommen hast und ob dein Installationsskript korrekt ist.

Erstellen Sie einen Amazon-Build GameLift

Wenn Sie einen Build erstellen und Ihre Dateien hochladen, haben Sie zwei Optionen:

- Erstellen Sie einen Build aus einem Dateiverzeichnis. Dies ist die einfachste und am häufigsten verwendete Option.
- Erstellen Sie einen Build mit Dateien in Amazon Simple Storage Service (Amazon S3). Mit dieser Option können Sie Ihre Build-Versionen in Amazon S3 verwalten.

Bei beiden Methoden GameLift erstellt Amazon eine neue Build-Ressource mit einer eindeutigen Build-ID und anderen Metadaten. Der Build startet im Status Initialisiert. Nachdem Amazon die GameLift Spielserverdateien erworben hat, wechselt der Build in den Status Bereit.

Wenn der Build fertig ist, können Sie ihn für eine neue GameLift Amazon-Flotte bereitstellen. Weitere Informationen finden Sie unter Erstellen Sie eine von Amazon GameLift verwaltete Flotte .Wenn Amazon die neue Flotte GameLift einrichtet, lädt es die Build-Dateien auf jede Flotteninstanz herunter und installiert die Build-Dateien.

Erstellen Sie einen Build aus einem Dateiverzeichnis

Verwenden Sie den `upload-build`AWS CLIBefehl, um einen Spiel-Build zu erstellen, der an einem beliebigen Ort gespeichert ist, einschließlich eines lokalen Verzeichnisses. Dieser Befehl erstellt einen neuen Build-Datensatz in Amazon GameLift und lädt Dateien von einem von Ihnen angegebenen Speicherort hoch.

Senden Sie eine Upload-Anfrage. Geben Sie in einem Befehlszeilenfenster den folgenden upload-build Befehl und die folgenden Parameter ein.

```
aws gamelift upload-build \
    --name user-defined name of build \
    --operating-system supported OS \
    --server-sdk-version Amazon GameLift server SDK version \
    --build-root build path \
    --build-version user-defined build number \
    --region region name
```

- operating-system— Die Laufzeitumgebung des Gameserver-Builds. Sie müssen einen Betriebssystemwert angeben. Sie können dies später nicht aktualisieren.
- server-sdk-version— Die Version des GameLift Amazon-Server-SDK, in die dein Spieleserver integriert ist. Wenn Sie keinen Wert angeben, GameLift verwendet Amazon den Standardwert`4.0.2`. Wenn Sie eine falsche Server-SDK-Version angeben, schlägt der Spieleserver-Build möglicherweise fehl, wenn `InitSdk` Sie aufrufen, um eine Verbindung zum GameLift Amazon-Service herzustellen.
- build-root— Der Verzeichnispfad Ihrer Build-Dateien.
- name— Ein beschreibender Name für den neuen Build.
- build-version— Die Versionsdetails für die Build-Dateien.
- region— Die AWS Region, in der Sie Ihren Build erstellen möchten. Erstelle den Build in der Region, in der du Flotten einsetzen möchtest. Wenn du dein Spiel in mehreren Regionen einsetzt, erstelle in jeder Region einen Build.

> **Note**
>
> Sieh dir deine aktuelle Standardregion an, indem du aws configure get region Verwenden Sie den aws configure set region ***region name***Befehl, um Ihre Standardregion zu ändern.

Beispiele

```
aws gamelift upload-build \
    --operating-system AMAZON_LINUX_2023 \

    --server-sdk-version "5.0.0" \
    --build-root "~/mygame" \
```

```
    --name "My Game Nightly Build" \
    --build-version "build 255" \
    --region us-west-2
```

```
aws gamelift upload-build \
    --operating-system WINDOWS_2016 \
    --server-sdk-version "5.0.0" \
    --build-root "C:\mygame" \
    --name "My Game Nightly Build" \
    --build-version "build 255" \
    --region us-west-2
```

Als Antwort auf Ihre Upload-Anfrage teilt Amazon den GameLift Upload-Status mit. Bei einem erfolgreichen Upload GameLift gibt Amazon die neue Build-Datensatz-ID zurück. Die Zeit für den Upload hängt von der Größe Ihrer Spieldateien und der Verbindungsgeschwindigkeit ab.

Erstellen Sie einen Build mit Dateien in Amazon S3

Sie können Ihre Build-Dateien in Amazon S3 speichern und GameLift von dort zu Amazon hochladen. Wenn Sie Ihren Build erstellen, geben Sie den Speicherort des S3-Buckets an, und Amazon GameLift ruft die Build-Dateien direkt von Amazon S3 ab.

Um eine Build-Ressource zu erstellen

1. Speichern Sie Ihre Build-Dateien in Amazon S3. Erstellen Sie eine .zip-Datei mit den gepackten Build-Dateien und laden Sie sie in einen S3-Bucket in Ihrem AWS-Konto hoch. Notieren Sie sich das Bucket-Label und den Dateinamen. Sie benötigen diese, wenn Sie einen GameLift Amazon-Build erstellen.
2. Gewähren Sie Amazon GameLift Zugriff auf Ihre Build-Dateien. Erstellen Sie eine IAM-Rolle, indem Sie den Anweisungen unter folgen. Zugriff auf eine Spiele-Build-Datei in Amazon S3 Nachdem Sie die Rolle erstellt haben, notieren Sie sich den Amazon-Ressourcennamen (ARN) der neuen Rolle. Sie benötigen ihn, wenn Sie einen Build erstellen.
3. Erstellen Sie einen Build. Verwenden Sie die GameLift Amazon-Konsole oder dieAWS CLI, um einen neuen Build-Datensatz zu erstellen. Sie benötigen die `PassRole` entsprechende Genehmigung, wie unter beschriebenBeispiele für IAM-Genehmigungen für Amazon GameLift.

Console

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Hosting, Builds aus.

2. Wählen Sie auf der Seite Builds die Option Create Build aus.
3. Gehen Sie auf der Seite Build erstellen unter Build-Einstellungen wie folgt vor:
   a. Geben Sie unter Name einen Skriptnamen ein.
   b. Geben Sie für Version eine Version ein. Da Sie den Inhalt eines Builds aktualisieren können, können Sie anhand von Versionsdaten Aktualisierungen nachverfolgen.
   c. Wählen Sie unter Betriebssystem (OS) das Betriebssystem Ihres Gameserver-Builds aus. Du kannst diesen Wert später nicht aktualisieren.
   d. Geben Sie für Game Server Build die S3-URI des Build-Objekts ein, das Sie auf Amazon S3 hochgeladen haben, und wählen Sie die Objektversion aus. Wenn Sie sich nicht an den Amazon S3 S3-URI und die Objektversion erinnern, wählen Sie Browse S3 und suchen Sie nach dem Build-Objekt.
   e. Wählen Sie für die IAM-Rolle die Rolle aus, die Sie erstellt haben und die Amazon GameLift Zugriff auf Ihren S3-Bucket und Ihr Build-Objekt gewährt.
4. (Optional) Fügen Sie dem Build unter Tags Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben.
5. Wählen Sie Erstellen aus.

Amazon GameLift weist dem neuen Build eine ID zu und lädt die angegebene ZIP-Datei hoch. Sie können den neuen Build einschließlich des Status auf der Build-Seite einsehen.

AWS CLI

Verwenden Sie den `create-build`Befehl, um den neuen Build zu definieren und Ihre Server-Build-Dateien hochzuladen.

1. Öffnen Sie ein Befehlszeilenfenster und wechseln Sie zu einem Verzeichnis, in dem Sie den verwenden könnenAWS CLI.
2. Geben Sie den folgenden create-build Befehl ein:

```
aws gamelift create-build \
    --name user-defined name of build \
    --server-sdk-version Amazon GameLift server SDK version \
    --operating-system supported OS \
    --build-version user-defined build number \
    --storage-location "Bucket"=S3 bucket label,"Key"=Build .zip file
 name,"RoleArn"=Access role ARN} \
```

```
    --region region name
```

- name— Ein beschreibender Name für den neuen Build.
- server-sdk-version— Die Version des GameLift Amazon-Server-SDK, mit der Sie Ihren Spieleserver in Amazon GameLift integriert haben. Wenn Sie keinen Wert angeben, GameLift verwendet Amazon den Standardwert`4.0.2`.
- operating-system— Die Laufzeitumgebung des Gameserver-Builds. Sie müssen einen Betriebssystemwert angeben. Sie können dies später nicht aktualisieren.
- build-version— Die Versionsdetails für die Build-Dateien. Diese Informationen können nützlich sein, da für jede neue Version deines Spielservers eine neue Build-Ressource erforderlich ist.
- storage-location
  - Bucket— Der Name des S3-Buckets, der deinen Build enthält. Zum Beispiel „my_build_files“.
  - Key— Der Name der ZIP-Datei, die Ihre Build-Dateien enthält. Zum Beispiel „my_game_build_7.0.1, 7.0.2".
  - RoleARN— Der ARN, der der von Ihnen erstellten IAM-Rolle zugewiesen wurde. Zum Beispiel „arn:aws:iam: :111122223333:role/“. GameLiftAccess Eine Beispielrichtlinie finden Sie unter Zugriff auf eine Spiele-Build-Datei in Amazon S3.
- region— Erstellen Sie den Build in der Region, in der Sie Flotten einsetzen möchten. AWS Wenn du dein Spiel in mehreren Regionen einsetzt, erstelle in jeder Region einen Build.

> **Note**
>
> Wir empfehlen, deine aktuelle Standardregion mithilfe des `configure get`Befehls zu überprüfen . Verwenden Sie den `configure set`Befehl, um Ihre Standardregion zu ändern.

Beispiel

```
aws gamelift create-build \
    --operating-system WINDOWS_2016 \
    --storage-location
 "Bucket"="my_game_build_files","Key"="mygame_build_101.zip","RoleArn"="arn:aws:iam::111
gamelift" \
```

```
    --name "My Game Nightly Build" \
    --build-version "build 101" \
    --region us-west-2
```

3. Verwenden Sie den `describe-build`Befehl, um den neuen Build anzuzeigen.

Aktualisieren Sie Ihre Build-Dateien

Sie können die Metadaten für eine Build-Ressource mithilfe der GameLift Amazon-Konsole oder des `update-build`AWS CLIBefehls aktualisieren.

Nachdem Sie einen GameLift Amazon-Build erstellt haben, können Sie die damit verknüpften Build-Dateien nicht aktualisieren. Erstellen Sie für jeden neuen Satz von Dateien einen neuen GameLift Amazon-Build. Mithilfe des `upload-build`Befehls erstellt Amazon GameLift automatisch einen neuen Build-Datensatz für jede Anfrage. Wenn Sie Build-Dateien mit dem `create-build`Befehl bereitstellen, laden Sie eine neue Build-.zip-Datei mit einem anderen Namen auf Amazon S3 hoch und erstellen Sie einen Build, indem Sie auf den neuen Dateinamen verweisen.

Berücksichtigen Sie bei der Bereitstellung aktualisierter Builds die folgenden Tipps:

- Verwenden Sie Warteschlangen und tauschen Sie Flotten nach Bedarf aus. Wenn du deinen Spielclient bei Amazon einrichtest GameLift, gib statt einer Flotte eine Warteschlange an. Mit Warteschlangen kannst du die neuen Flotten mit dem neuen Build zu deiner Warteschlange hinzufügen und die alten Flotten entfernen. Weitere Informationen finden Sie unter GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten.
- Verwende Aliase, um Spieler auf einen neuen Spiel-Build zu übertragen. Wenn du deinen Spielclient mit Amazon integrierst GameLift, gib statt einer Flotten-ID einen Flottenalias an. Weitere Informationen finden Sie unter Einer GameLift Amazon-Flotte einen Alias hinzufügen.
- Richten Sie automatische Build-Updates ein. Beispielskripts und Informationen zur Integration von GameLift Amazon-Bereitstellungen in Ihr Build-System finden Sie unter Automating Deployments to GameLift Amazon im AWS Game Tech Blog.

Hinzufügen eines Build-Installationsskripts

Erstellen Sie ein Installationsskript für das Betriebssystem (OS) Ihres Spiel-Builds:

- Windows: Erstelle eine Batch-Datei mit dem Name`install.bat`.
- Linux: Erstellen Sie eine Shell-Skriptdatei mit dem Name`install.sh`.

Beachten Sie beim Erstellen eines Installationsskripts Folgendes:

- Das Skript kann keine Benutzereingaben annehmen.
- Amazon GameLift installiert den Build und erstellt die Dateiverzeichnisse in Ihrem Build-Paket auf einem Hosting-Server an den folgenden Speicherorten neu:
  - Windows-Flotten: `C:\game`
  - Linux-Flotten: `/local/game`
- Während des Installationsvorgangs für Linux-Flotten hat der Run-as-Benutzer eingeschränkten Zugriff auf die Instanzdateistruktur. Dieser Benutzer hat volle Rechte für das Verzeichnis, in dem Ihre Build-Dateien installiert sind. Wenn Ihr Installationsskript Aktionen ausführt, für die Administratorrechte erforderlich sind, geben Sie den Administratorzugriff mit ansudo. Der Run-as-Benutzer für Windows-Flotten verfügt standardmäßig über Administratorrechte. Fehlende Berechtigungen für das Installationsskript generieren eine Ereignismeldung, die auf ein Problem mit dem Skript hinweist.
- Unter Linux GameLift unterstützt Amazon gängige Shell-Interpreter-Sprachen wie Bash. Fügen Sie einen shebang (z. B. `#!/bin/bash`) oben in Ihrem Installationsskript ein. Um die Unterstützung Ihrer bevorzugten Shell-Befehle zu überprüfen, greifen Sie remote auf eine aktive Linux-Instance zu und öffnen Sie eine Shell-Eingabeaufforderung. Weitere Informationen finden Sie unter Remote-Verbindung zu Amazon- GameLift Flotten-Instances.
- Das Installationsskript kann sich nicht auf eine VPC-Peering-Verbindung verlassen. Eine VPC-Peering-Verbindung ist erst verfügbar, nachdem Amazon die Build on Fleet-Instances GameLift installiert hat.

Example Windows installiert die Bash-Datei

Diese `install.bat` Beispieldatei installiert die für den Spieleserver erforderlichen Visual C++-Laufzeitkomponenten und schreibt die Ergebnisse in eine Protokolldatei. Das Skript enthält die Komponentendatei im Build-Paket im Stammverzeichnis.

```
vcredist_x64.exe /install /quiet /norestart /log c:\game\vcredist_2013_x64.log
```

Example Linux-Shell-Skript installieren

Diese `install.sh` Beispieldatei verwendet Bash im Installationsskript und schreibt die Ergebnisse in eine Protokolldatei.

```
#!/bin/bash
```

```
echo 'Hello World' > install.log
```

Diese `install.sh` Beispieldatei zeigt, wie Sie den CloudWatch Amazon-Agenten verwenden können, um Metriken auf Systemebene und benutzerdefinierte Messwerte zu sammeln und die Protokollrotation zu verwalten. Da Amazon in einer Service-VPC GameLift ausgeführt wird, müssen Sie Amazon GameLift Berechtigungen erteilen, um in Ihrem Namen eine AWS Identity and Access Management (IAM-) Rolle zu übernehmen. Damit Amazon GameLift eine Rolle übernehmen kann, erstellen Sie eine Rolle, die die AWS verwaltete Richtlinie enthält`CloudWatchAgentAdminPolicy`, und verwenden Sie diese Rolle, wenn Sie eine Flotte erstellen.

```
sudo yum install -y amazon-cloudwatch-agent
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
sudo yum install -y collectd
cat <<'EOF' > /tmp/config.json
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "root",
        "credentials": {
            "role_arn": "arn:aws:iam::account#:role/rolename"
        }
    },
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/tmp/log",
                        "log_group_name": "gllog",
                        "log_stream_name": "{instance_id}"
                    }
                ]
            }
        }
    },
    "metrics": {
       "namespace": "GL_Metric",
        "append_dimensions": {
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
```

```
        },
        "metrics_collected": {
            // Configure metrics you want to collect.
            // For more information, see Manually create or edit the CloudWatch agent
 configuration file.
        }
    }
}
EOF
sudo mv /tmp/config.json /opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo systemctl enable amazon-cloudwatch-agent.service
```

Laden Sie ein Realtime Server-Skript auf Amazon hoch GameLift

Wenn Sie bereit sind, Echtzeitserver für Ihr Spiel bereitzustellen, laden Sie die fertigen Echtzeit-Server-Skriptdateien auf Amazon hoch. GameLift Erstellen Sie dazu eine GameLift Amazon-Skriptressource und geben Sie den Speicherort Ihrer Skriptdateien an. Sie können Serverskriptdateien, die bereits bereitgestellt wurden, auch aktualisieren, indem Sie neue Dateien für eine vorhandene Skriptressource hochladen.

Wenn Sie eine neue Skriptressource erstellen, GameLift weist Amazon ihr eine eindeutige Skript-ID zu (z. B.`script-1111aaaa-22bb-33cc-44dd-5555eeee66ff`) und lädt eine Kopie der Skriptdateien hoch. Die Uploadzeit hängt von der Größe Ihrer Skriptdateien und von Ihrer Verbindungsgeschwindigkeit ab.

Nachdem Sie die Skriptressource erstellt haben, GameLift stellt Amazon das Skript mit einer neuen Realtime Server-Flotte bereit. Amazon GameLift installiert Ihr Serverskript auf jeder Instance in der Flotte und platziert die Skriptdateien darin`/local/game`.

Informationen zur Behebung von Problemen mit der Flottenaktivierung im Zusammenhang mit dem Serverskript finden Sie unter. Probleme mit der GameLift Amazon-Flotte beheben

Skriptdateien verpacken

Ihr Serverskript kann eine oder mehrere Dateien enthalten, die zu einer einzigen ZIP-Datei zum Hochladen zusammengefasst sind. Die ZIP-Datei muss alle Dateien enthalten, die Ihr Skript zur Ausführung benötigt.

Sie können Ihre komprimierten Skriptdateien entweder in einem lokalen Dateiverzeichnis oder in einem Amazon Simple Storage Service (Amazon S3) -Bucket speichern.

Laden Sie Skriptdateien aus einem lokalen Verzeichnis hoch

Wenn Sie Ihre Skriptdateien lokal gespeichert haben, können Sie sie GameLift von dort auf Amazon hochladen. Um die Skriptressource zu erstellen, verwenden Sie entweder die GameLift Amazon-Konsole oder die AWS Command Line Interface(AWS CLI).

Amazon GameLift console

Um eine Skriptressource zu erstellen

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich Hosting, Scripts aus.
3. Wählen Sie auf der Seite Scripts die Option Script erstellen aus.
4. Gehen Sie auf der Seite Skript erstellen unter Skripteinstellungen wie folgt vor:
   a. Geben Sie für Name einen Skriptnamen ein.
   b. (Optional) Geben Sie für Version Versionsinformationen ein. Da Sie den Inhalt eines Skripts aktualisieren können, können Versionsdaten bei der Nachverfolgung von Updates hilfreich sein.
   c. Wählen Sie als Skriptquelle die Option Eine ZIP-Datei hochladen aus.
   d. Wählen Sie für Skriptdateien die Option Datei auswählen, suchen Sie nach der ZIP-Datei, die Ihr Skript enthält, und wählen Sie dann diese Datei aus.
5. (Optional) Fügen Sie dem Skript unter Tags Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben.
6. Wählen Sie Erstellen aus.

   Amazon GameLift weist dem neuen Skript eine ID zu und lädt die angegebene ZIP-Datei hoch. Sie können das neue Skript einschließlich seines Status auf der Seite Scripts einsehen.

AWS CLI

Verwenden Sie den `create-script`AWS CLIBefehl, um das neue Skript zu definieren und Ihre Server-Skriptdateien hochzuladen.

Um eine Skriptressource zu erstellen

1. Platzieren Sie die ZIP-Datei in einem Verzeichnis, in dem Sie die AWS CLI verwenden können.
2. Öffnen Sie ein Befehlszeilenfenster und wechseln Sie in das Verzeichnis, in dem Sie die ZIP-Datei abgelegt haben.
3. Geben Sie den folgenden create-script Befehl und die folgenden Parameter ein. Stellen Sie sicher, dass Sie für den --zip-file Parameter die Zeichenfolge `fileb://` zum Namen der ZIP-Datei hinzufügen. Es identifiziert die Datei als Binärdatei, sodass Amazon den komprimierten Inhalt GameLift verarbeitet.

```
aws gamelift create-script \
    --name user-defined name of script \
    --script-version user-defined version info \
    --zip-file fileb://name of zip file \
    --region region name
```

Beispiel

```
aws gamelift create-script \
    --name "My_Realtime_Server_Script_1" \
    --script-version "1.0.0" \
    --zip-file fileb://myrealtime_script_1.0.0.zip \
    --region us-west-2
```

Als Antwort auf Ihre Anfrage GameLift gibt Amazon das neue Skriptobjekt zurück.

4. Um das neue Skript anzusehen, rufen Sie an`describe-script`.

Laden Sie Skriptdateien von Amazon S3 hoch

Sie können Ihre Skriptdateien in einem Amazon S3-Bucket speichern und sie GameLift von dort zu Amazon hochladen. Wenn Sie Ihr Skript erstellen, geben Sie den Speicherort des S3-Buckets an und Amazon GameLift ruft Ihre Skriptdateien von Amazon S3 ab.

Um eine Skriptressource zu erstellen

1. Speichern Sie Ihre Skriptdateien in einem S3-Bucket. Erstellen Sie eine ZIP-Datei, die Ihre Serverskriptdateien enthält, und laden Sie sie in einen S3-Bucket hochAWS-Konto, den Sie

kontrollieren. Notieren Sie sich die Objekt-URI — Sie benötigen diese, wenn Sie ein GameLift Amazon-Skript erstellen.

> **Note**
>
> Amazon unterstützt das Hochladen von S3-Buckets mit Namen, die einen Punkt (.) enthalten, GameLift nicht.

2. Geben Sie Amazon GameLift Zugriff auf Ihre Skriptdateien. Um eine AWS Identity and Access Management (IAM-) Rolle zu erstellen, die Amazon den GameLift Zugriff auf den S3-Bucket ermöglicht, der Ihr Serverskript enthält, folgen Sie den Anweisungen unterEinrichten einer IAM-Servicerolle für Amazon GameLift. Nachdem Sie die neue Rolle erstellt haben, notieren Sie sich ihren Namen, den Sie beim Erstellen eines Skripts benötigen.
3. Erstellen Sie ein Skript. Verwenden Sie die GameLift Amazon-Konsole oder dieAWS CLI, um einen neuen Skriptdatensatz zu erstellen. Um diese Anfrage stellen zu können, benötigen Sie die `PassRole` IAM-Berechtigung, wie unter beschriebenBeispiele für IAM-Genehmigungen für Amazon GameLift.

Amazon GameLift console

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Hosting, Scripts aus.
2. Wählen Sie auf der Seite Scripts die Option Script erstellen aus.
3. Gehen Sie auf der Seite Skript erstellen unter Skripteinstellungen wie folgt vor:
   a. Geben Sie für Name einen Skriptnamen ein.
   b. (Optional) Geben Sie für Version Versionsinformationen ein. Da Sie den Inhalt eines Skripts aktualisieren können, können Versionsdaten bei der Nachverfolgung von Updates hilfreich sein.
   c. Wählen Sie als Skriptquelle Amazon S3-URI aus.
   d. Geben Sie die S3-URI des Skriptobjekts ein, das Sie in Amazon S3 hochgeladen haben, und wählen Sie dann die Objektversion aus. Wenn Sie sich nicht an die Amazon S3-URI und die Objektversion erinnern, wählen Sie „S3 durchsuchen“ und suchen Sie dann nach dem Skriptobjekt.
4. (Optional) Fügen Sie dem Skript unter Tags Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben.
5. Wählen Sie Erstellen aus.

Amazon GameLift weist dem neuen Skript eine ID zu und lädt die angegebene ZIP-Datei hoch. Sie können das neue Skript einschließlich seines Status auf der Seite Scripts einsehen.

AWS CLI

Verwenden Sie den `create-script`AWS CLIBefehl, um das neue Skript zu definieren und Ihre Server-Skriptdateien hochzuladen.

1. Öffnen Sie ein Befehlszeilenfenster und wechseln Sie in ein Verzeichnis, in dem Sie das verwenden könnenAWS CLI.
2. Geben Sie den folgenden create-script Befehl und die folgenden Parameter ein. Der --storage-location Parameter gibt den Amazon S3-Bucket-Speicherort Ihrer Skriptdateien an.

```
aws gamelift create-script \
    --name [user-defined name of script] \
    --script-version [user-defined version info] \
    --storage-location "Bucket"=S3 bucket name,"Key"=name of zip file in S3
 bucket,"RoleArn"=Access role ARN \
    --region region name
```

Beispiel

```
aws gamelift create-script \
    --name "My_Realtime_Server_Script_1" \
    --script-version "1.0.0" \
    --storage-location "Bucket"="gamelift-
script","Key"="myrealtime_script_1.0.0.zip","RoleArn"="arn:aws:iam::123456789012:role/
S3Access" \
    --region us-west-2
```

Als Antwort auf Ihre Anfrage GameLift gibt Amazon das neue Skriptobjekt zurück.

3. Um das neue Skript anzusehen, rufen Sie an `describe-script`.

Skriptdateien aktualisieren

Sie können die Metadaten für eine Skriptressource entweder mit der GameLift Amazon-Konsole oder mit dem `update-script`AWS CLIBefehl aktualisieren.

Sie können auch den Skriptinhalt für eine Skriptressource aktualisieren. Amazon GameLift stellt Skriptinhalte für alle Flotteninstanzen bereit, die die aktualisierte Skriptressource verwenden. Wenn das aktualisierte Skript bereitgestellt wird, verwenden Instanzen es, wenn neue Spielsitzungen gestartet werden. Spielsitzungen, die zum Zeitpunkt des Updates bereits laufen, verwenden das aktualisierte Skript nicht.

Um Skriptdateien zu aktualisieren

- Verwenden Sie für lokal gespeicherte Skriptdateien entweder die GameLift Amazon-Konsole oder den update-script Befehl, um die aktualisierte Skriptdatei hochzuladen.
- Laden Sie für Skriptdateien, die in einem Amazon S3-Bucket gespeichert sind, die aktualisierten Skriptdateien in den S3-Bucket hoch. Amazon sucht GameLift regelmäßig nach aktualisierten Skriptdateien und ruft diese direkt aus dem S3-Bucket ab.

GameLiftAmazon-Flotten einrichten

Dieser Abschnitt enthält detaillierte Informationen über die Planung, den Aufbau und die Wartung von Flotten für AmazonGameLift. Sie können GameLift Amazon-Flotten verwenden, um benutzerdefinierte Spieleserver und Echtzeitserver bereitzustellen.

Eine Flotte stellt Ihre Hosting-Ressourcen als Satz von Amazon Elastic Compute Cloud (Amazon EC2) -Instances oder physischer Hardware dar. Der Standort einer Flotte bestimmt, wo Instanzen oder Hardware eingesetzt werden, um Spielsitzungen für Ihre Spieler zu veranstalten. Die Größe einer Flotte und die Anzahl der Spielsitzungen und Spieler, die sie unterstützen kann, hängen von der Anzahl der Instanzen oder der Menge an Hardware ab, die Sie ihr zur Verfügung stellen. Sie können virtuelle Instanzen manuell oder mithilfe der automatischen Skalierung anpassen.

Viele Spiele in der Produktion verwenden mehr als eine Flotte. Du kannst mehrere Flotten verwenden, um beispielsweise mehr als eine Version deines Gameservers gleichzeitig laufen zu lassen, um Backup-Kapazität für Spot-Flotten bereitzustellen oder um Redundanz einzubauen.

Um zu lernen, wie Sie Flotten erstellen, die auf die Bedürfnisse Ihres Spiels zugeschnitten sind, beginnen Sie mitLeitfaden zur GameLift Amazon-Flottenplanung. Nachdem Ihre Flotte läuft, sehen SieSkalierung der GameLift Amazon-Hosting-Kapazität,Einer GameLift Amazon-Flotte einen Alias hinzufügen, undGameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten.

Themen

- Leitfaden zur GameLift Amazon-Flottenplanung

- Erstellen Sie eine neue GameLift Amazon-Flotte
- Verwalten Sie Ihre GameLift Amazon-Flotten
- Einer GameLift Amazon-Flotte einen Alias hinzufügen
- Probleme mit der GameLift Amazon-Flotte beheben
- Remote-Verbindung zu Amazon- GameLift Flotten-Instances

Leitfaden zur GameLift Amazon-Flottenplanung

In diesem Designleitfaden werden bewährte Methoden für die Erstellung einer Flotte von Hosting-Ressourcen zur Verwendung mit Amazon behandeltGameLift. Wähle eine Kombination aus Hosting-Ressourcen und lerne, wie du sie an dein Spiel anpassen kannst.

Themen

- Auswahl von GameLift Amazon-Rechenressourcen
- Managen Sie, wie Amazon Spieleserver GameLift startet
- Spot-Instances mit Amazon verwenden GameLift

Auswahl von GameLift Amazon-Rechenressourcen

Um Ihre Spieleserver bereitzustellen und Spielsitzungen für Ihre Spieler zu hosten, GameLift verwendet Amazon Elastic Compute Cloud (Amazon EC2) -Ressourcen, die als Instances bezeichnet werden, oder Ihre physische Hardware. Entscheiden Sie bei der Einrichtung einer neuen Flotte mithilfe von Instances, welche Art von Instances Sie benötigen und wie Sie Spielserverprozesse auf diesen ausführen. Wenn eine verwaltete EC2-Flotte aktiv und bereit ist, Spielsitzungen zu veranstalten, können Sie nach Bedarf Instances hinzufügen oder entfernen, um der Nachfrage der Spieler gerecht zu werden.

Sie können Ihre GameLift Amazon-Spieleserver auf einer Kombination aus zwei Rechenarten bereitstellen:

- Managed EC2 — Verwaltete EC2-Flotten verwenden Amazon EC2 EC2-Instances, um Ihre Spieleserver zu hosten. Amazon GameLift verwaltet die Instanzen und nimmt Ihnen die Last der Hardware- und Softwaremanagement beim Hosten Ihrer Spiele ab.
- Amazon GameLift Anywhere — GameLift Anywhere Amazon-Flotten nutzen Ihre bestehende Infrastruktur, um Spieleserver zu hosten, während Amazon Ihr Matchmaking und Ihre Warteschlangen GameLift verwaltet.

Berücksichtigen Sie bei der Auswahl der Rechenressourcen für Ihre Flotte die folgenden Faktoren:

- Verfügbare Hardware
- Standort der Flotte
- On-Demand-Instances versus Spot-Instances
- Betriebssysteme
- Instance-Typen
- Servicekontingente

Verfügbare Hardware

Berücksichtigen Sie die bestehende Infrastruktur in Ihrer Implementierung. Während Sie Spiele zu Amazon migrieren GameLift, können Sie Ihre Infrastruktur weiterhin nutzen. Mit Amazon GameLift Anywhere können Sie Ihre eigene Infrastruktur zusammen mit von Amazon GameLift verwalteten EC2-Instances verwenden. Sie können auch Ihre bestehende Infrastruktur nutzen, um Spiele in der Nähe Ihrer Spieler zu hosten, als es die unterstützten GameLift Amazon-Standorte zulassen. Weitere Informationen zur Einrichtung von GameLift Anywhere Amazon-Flotten finden Sie unterErstellen Sie eine GameLift Anywhere Amazon-Flotte.

Standort der Flotte

Berücksichtigen Sie die geografischen Standorte, an denen Sie Ihre Spieleserver bereitstellen möchten. Die Verfügbarkeit des Instanztyps variiert je nach AWS-Region lokaler Zone.

Bei Flotten mit mehreren Standorten hängen Instance-Verfügbarkeit und Kontingente von einer Kombination aus der Heimatregion der Flotte und ausgewählten Remote-Standorten ab. Weitere Informationen zu Flottenstandorten finden Sie unter. Amazon GameLift -Hosting-Standorte

Für GameLift Anywhere Amazon-Flotten bestimmen Sie den Standort Ihrer physischen Hardware. Weitere Informationen zu benutzerdefinierten Standorten finden Sie unterAmazon GameLift Anywhere.

On-Demand-Instances versus Spot-Instances

Amazon EC2 On-Demand-Instances und Spot-Instances bieten dieselbe Hardware und Leistung, unterscheiden sich jedoch in Verfügbarkeit und Kosten.

On-Demand Instances

Sie können eine On-Demand-Instance erwerben, wenn Sie sie benötigen, und sie so lange behalten, wie Sie möchten. On-Demand-Instances haben feste Kosten, d. h. Sie zahlen für die Zeit, für die Sie sie nutzen, und es bestehen keine langfristigen Verpflichtungen.

Spot-Instances

Spot-Instances können durch die Nutzung ungenutzter AWS Rechenkapazität eine kosteneffiziente Alternative zu On-Demand-Instances bieten. Die Preise für Spot-Instances schwanken je nach Angebot und Nachfrage für jeden Instance-Typ an jedem Standort. AWS kann Spot-Instances unterbrechen, wann immer die Kapazität wieder benötigt wird. Amazon GameLift verwendet Warteschlangen und den FleetIQ-Algorithmus, um zu bestimmen, dass eine Spot-Instance unterbrochen AWS wird. Dadurch wird die Instance in einen Recyclingstatus versetzt. Wenn es dann keine aktiven Spielsitzungen auf der Instance gibt, GameLift versucht Amazon, sie zu ersetzen.

Weitere Informationen zur Verwendung von Spot-Instances finden Sie unterSpot-Instances mit Amazon verwenden GameLift.

Betriebssysteme

GameLift Amazon-Instances unterstützen Spieleserver-Builds, die unter Microsoft Windows oder Amazon Linux ausgeführt werden. Wenn Sie einen Spiel-Build auf Amazon hochladen GameLift, geben Sie das Betriebssystem für das Spiel an. Wenn Sie eine Amazon EC2 EC2-Flotte erstellen, um den Spiel-Build bereitzustellen, richtet Amazon GameLift automatisch Instances mit dem Betriebssystem des Builds ein. Weitere Informationen zu den unterstützten Betriebssystemen für Spieleserver finden Sie unterEntwicklungsunterstützung mit Amazon GameLift.

Wenn Sie eine GameLift Anywhere Amazon-Flotte verwenden, können Sie jedes Betriebssystem verwenden, das Ihre Hardware unterstützt. Bei GameLift Anywhere Amazon-Flotten müssen Sie Ihren Game-Build auf der Hardware bereitstellen und gleichzeitig Amazon verwenden GameLift , um Ihre Ressourcen von einem Ort aus zu verwalten.

Instance-Typen

Der Instance-Typ einer Amazon EC2 EC2-Flotte bestimmt die Art der Hardware, die die Instances verwenden. Verschiedene Instance-Typen bieten unterschiedliche Kombinationen von Rechenleistung, Arbeitsspeicher, Speicher und Netzwerkfunktionen.

Wenn du einen der verfügbaren Instance-Typen für dein Spiel auswählst, solltest du Folgendes berücksichtigen:

- Die Rechenarchitektur deines Spieleservers: x64 oder Arm (AWS Graviton).

> **Note**
>
> Graviton Arm-Instances erfordern einen GameLift Amazon-Server, der auf einem Linux-Betriebssystem basiert. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich. Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

- Die Rechen-, Arbeitsspeicher- und Speicheranforderungen Ihres Gameserver-Builds.
- Die Anzahl der Serverprozesse, die Sie pro Instanz ausführen möchten.

Wenn Sie einen größeren Instanztyp verwenden, können Sie möglicherweise mehrere Serverprozesse auf jeder Instanz ausführen. Dadurch kann die Anzahl der Instanzen reduziert werden, die erforderlich sind, um die Nachfrage der Spieler zu decken.

Weitere Informationen:

- Informationen zu Instance-Typen finden Sie unter Amazon EC2 EC2-Instance-Typen.
- Informationen zum Ausführen mehrerer Prozesse pro Instance finden Sie unterManagen Sie, wie Amazon Spieleserver GameLift startet.

Servicekontingente

Gehen Sie wie folgt vor GameLift, um die Standard-Servicekontingente für Amazon und die aktuellen Kontingente für Sie AWS-Konto zu sehen:

- Allgemeine Informationen zu Servicekontingenten für Amazon GameLift finden Sie unter GameLiftAmazon-Endpunkte und Kontingente in der Allgemeine AWS-Referenz.
- Eine Liste der verfügbaren Instance-Typen pro Standort für Ihr Konto finden Sie auf der Seite Service-Kontingente der GameLift Amazon-Konsole. Auf dieser Seite wird auch die aktuelle Nutzung Ihres Kontos für jeden Instance-Typ an jedem Standort angezeigt.
- Führen Sie den Befehl AWS Command Line Interface (AWS CLI) aus, um eine Liste der aktuellen Kontingente Ihres Kontos für Instance-Typen pro Region zu erhalten `describe-ec2-instance-limits`. Dieser Befehl gibt die Anzahl der aktiven Instances zurück, die Sie in Ihrer Standardregion (oder in einer anderen von Ihnen angegebenen Region) haben.

Wenn Sie sich auf den Start Ihres Spiels vorbereiten, füllen Sie in der GameLift Amazon-Konsole einen Fragebogen zum Start aus. Das GameLift Amazon-Team verwendet den Fragebogen zur Markteinführung, um die richtigen Kontingente und Limits für Ihr Spiel zu ermitteln.

Managen Sie, wie Amazon Spieleserver GameLift startet

Sie können die Laufzeitkonfiguration einer verwalteten EC2-Flotte so einrichten, dass mehrere Gameserverprozesse pro Instance ausgeführt werden. Dadurch werden Ihre Hosting-Ressourcen effizienter genutzt.

So verwaltet eine Flotte mehrere Prozesse

Amazon GameLift verwendet die Laufzeitkonfiguration einer Flotte, um den Typ und die Anzahl der Prozesse zu bestimmen, die auf jeder Instance ausgeführt werden sollen. Eine Laufzeitkonfiguration enthält mindestens eine Serverprozesskonfiguration, die eine ausführbare Spieleserverdatei darstellt. Du kannst zusätzliche Serverprozesskonfigurationen definieren, um andere Arten von Prozessen im Zusammenhang mit deinem Spiel auszuführen. Jede Serverprozesskonfiguration enthält die folgenden Informationen:

- Den Dateinamen und den Pfad einer ausführbaren Datei in Ihrem Spiele-Build
- (Optional) Parameter, die beim Start an den Prozess weitergeleitet werden
- Die Anzahl der Prozesse, die gleichzeitig ausgeführt werden sollen.

Wenn eine Instance in der Flotte aktiviert wird, startet sie die in der Laufzeitkonfiguration definierten Serverprozesse. Bei mehreren Prozessen verschiebt GameLift Amazon den Start jedes Prozesses. Serverprozesse haben eine begrenzte Lebensdauer. Nach deren Ende GameLift startet Amazon neue Prozesse, um die Anzahl und Art der Serverprozesse beizubehalten, die in der Laufzeitkonfiguration definiert sind.

Sie können die Laufzeitkonfiguration jederzeit ändern, indem Sie Serverprozesskonfigurationen hinzufügen, ändern oder entfernen. Jede Instanz überprüft regelmäßig, ob die Laufzeitkonfiguration der Flotte aktualisiert wurde, um die Änderungen zu implementieren. So GameLift nimmt Amazon Änderungen an der Laufzeitkonfiguration an:

1. Die Instance sendet eine Anfrage an Amazon GameLift für die neueste Version der Laufzeitkonfiguration.
2. Die Instanz vergleicht ihre aktiven Prozesse mit der neuesten Laufzeitkonfiguration und geht dann wie folgt vor:

- Wenn die aktualisierte Laufzeitkonfiguration einen Serverprozesstyp entfernt, werden aktive Serverprozesse dieses Typs weiter ausgeführt, bis sie beendet werden. Die Instanz ersetzt diese Serverprozesse nicht.
- Wenn die aktualisierte Laufzeitkonfiguration die Anzahl der gleichzeitigen Prozesse für einen Serverprozesstyp verringert, werden überschüssige Serverprozesse dieses Typs weiter ausgeführt, bis sie beendet werden. Die Instanz ersetzt diese überschüssigen Serverprozesse nicht.
- Wenn die aktualisierte Laufzeitkonfiguration einen neuen Serverprozesstyp hinzufügt oder die Anzahl gleichzeitiger Prozesse für einen vorhandenen Typ erhöht, startet die Instance neue Serverprozesse bis zum GameLift Amazon-Maximum. In diesem Fall startet die Instanz neue Serverprozesse, wenn bestehende Prozesse enden.

Optimieren Sie eine Flotte für mehrere Prozesse

Gehen Sie wie folgt vor, um mehrere Prozesse in einer Flotte zu verwenden:

- Erstellen Sie einen Build, der die ausführbaren Spieleserverdateien enthält, die Sie für eine Flotte bereitstellen möchten, und laden Sie den Build dann auf Amazon hoch. GameLift Alle Spieleserver in einem Build müssen auf derselben Plattform laufen und das Amazon GameLift Server SDK verwenden.
- Erstellen Sie eine Laufzeitkonfiguration mit einer oder mehreren Serverprozesskonfigurationen und mehreren gleichzeitigen Prozesse.
- Integrieren Sie Spieleclients mit der AWS SDK-Version 2016-08-04 oder höher.

Um die Flottenleistung zu optimieren, empfehlen wir Ihnen, Folgendes zu tun:

- Behandeln Sie Szenarien zum Herunterfahren von Serverprozessen, damit Amazon Prozesse GameLift effizient recyceln kann. Beispiele:
  - Füge deinem Spieleservercode eine Shutdown-Prozedur hinzu, die die Server-API `ProcessEnding()` aufruft.
  - Implementiere die Callback-Funktion `OnProcessTerminate()` in deinem Gameservercode, um Kündigungsanfragen von Amazon GameLift zu bearbeiten.
- Stellen Sie sicher, dass Amazon GameLift fehlerhafte Serverprozesse herunterfährt und neu startet. Melden Sie den Gesundheitsstatus an Amazon zurück, GameLift indem Sie die `OnHealthCheck()` Rückruffunktion in Ihrem Gameservercode implementieren. Amazon fährt

Serverprozesse, die in drei aufeinanderfolgenden Berichten als fehlerhaft gemeldet wurden, GameLift automatisch ab. Wenn Sie dies nicht implementieren`OnHealthCheck()`, GameLift geht Amazon davon aus, dass ein Serverprozess fehlerfrei ist, es sei denn, der Prozess reagiert nicht auf eine Kommunikation.

Wählen Sie die Anzahl der Prozesse pro Instanz

Beachten Sie bei der Entscheidung, wie viele Prozesse gleichzeitig auf einer Instanz ausgeführt werden sollen, Folgendes:

- Amazon GameLift begrenzt jede Instance auf eine maximale Anzahl gleichzeitiger Prozesse. Die Summe aller gleichzeitigen Prozesse für die Serverprozesskonfigurationen einer Flotte darf dieses Kontingent nicht überschreiten.
- Um ein akzeptables Leistungsniveau aufrechtzuerhalten, begrenzt der Amazon EC2-Instance-Typ möglicherweise die Anzahl der Prozesse, die gleichzeitig ausgeführt werden können. Testen Sie verschiedene Konfigurationen für Ihr Spiel, um die richtige Anzahl von Prozessen für Ihren bevorzugten Instanztyp zu finden.
- Amazon führt GameLift nicht mehr Prozesse gleichzeitig aus als die konfigurierte Gesamtzahl. Dies bedeutet, dass der Übergang von der vorherigen Laufzeitkonfiguration zur neuen Konfiguration schrittweise erfolgen kann.

Spot-Instances mit Amazon verwenden GameLift

Bei der Einrichtung Ihrer von Amazon GameLift verwalteten EC2-Flotte können Sie Spot-Instances, On-Demand-Instances oder eine Kombination aus beiden verwenden. Erfahren Sie mehr darüber, wie Amazon Spot-Instances GameLift verwendet inOn-Demand-Instances versus Spot-Instances. Um Spot-Flotten verwenden zu können, sind für Ihre Spielintegration die auf dieser Seite aufgeführten Anpassungen erforderlich.

Benutzt du es FlexMatch für das Matchmaking? Sie können Spot-Flotten zu Ihren vorhandenen Spiel-Sitzungswarteschlangen hinzufügen, um Matchmaking-Platzierungen zu erhalten.

1. Entwerfen Sie Ihre Warteschlange für Spielsitzungen für Spot-Instances.

   Es hat sich bewährt, die Platzierung von Spielsitzungen mit einer Warteschlange zu verwalten. Dies ist auch bei der Verwendung von Spot-Instances erforderlich. Beachten Sie bei der Gestaltung Ihrer Warteschlange Folgendes:

- Standorte — Um das beste Spielerlebnis zu erzielen, wähle Standorte, die sich geografisch in der Nähe deiner Spieler befinden.
- Instanztypen — Berücksichtigen Sie die Hardwareanforderungen Ihrer Spieleserver und die Verfügbarkeit von Instanzen an den von Ihnen ausgewählten Standorten.

Informationen zum Testen einer Warteschlange, die die Verfügbarkeit und Resilienz von Spot optimiert, finden Sie unter. Tutorial: Richten Sie eine Warteschlange für Spielsitzungen für Spot-Instances ein

2. Erstellen Sie die Flotten für Ihre Spot-optimierte Warteschlange.

   Erstellen Sie auf der Grundlage Ihres Warteschlangendesigns Flotten, um Ihre Spieleserver an den von Ihnen gewünschten Standorten und Instanztypen bereitzustellen. Weitere Informationen zum Erstellen und Konfigurieren neuer Flotten finden Sie unter Erstellen Sie eine von Amazon GameLift verwaltete Flotte.

3. Erstelle deine Warteschlange für Spielsitzungen.

   Füge die Flottenziele hinzu, konfiguriere den Prozess für die Platzierung von Spielsitzungen und definiere Platzierungsprioritäten. Weitere Informationen zum Erstellen und Konfigurieren der neuen Warteschlange finden Sie unter Erstellen Sie eine Warteschlange für Spielsitzungen.

4. Aktualisiere deinen Spiel-Client-Dienst, sodass er die Warteschlange nutzen kann.

   Wenn dein Spielclient eine Warteschlange verwendet, um Ressourcen anzufordern, vermeidet die Warteschlange Ressourcen, bei denen die Wahrscheinlichkeit einer Unterbrechung hoch ist, und wählt den Ort aus, der deinen definierten Prioritäten entspricht. Hilfe bei der Implementierung der Platzierungen von Spielsitzungen in Ihrem Spielclient finden Sie unter Spielsitzungen erstellen.

5. Aktualisiere deinen Spieleserver, um mit einer Spot-Unterbrechung fertig zu werden.

   AWS kann Spot-Instances mit einer zweiminütigen Benachrichtigung unterbrechen, wenn die Kapazität wieder benötigt wird. Richten Sie Ihren Spieleserver so ein, dass er mit Unterbrechungen umgehen kann, um die Auswirkungen auf die Spieler zu minimieren.

   Bevor eine Spot-Instance AWS zurückgefordert wird, sendet sie eine Kündigungsbenachrichtigung. Amazon GameLift leitet die Benachrichtigung an alle betroffenen Serverprozesse weiter, indem es die Amazon GameLift Server SDK-Callback-Funktion aufruft. `onProcessTerminate()` Implementieren Sie diesen Callback, um die Spielsitzung zu

beenden oder die Spielsitzung und die Spieler auf eine neue Instanz zu verschieben. Hilfe zur Implementierung von `onProcessTerminate()` finden Sie unter Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses.

> **Note**
>
> AWS bemüht sich nach Kräften, die Benachrichtigung bereitzustellen, bevor es eine Instance zurückfordert, aber es ist möglich, dass die Spot-Instance AWS zurückgefordert wird, bevor die Warnung eintrifft. Bereite deinen Spieleserver auf unerwartete Unterbrechungen vor.

6. Überprüfe die Leistung deiner Spot-Flotten und Warteschlangen.

   Sehen Sie sich GameLift Amazon-Metriken in der GameLift Amazon-Konsole oder bei Amazon CloudWatch an, um die Leistung zu überprüfen. Weitere Informationen zu GameLift Amazon-Metriken finden Sie unterÜberwachen Sie Amazon GameLift mit Amazon CloudWatch. Zu den wichtigsten Metriken gehören:

   - Unterbrechungsrate — Verwenden Sie die `GameSessionInterruptions` Messwerte `InstanceInterruptions` und, um die Anzahl und Häufigkeit von SPOT-bedingten Unterbrechungen für Instances und Spielsitzungen nachzuverfolgen. Spielsitzungen, die von zurückgefordert wurden, AWS haben den Status `TERMINATED` und den Statusgrund von. `INTERRUPTED`
   - Effektivität der Warteschlangen — Verfolge die Erfolgsquoten bei der Platzierung, die durchschnittliche Wartezeit und die Warteschlangentiefe, um sicherzugehen, dass Spot-Flotten deine Leistung in der Warteschlange nicht beeinträchtigen.
   - Flottennutzung — Überwachen Sie Daten zu Instanzen, Spiel- und Spielersitzungen. Die Nutzung Ihrer On-Demand-Flotten kann ein Indikator dafür sein, dass Warteschlangen dazu führen, dass Sie nicht in Ihre Spot-Flotten aufgenommen werden, um Störungen zu vermeiden.

Erstellen Sie eine neue GameLift Amazon-Flotte

Erstellen Sie eine neue Flotte und stellen Sie Ihren benutzerdefinierten Gameserver-Build oder Echtzeitserver für das Hosting bereit. Sie können jede Game-Build- oder Skriptressource bereitstellen, die Sie auf Amazon hochladenGameLift.

Themen

So funktioniert die GameLift Amazon-Flottenerstellung

Wenn Sie eine neue Flotte erstellen, GameLift startet Amazon einen Workflow, der eine Flotte mit einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance an jedem Flottenstandort erstellt. Sobald Amazon jeden Schritt des Workflows GameLift abgeschlossen hat, sendet die Flotte Ereignisse aus und Amazon GameLift aktualisiert den Status der Flotte. Sie können alle Ereignisse mithilfe der GameLift Amazon-Konsole oder durch Aufrufen des GameLift Amazon-API-Vorgangs verfolgen DescribeFleetEvents. Sie können den Status einzelner Standorte auch mithilfe von verfolgen DescribeFleetLocationAttributes.

Arbeitsablauf bei der EC2-Flottenerstellung:

- Amazon GameLift erstellt eine Flottenressource in der Heimatregion der Flotte und an jedem in der Flotte definierten entfernten Standort.
- Amazon GameLift legt die gewünschte Kapazität auf eine Instance fest.
- Amazon GameLift setzt den Flotten- und Standortstatus auf Neu.
- Amazon GameLift beginnt, Ereignisse in das Flottenereignisprotokoll zu schreiben.
- Amazon weist GameLift die angeforderten Rechenressourcen für eine neue Instance an jedem Flottenstandort zu.
- Amazon GameLift lädt die Spieleserverdateien auf jede Instanz herunter und setzt den Flottenstatus auf Wird heruntergeladen.
- Amazon GameLift überprüft die heruntergeladenen Spieleserverdateien auf jeder Instanz, um sicherzustellen, dass beim Herunterladen keine Fehler aufgetreten sind. Amazon GameLift setzt den Flottenstatus auf Validierend.
- Amazon GameLift erstellt den Spielserver auf jeder Instance und setzt den Flottenstatus auf Building.
- Amazon GameLift beginnt mit dem Starten von Serverprozessen auf jeder Instance und folgt dabei den Anweisungen in der Laufzeitkonfiguration der Flotte. Wenn Sie die Flotte so konfiguriert haben, dass mehrere Serverprozesse pro Instance gleichzeitig ausgeführt werden, GameLift verschiebt Amazon die Prozessstarts um einige Sekunden. Sobald jeder Prozess online geht, wird Amazon über die Verfügbarkeit informiertGameLift. Amazon GameLift setzt den Flottenstatus auf Aktiviert.

- Amazon GameLift setzt den Flottenstatus und den Standortstatus auf Aktiv, wenn Serverprozesse die Bereitschaft melden.

Amazon GameLift Anywhere fleet creation

- Amazon GameLift erstellt eine Flottenressource. Für die Heimatregion der Flotte und jeden in der Flotte definierten benutzerdefinierten Standort GameLift setzt Amazon den Flotten- und Standortstatus auf Neu.
- Amazon GameLift beginnt, Ereignisse in das Flottenereignisprotokoll zu schreiben.
- Nachdem ein Serverprozess in einer Flotte Amazon darüber informiert hatGameLift, dass er bereit ist, GameLift setzt Amazon den Flottenstatus und den Standortstatus auf Aktiv. Da Serverprozesse an anderen Flottenstandorten die Bereitschaft melden, GameLift setzt Amazon den Status jedes Flottenstandorts auf Aktiv.

Hilfe bei der Behebung von Problemen mit der Flottenerstellung finden Sie unterProbleme mit der GameLift Amazon-Flotte beheben.

Erstellen Sie eine von Amazon GameLift verwaltete Flotte

Verwenden Sie entweder die GameLift Amazon-Konsole oder die AWS Command Line Interface (AWS CLI), um eine verwaltete Flotte zu erstellen.

Nachdem Sie eine neue verwaltete EC2-Flotte erstellt haben, durchläuft der Status der Flotte mehrere Phasen, während Amazon die Flotte GameLift bereitstellt und die Spieleserver installiert und startet. Die Flotte ist bereit, Spielsitzungen abzuhalten, sobald sie den Status erreicht hat`ACTIVE`. Hilfe bei Problemen mit der Flottenerstellung finden Sie unter Probleme mit der GameLift Amazon-Flotte beheben.

Console

Um eine verwaltete EC2-Flotte zu erstellen

1. Wählen Sie in der GameLift Amazon-Konsole im Navigationsbereich Fleets aus.
2. Klicken Sie auf der Seite Fleets (Flotten) auf Create fleet (Flotte erstellen).
3. Wählen Sie Managed EC2.
4. Gehen Sie auf der Seite mit den Flottendetails wie folgt vor:

a. Geben Sie unter Name einen Flottennamen ein. Wir empfehlen, den Flottentyp (Spot oder On-Demand) in Ihre Flottennamen aufzunehmen. Dies macht es viel einfacher, Flottenarten zu identifizieren, wenn Sie sich eine Flottenliste ansehen.

b. Geben Sie unter Beschreibung eine kurze Beschreibung der Flotte ein.

c. Wählen Sie für Binärtyp die Option Build oder Script aus, um den Spieleservertyp zu definieren, den Amazon für diese Flotte GameLift bereitstellt.

d. Wählen Sie ein Skript oder einen Build aus der Drop-down-Liste der hochgeladenen Skripte oder Builds aus.

5. (Optional) Unter Zusätzliche Details für Folgendes:

a. Geben Sie zum Beispiel „Instance-Rolle“ eine IAM-Rolle an, die Anwendungen in Ihrem Spiel-Build autorisiert, auf andere AWS Ressourcen in Ihrem Konto zuzugreifen. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten. Um eine Flotte mit einer Instance-Rolle zu erstellen, muss Ihr Konto über die `PassRole` IAM-Berechtigung verfügen. Weitere Informationen finden Sie unter Beispiele für IAM-Genehmigungen für Amazon GameLift.

   Wenn Sie Anwendungen autorisieren möchten, die keine ausführbaren Serverdateien sind, z. B. CloudWatch Agenten, aktivieren Sie die Option für gemeinsame Anmeldeinformationen.

   Sie können diese Einstellungen nach der Flottenerstellung nicht aktualisieren.

b. Wählen Sie für die Generierung von Zertifizierungen, dass Amazon ein TLS-Zertifikat für die Flotte GameLift generiert. Sie können ein Flotten-TLS-Zertifikat verwenden, damit Ihr Spiele-Client einen Spiel-Server authentifiziert, wenn eine Verbindung hergestellt wird, und die gesamte Client-/Serverkommunikation verschlüsselt. Für jede Instance in einer TLS-fähigen Flotte erstellt Amazon GameLift außerdem einen neuen DNS-Eintrag mit dem Zertifikat. Verwenden Sie diese Ressourcen, um Authentifizierung und Verschlüsselung für Ihr Spiel einzurichten.

c. Geben Sie für Metric Group den Namen einer neuen oder vorhandenen Flotten-Metrikgruppe ein. Sie können die Kennzahlen für mehrere Flotten aggregieren, indem Sie sie derselben Metrikgruppe hinzufügen.

   Sie können die Metrikgruppe nach der Flottenerstellung nicht aktualisieren.

6. Wählen Sie Weiter aus.

7. Wählen Sie auf der Seite Standorte auswählen einen oder mehrere zusätzliche Remote-Standorte aus, an denen Instances bereitgestellt werden sollen. Die Heimatregion wird automatisch basierend auf der Region ausgewählt, von der aus Sie auf die Konsole zugreifen. Wenn Sie zusätzliche Standorte auswählen, werden Flotteninstanzen auch an diesen Standorten bereitgestellt.

   > **Important**
   >
   > Um Regionen zu verwenden, die nicht standardmäßig aktiviert sind, aktivieren Sie sie in IhremAWS-Konto.
   >
   > - Flotten mit Regionen, die nicht aktiviert sind und die du vor dem 28. Februar 2022 erstellt hast, sind davon nicht betroffen.
   > - Um neue Flotten mit mehreren Standorten zu erstellen oder bestehende Flotten mit mehreren Standorten zu aktualisieren, aktivieren Sie zunächst alle Regionen, die Sie verwenden möchten.
   >
   > Weitere Informationen zu Regionen, die standardmäßig nicht aktiviert sind, und zu deren Aktivierung finden Sie unter Verwaltung in der. AWS-RegionenAllgemeine AWS-Referenz

8. Wählen Sie Weiter aus.
9. Wählen Sie auf der Seite „Instanzdetails definieren“

   a. On-Demand-Instances oder Spot-Instances für diese Flotte. Weitere Informationen zu Flottenarten finden Sie unterOn-Demand-Instances versus Spot-Instances.

   b. Wählen Sie im Menü „Filterarchitektur“ die Option x64 oder Arm aus.

      > **Note**
      >
      > Graviton Arm-Instances erfordern einen GameLift Amazon-Server, der auf dem Linux-Betriebssystem basiert. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich. Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

Informationen zu Amazon EC2 Arm-Architekturen finden Sie unter AWSGraviton Processor und Amazon EC2 EC2-Instance-Typen.

Informationen zu den von Amazon GameLift unterstützten Instance-Typen finden Sie in den `EC2InstanceType` Werten unter CreateFleet() -Anforderungsparametern.

10. Wählen Sie einen Amazon EC2 EC2-Instance-Typ aus der Liste aus. Weitere Informationen zur Auswahl eines Instance-Typs finden Sie unterInstance-Typen. Nachdem Sie die Flotte erstellt haben, können Sie den Instance-Typ nicht mehr ändern.
11. Wählen Sie Weiter aus.
12. Gehen Sie auf der Seite Laufzeit konfigurieren unter Laufzeitkonfiguration wie folgt vor:

    a. Geben Sie als Startpfad den Pfad zur ausführbaren Datei des Spiels in Ihrem Build oder Skript ein. Auf Windows-Instances befinden sich die Builds der Spiel-Server in dem Pfad `C:\game`. Auf Linux-Instances sind Spieleserver darauf ausgelegt`/local/game`. Beispiele:**`C:\game\MyGame\server.exe`**,**`/local/game/MyGame/server.exe`**, oder**`MyRealtimeLaunchScript.js`**.

    b. (Optional) Geben Sie unter Startparameter Informationen ein, die als Befehlszeilenparameter an die ausführbare Datei Ihres Spiels übergeben werden sollen. Beispiel: **`+sv_port 33435 +start_lobby`**.

    c. Wählen Sie unter Gleichzeitige Prozesse die Anzahl der Serverprozesse aus, die gleichzeitig auf jeder Instance in der Flotte ausgeführt werden sollen. Sehen Sie sich die GameLift Amazon-Beschränkungen für die Anzahl gleichzeitiger Serverprozesse an.

    Grenzwerte zu gleichzeitigen Serverprozessen pro Instance gelten für die Summe aller gleichzeitig ausgeführten Prozesse für alle Konfigurationen. Wenn Sie die Flotte so konfigurieren, dass sie das Limit überschreitet, kann die Flotte nicht aktiviert werden.
13. Gib unter Aktivierung von Spielsitzungen Beschränkungen für die Aktivierung neuer Spielsitzungen auf den Instances in dieser Flotte an:

    a. Geben Sie für die maximale Anzahl gleichzeitiger Spielsitzungen die Anzahl der Spielsitzungen auf einer Instance ein, die gleichzeitig aktiviert werden. Dieser Grenzwert ist nützlich, wenn der Start von mehreren neuen Spielsitzungen Auswirkungen auf die Leistung anderer Spielsitzungen hat, die in derselben Instance ausgeführt werden.

b. Geben Sie unter Zeitlimit für neue Aktivierung ein, wie lange auf die Aktivierung einer Sitzung gewartet werden soll. Wenn die Spielsitzung nicht vor Ablauf des Timeouts in den `ACTIVE` Status wechselt, GameLift beendet Amazon die Aktivierung der Spielsitzung.

14. (Optional) Gehen Sie unter EC2-Port-Einstellungen wie folgt vor:

    a. Wählen Sie Porteinstellung hinzufügen aus, um Zugriffsberechtigungen für eingehenden Datenverkehr zu definieren, der eine Verbindung zu dem auf der Flotte bereitgestellten Serverprozess herstellt.

    b. Wählen Sie als Typ Benutzerdefiniertes TCP oder Benutzerdefiniertes UDP.

    c. Geben Sie für Portbereich einen Bereich von Portnummern ein, die eingehende Verbindungen zulassen. Ein Portbereich muss das Format `nnnnn[-nnnnn]` mit Werten zwischen 1026 und 60000 verwenden. Beispiel: **`1500`** oder **`1500-20000`**.

    d. Geben Sie für IP-Adressbereich einen Bereich von IP-Adressen ein. Verwenden Sie CIDR-Notation. Beispiel: **`0.0.0.0/0`** (Dieses Beispiel ermöglicht allen Benutzern, sich zu verbinden.)

15. (Optional) Gehen Sie unter Ressourceneinstellungen für Spielsitzungen wie folgt vor:

    a. Wählen Sie unter Richtlinie zum Skalierungsschutz für Spiele die Option Skalierungsschutz ein- oder ausschalten. Amazon beendet Instances mit Schutz während eines Scale-Down-Events GameLift nicht, wenn sie eine aktive Spielsitzung veranstalten.

    b. Geben Sie als Limit für die Erstellung von Ressourcen eine maximale Anzahl von Spielsitzungen ein, die ein Spieler während des Versicherungszeitraums erstellen kann.

16. Wählen Sie Weiter aus.

17. (Optional) Fügen Sie dem Build Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben. Wählen Sie Weiter, um mit der Überprüfung der Flottenerstellung fortzufahren.

18. Wählen Sie Erstellen aus. Amazon GameLift weist der neuen Flotte eine ID zu und beginnt mit der Flottenaktivierung. Sie können den Status der neuen Flotte auf der Seite Fleets (Flotten) verfolgen.

Sie können die Metadaten und die Konfiguration der Flotte jederzeit aktualisieren, unabhängig vom Flottenstatus. Weitere Informationen finden Sie unter Verwalten Sie Ihre GameLift Amazon-Flotten. Sie können die Flottenkapazität aktualisieren, nachdem die Flotte den Status AKTIV

erreicht hat. Weitere Informationen finden Sie unter Skalierung der GameLift Amazon-Hosting-Kapazität. Sie können auch entfernte Standorte hinzufügen oder entfernen.

AWS CLI

Um eine Flotte mit dem zu erstellenAWS CLI, öffnen Sie ein Befehlszeilenfenster und verwenden Sie den `create-fleet` Befehl. Weitere Informationen über den Befehl `create-fleet` finden Sie unter `create-fleet` in der AWS CLI-Befehlsreferenz.

Die unten gezeigte `create-fleet`-Beispielanforderung legt eine neue Flotte mit den folgenden Merkmalen an:

- Die Flotte verwendet c5.large On-Demand-Instances mit dem Betriebssystem, das für den ausgewählten Spiel-Build geeignet ist.
- Sie stellt den angegebenen Spielserver-Build, der sich im Status Bereit befinden muss, an den folgenden Speicherorten bereit bereit bereit bereit bereit bereit bereit bereit bereit bereit bereit:
  - us-west-2 (Heimatregion)
  - sa-east-1 (abgelegener Standort)
- Die Generierung des TLS-Zertifikats ist aktiviert.
- Jede Instance in der Flotte wird zehn identische Prozesse des Spielservers gleichzeitig ausführen, sodass jede Instance bis zu zehn Spielsitzungen gleichzeitig hosten kann.
- Auf jeder Instanz GameLift erlaubt Amazon die gleichzeitige Aktivierung von zwei neuen Spielsitzungen. Außerdem wird jede aktivierende Spielsitzung beendet, wenn sie nicht innerhalb von 300 Sekunden bereit sind, Spieler zu empfangen.
- Bei allen auf diesen Instances gehosteten Spielsitzungen ist der Schutz für Spielsitzungen aktiviert.
- Einzelne Spieler können innerhalb von 15 Minuten drei neue Spielsitzungen erstellen.
- Jede auf dieser Flotte gehostete Spielsitzung hat einen Verbindungspunkt, der innerhalb der angegebenen IP-Adresse und Portbereiche liegt.
- Amazon GameLift fügt Metriken für diese Flotte zur `EMEAfleets` Metrikgruppe hinzu, in der (in diesem Beispiel) Kennzahlen für alle Flotten in den EMEA-Regionen zusammengefasst werden.

```
aws gamelift create-fleet \
    --name SampleFleet123 \
    --description "The sample test fleet" \
```

```
    --ec2-instance-type c5.large \
    --region us-west-2 \
    --locations "Location=sa-east-1" \
    --fleet-type ON_DEMAND \
    --build-id build-92f061ed-27c9-4a02-b1f4-6f85b2385620 \
    --certificate-configuration "CertificateType=GENERATED" \
    --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
 MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=C:\game
\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe, Parameters=+sv_port
 33435 +start_lobby, ConcurrentExecutions=10}]" \
    --new-game-session-protection-policy "FullProtection" \
    --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
 PolicyPeriodInMinutes=15" \
    --ec2-inbound-permissions
 "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
 "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" \
    --metric-groups  "EMEAfleets"
```

Wenn die Anfrage „Flotte erstellen“ erfolgreich ist, GameLift gibt Amazon eine Reihe von Flottenattributen zurück, die die von Ihnen angeforderten Konfigurationseinstellungen und eine neue Flotten-ID enthalten. Amazon leitet GameLift dann den Flottenaktivierungsprozess ein und setzt den Flottenstatus und den Standortstatus auf Neu. Sie können den Status der Flotte nachverfolgen und andere Informationen zu der Flotte über die folgenden CLI-Befehle anzeigen:

- describe-fleet-events
- describe-fleet-attributes
- describe-fleet-capacity
- describe-fleet-port-settings
- describe-fleet-utilization
- describe-runtime-configuration
- describe-fleet-location-attributes
- describe-fleet-location-capacity
- describe-fleet-location-utilization

Mit diesen Befehlen können Sie die Kapazität der Flotte und andere Konfigurationseinstellungen nach Bedarf ändern:

- update-fleet-attributes

- update-fleet-capacity
- update-fleet-port-settings
- update-runtime-configuration
- create-fleet-locations
- delete-fleet-locations

Erstellen Sie eine GameLift Anywhere Amazon-Flotte

Verwenden Sie Amazon GameLift , um Hardware aus Ihrer Umgebung in Ihr GameLift Amazon-Spielehosting zu integrieren. Amazon GameLift Anywhere registriert Ihre Hardware bei Amazon GameLift in einer Anywhere Flotte. Sie können EC2-Flotten in Matchmaker Anywhere - und Spielesitzungswarteschlangen integrieren und verwalten, um das Matchmaking und die Platzierung von Spielen zu verwalten.

Weitere Informationen zum Testen Ihrer Spieleserver mit Amazon GameLift Anywhere finden Sie unterTesten Sie Ihre Integration mit GameLift Anywhere Amazon-Flotten.

Verwenden Sie zunächst Entwicklungsunterstützung mit Amazon GameLift Version 5 oder höher und lesen Sie sich die folgenden Konzepte für die Verwendung einer GameLift Anywhere Amazon-Flotte durch.

Benutzerdefinierte Standorte

GameLift AnywhereAmazon-Flotten verwenden benutzerdefinierte Standorte, um die physischen Standorte Ihrer Infrastruktur darzustellen.

Geräteregistrierung

Damit eine GameLift Anywhere Amazon-Flotte mit Ihren Rechenressourcen kommunizieren kann, registrieren Sie zunächst Ihr Gerät. Sie können die Geräteregistrierung über das Amazon GameLift AWS SDK abschließen, indem Sie den `RegisterCompute`Vorgang verwenden. Dieser Vorgang verwendet die IP-Adresse des Geräts, um es einem Flottenstandort zuzuordnen und mit Amazon zu kommunizieren GameLift.

Authentifizierungstoken

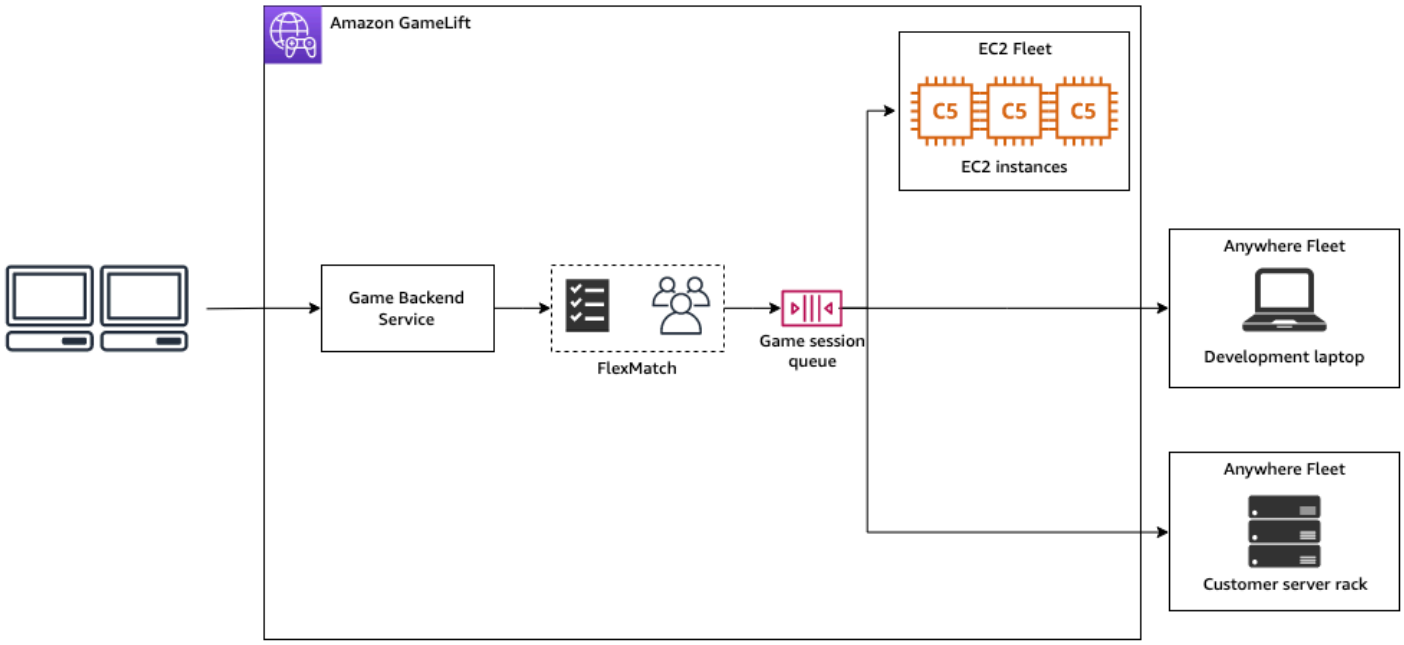
Wenn Sie einen Spieleserver auf Ihrem Computer initialisieren, verwendet das Amazon GameLift Server SDK ein Authentifizierungstoken, um Ihren Spieleserver bei Amazon zu authentifizieren. GameLift Sie können dasselbe Authentifizierungstoken für alle Spieleserver auf demselben

Computer bis zum Ablauf des Authentifizierungstokens wiederverwenden. Rufen Sie den Befehl () auf, um das Authentifizierungstoken `get-compute-auth-token` AWS Command Line Interface abzurufen.AWS CLIÜbergeben Sie das Token nach Bedarf an jeden Spieleserver.

Spielsitzungen

Jede Spielsitzung auf einem Computer verwendet dasselbe Authentifizierungstoken, das bei der Registrierung des Computers an einem Flottenstandort erstellt wurde.

Das folgende Diagramm zeigt eine Warteschlange für Spielsitzungen, bei der FlexMatch Spielersuche und mehrere Flotten verwendet werden. Zu den Flotten gehören eine EC2-Flotte mit C5-Instances, eine Anywhere Flotte mit einem Entwicklungs-Laptop und eine Flotte mit einem vom Kunden gehosteten Anywhere Server-Rack.



Themen

- Erstellen Sie einen benutzerdefinierten Standort
- Erstellen einer Flotte
- Registrieren Sie Ihren Computer
- Führe einen Serverprozess aus
- Spielsitzungen erstellen
- Migrieren Sie zu verwaltetem EC2

Erstellen Sie einen benutzerdefinierten Standort

Um mit dem Hosten von Spielen auf Ihren Rechenressourcen zu beginnen, erstellen Sie einen benutzerdefinierten Speicherort, der beschreibt, wo sich Ihr Computer befindet.

Console

Um einen benutzerdefinierten Standort zu erstellen

1. Öffnen Sie die GameLift Amazon-Konsole.
2. Wählen Sie im Navigationsbereich unter Hosting die Option Standorte aus.
3. Wählen Sie auf der Seite Locations (Speicherorte) die Option Create location (Speicherort erstellen) aus.
4. Gehen Sie im Dialogfeld Standort erstellen wie folgt vor:
   a. Geben Sie einen Standortnamen ein. Dies kennzeichnet den Standort Ihrer Hardware, die Amazon GameLift verwendet, um Ihre Spiele in Anywhere Flotten auszuführen. Amazon GameLift hängt den Namen Ihres benutzerdefinierten Standorts mit custom- an.
   b. (Optional) Fügen Sie Ihrem benutzerdefinierten Standort Tags als Schlüssel-Wert-Paare hinzu. Wählen Sie für jedes Tag, das Sie hinzufügen möchten, die Option Neues Tag hinzufügen aus.
   c. Wählen Sie Erstellen.

AWS CLI

Erstellen Sie mit dem `create-location`Befehl eine benutzerdefinierte Position. Die `location-name` beschriftet den Standort Ihrer Hardware, die Amazon GameLift verwendet, um Ihre Spiele in Anywhere Flotten auszuführen. Wenn Sie Ihren benutzerdefinierten Standort erstellen, muss der Ortsname mit `custom-` beginnen.

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

Output

```
{
    "Location": {
        "LocationName": "custom-location-1",
```

```
        "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-
location-1"
    }
}
```

Erstellen einer Flotte

Verwenden Sie entweder die GameLift Amazon-Konsole oder die AWS CLI , um eine Anywhere Flotte zu erstellen.

Nachdem Sie eine neue Anywhere Flotte erstellt haben, wechselt der Status der Flotte von `NEW` zu`ACTIVE`. Wenn der `ACTIVE` Status erreicht ist, ist die Flotte bereit, Spielsitzungen abzuhalten. Hilfe bei Problemen mit der Flottenerstellung finden Sie unter Probleme mit der GameLift Amazon-Flotte beheben.

Console

Um eine Anywhere Flotte zu erstellen

1. Öffnen Sie die GameLift Amazon-Konsole.
2. Wählen Sie im Navigationsbereich unter Hosting die Option Fleets aus.
3. Klicken Sie auf der Seite Fleets (Flotten) auf Create fleet (Flotte erstellen).
4. Wählen Sie Anywhereim Schritt Compute-Typ die Option und anschließend Weiter aus.
5. Definieren Sie im Schritt Flottendetails die Details und wählen Sie dann Weiter aus.
6. Wählen Sie im Schritt Benutzerdefinierte Standorte den benutzerdefinierten Standort aus, den Sie erstellt haben, und klicken Sie dann auf Weiter. Amazon wählt das Zuhause GameLift automatisch AWS-Region als die Region aus, in der Sie die Flotte erstellen. Sie können die Heimatregion verwenden, um auf Ihre Ressourcen zuzugreifen und diese zu nutzen.
7. Schließen Sie die verbleibenden Schritte zur Flottenerstellung ab und wählen Sie dann Absenden, um Ihre Anywhere Flotte zu erstellen.

AWS CLI

Erstellen Sie mit dem `create-fleet`Befehl eine Anywhere Flotte. Fügen Sie Ihren benutzerdefinierten Standort hinzu`locations`. Amazon GameLift erstellt die Flotte in Ihrer Heimatregion und an den von Ihnen angegebenen benutzerdefinierten Standorten. Ersetzen Sie im folgenden Beispiel *`FleetName`* und *`custom-location-1`* durch Ihre eigenen Informationen.

Die Variable `custom-location-1` ist der Name des Speicherorts, der in dem Erstellen Sie einen benutzerdefinierten Standort Schritt erstellt wurde.

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

Beispielausgabe

```
{
    "FleetAttributes": {
        "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
        "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
        "Name": "HardwareAnywhere",
        "CreationTime": "2023-02-23T17:57:42.293000+00:00",
        "Status": "ACTIVE",
        "MetricGroups": [
            "default"
        ],
        "CertificateConfiguration": {
            "CertificateType": "DISABLED"
        },
        "ComputeType": "ANYWHERE"
    }
}
```

Registrieren Sie Ihren Computer

Verwenden Sie den `register-compute`Befehl, um Ihre Rechenressource in der von Ihnen erstellten Flotte zu registrieren. Ersetzen Sie den *`fleet-id`* durch den im vorherigen Schritt `fleet-id` zurückgegebenen oder Flotten-ARN, den Sie auf der Detailseite Ihrer Flotte in der Konsole finden. Ersetzen Sie *`compute-name`* das und *`ip-address`* durch die IP-Adresse Ihrer Rechenressource.

> **Note**
>
> Wir empfehlen, sowohl die `register-compute` `get-compute-auth-token` Befehle als auch von einem Skript- oder Prozessmanager unabhängig von deinem Spieleserver aufzurufen.

```
aws gamelift register-compute \
    --compute-name HardwareAnywhere \
    --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --ip-address 10.1.2.3 \
    --location custom-location-1
```

Beispielausgabe

```
{
    "Compute": {
        "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
        "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
        "ComputeName": "HardwareAnywhere",
        "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/
HardwareAnywhere",
        "IpAddress": "10.1.2.3",
        "ComputeStatus": "Active",
        "Location": "custom-location-1",
        "CreationTime": "2023-02-23T18:09:26.727000+00:00",
        "GameLiftServiceSdkEndpoint": "wss://us-east-1.api.amazongamelift.com"
    }
}
```

Führe einen Serverprozess aus

1. Rufen Sie das Authentifizierungstoken für Ihre Rechenressource aus der Flotte ab, die Sie erstellt haben.

   Ihr Spieleserver verwendet das Authentifizierungstoken, um sich bei Amazon GameLift zu authentifizieren. Jedes Authentifizierungstoken hat eine Ablaufzeit. Um die Rechenressource weiterhin zum Hosten deines Spieleservers zu verwenden, rufe vor Ablauf ein neues Authentifizierungstoken ab.

> **Note**
>
> Amazon GameLift empfiehlt, sowohl die `register-compute` `get-compute-auth-token` Befehle als auch von einem Skript- oder Prozessmanager unabhängig von Ihrem Spieleserver aufzurufen.

Ersetzen Sie im folgenden Beispiel die durch den `fleet-id` ARN oder die Flotten-ID der Flotte, die in den vorherigen Schritten erstellt wurde. Ersetzen Sie das `compute-name` durch den Namen des Computers, den Sie mit dem `register-compute` Befehl in einem vorherigen Schritt erstellt haben.

```
aws gamelift get-compute-auth-token \
    --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --compute-name HardwareAnywhere
```

Beispielausgabe:

```
{
    "FleetId": "fleet-cebb4da2-52a8-4c27-9b85-587f945c6445",
    "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445",
    "ComputeName": "HardwareAnywhere",
    "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/
HardwareAnywhere",
    "AuthToken": "0c728041-3e84-4aaa-b927-a0fb202684c0",
    "ExpirationTimestamp": "2023-02-23T18:47:54+00:00"
}
```

2. Führe eine Instanz der ausführbaren Datei deines Gameservers aus.

   Um deinen Gameserver zu starten, initialisiere deinen Gameserver, indem du deine Serverparameter aufrufst `InitSDK()` und ihm übergibst. Weitere Informationen finden Sie unter ServerParameters.

   Server-SDK-Eingabe:

   ```
   //Define the server parameters
   ```

```
ServerParameters serverParameters = new ServerParameters(
    webSocketUrl=wss://us-east-1.api.amazongamelift.com,
    processId=PID1234,
    hostId=HardwareAnywhere,
    fleetId=arn:aws:gamelift:us-east-1:111122223333:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445,
    authToken=0c728041-3e84-4aaa-b927-a0fb202684c0);

//InitSDK establishes a connection with GameLift's websocket server for
 communication.
var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

3. Wenn der Serverprozess bereit ist, eine Spielsitzung zu hosten, rufen Sie Amazon `ProcessReady()` von Ihrem Spieleserver aus an GameLift. Weitere Informationen zu Prozessparametern finden Sie unter ProcessParameters

```
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
   this.OnStartGameSession,
   this.OnProcessTerminate,
   this.OnHealthCheck,
   this.OnUpdateGameSession,
   port=1024,
   new LogParameters(new List<string>()          // Examples of log and error files
 written by the game server
   {
      "C:\\game\\logs",
      "C:\\game\\error"
   })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

Spielsitzungen erstellen

1. Fügen Sie Ihrem Spieleserver Logik hinzu, sodass Ihr Serverprozess auf die `onStartGameSession()` Nachricht mit reagiert`ActivateGameSession()`. Dieser Vorgang hat keine Parameter, sendet aber eine Bestätigung an Amazon, GameLift dass Ihr Server die Nachricht „Spielsitzung erstellen“ empfangen und akzeptiert hat.

```
void OnStartGameSession(GameSession gameSession)
```

```
{
    // game-specific tasks when starting a new game session, such as loading map

    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

2. Starten Sie von Ihrem Game-Client-Backend-Service aus Ihre Spielsitzung mit dem Befehl start-matchmaking, start-game-session-placement, oder. create-game-session

```
aws gamelift create-game-session \
    --fleet-id arn:aws:gamelift:us-east-1:682428703967:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445 \
    --name GameSession1 \
    --maximum-player-session-count 2 \
    --location custom-location-1
```

Beispielausgabe:

```
GameSession {
  FleetId = arn:aws:gamelift:us-east-1:682428703967:fleet/fleet-
cebb4da2-52a8-4c27-9b85-587f945c6445,
  GameSessionId = 4444-4444,
  Name = GameSession1,
  Location = custom-location-1,
  IpAddress = 10.2.3.4,
  Port = 1024,
  ...
}
```

Amazon GameLift sendet eine onStartGameSession() Nachricht an Ihren registrierten Serverprozess. Die Nachricht enthält das GameSession Objekt aus dem vorherigen Schritt mit Spieleigenschaften, Spielsitzungsdaten, Matchmaker-Daten und mehr über die Spielsitzung.

3. Wenn die Spielsitzung abgeschlossen ist, beenden Sie den Spielserver-Prozess.

Server-SDK-Eingabe:

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
   Environment.Exit(0);
// otherwise, exit with error code
```

```
Environment.Exit(errorCode);
```

4. Starte einen weiteren Spieleserver-Prozess, indem du aufrufst`ProcessReady(processParams)`.

Migrieren Sie zu verwaltetem EC2

Nachdem Sie Ihren Spieleserver entwickelt haben und bereit sind, sich auf die Produktion vorzubereiten, können Sie Amazon Ihre Hardware GameLift verwalten lassen. Um zu einer verwalteten EC2-Flotte zu migrieren, laden Sie Ihren Build auf Amazon hoch GameLift und erstellen Sie eine verwaltete EC2-Flotte. Weitere Informationen zum Hochladen Ihres Builds und zum Einrichten einer Flotte finden Sie unter und. Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift Erstellen Sie eine von Amazon GameLift verwaltete Flotte

Verwalten Sie Ihre GameLift Amazon-Flotten

Verwenden Sie die GameLift Amazon-Konsole oder die AWS CLI, um Ihre Flotteneinstellungen zu aktualisieren, Remote-Standorte zu ändern oder eine Flotte zu löschen.

Aktualisieren Sie eine Flottenkonfiguration

Sie können veränderliche Flottenattribute, Porteinstellungen und Laufzeitkonfigurationen mithilfe der GameLift Amazon-Konsole oder der AWS CLI aktualisieren. Informationen zum Ändern der Skalierungsgrenzen finden Sie unterAutomatische Skalierung der Flottenkapazität mit Amazon GameLift.

Amazon GameLift console

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Flotten aus.
2. Wählen Sie die Flotte aus, die Sie aktualisieren möchten. Eine Flotte muss sich im `ACTIVE` Status befinden, bevor Sie sie bearbeiten können.
3. Wählen Sie auf der Flottendetailseite in einem der folgenden Abschnitte die Option Bearbeiten aus.
   - Flotteneinstellungen
     - Ändern Sie Flottenattribute wie Name (Name) und Description (Beschreibung).
     - Fügen Sie Metrikgruppen hinzu oder entfernen Sie sie, die Amazon CloudWatch verwendet, um aggregierte GameLift Amazon-Metriken für mehrere Flotten zu verfolgen.

- Aktualisieren Sie die Einstellungen für die Ressourcenerstellung.
- Aktivieren oder deaktivieren Sie den Spielsitzungsschutz.

- Laufzeitkonfiguration — Sie können jede der folgenden Einstellungen Ihrer Laufzeitkonfigurationen ändern und Laufzeitkonfigurationen hinzufügen oder entfernen.
  - Ändere den Startpfad deines Gameservers.
  - Fügen Sie optionale Startparameter hinzu, entfernen oder ändern Sie sie.
  - Ändere die Anzahl der gleichzeitigen Prozesse, die deine Spieleserver ausführen.
- Aktivierung von Spielsitzungen — Ändere, wie Serverprozesse ausgeführt und Spielsitzungen hosten sollen, indem du Max. Aktivierungen für gleichzeitige Spielsitzungen und Neues Aktivierungs-Timeout aktualisierst.
- EC2-Porteinstellungen — Aktualisieren Sie die IP-Adressen und Portbereiche, die eingehenden Zugriff auf die Flotte ermöglichen.

4. Wählen Sie Bestätigen, um die Änderungen zu speichern.

AWS CLI

Verwenden Sie die folgenden AWS CLI-Befehle, um eine Flotte zu aktualisieren:

- update-fleet-attributes
- update-fleet-port-settings
- update-runtime-configuration

Flottenstandorte aktualisieren

Sie können die Remote-Standorte einer Flotte mithilfe der GameLift Amazon-Konsole oder der AWS CLI hinzufügen oder entfernen. Sie können die Heimatregion einer Flotte nicht ändern.

Amazon GameLift console

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Flotten aus.
2. Wählen Sie die Flotte aus, die Sie aktualisieren möchten. Eine Flotte muss sich im `ACTIVE` Status befinden, bevor Sie sie bearbeiten können.
3. Wählen Sie auf der Flottendetailseite den Tab Standorte, um die Standorte der Flotte anzuzeigen.

4. Um neue Remote-Standorte hinzuzufügen, wählen Sie Hinzufügen und wählen Sie die Standorte aus, an denen Sie Instances bereitstellen möchten. Diese Liste enthält keine Instances, bei denen der Instance-Typ der Flotte nicht verfügbar ist.
5. Wenn neue Standorte ausgewählt sind, wählen Sie Hinzufügen. Amazon GameLift fügt die neuen Standorte zur Liste hinzu, wobei der Status auf gesetzt ist`NEW`. Amazon beginnt GameLift dann mit der Bereitstellung einer Instance an jedem hinzugefügten Standort und bereitet sie für die Ausrichtung von Spielesitzungen vor.
6. Um bestehende Remote-Standorte aus der Flotte zu entfernen, wählen Sie mithilfe der Kontrollkästchen einen oder mehrere aufgelistete Standorte aus.
7. Wenn eine oder mehrere Flotten ausgewählt sind, wählen Sie Entfernen. Die entfernten Standorte bleiben in der Liste, wobei der Status auf gesetzt ist`DELETING`. Amazon beginnt GameLift dann mit der Beendigung der Aktivität am entfernten Standort. Wenn es aktive Instances gibt, die Spielesitzungen hosten, GameLift verwendet Amazon den Prozess zur Kündigung des Spielservers, um Spielesitzungen ordnungsgemäß zu beenden, Spieleserver zu beenden und Instances herunterzufahren.

AWS CLI

Verwenden Sie die folgenden AWS CLI-Befehle, um die Flottenstandorte zu aktualisieren:

- create-fleet-locations
- delete-fleet-locations

Eine Flotte löschen

Sie können eine Flotte löschen, wenn Sie sie nicht mehr benötigen. Durch das Löschen einer Flotte werden alle Daten, die mit Spiel- und Spielersitzungen verknüpft sind, sowie die gesammelten Metrikdaten dauerhaft entfernt. Alternativ können Sie die Flotte beibehalten, die automatische Skalierung deaktivieren und die Flotte manuell auf 0 Instances skalieren.

> **Note**
>
> Wenn die Flotte über eine VPC-Peering-Verbindung verfügt, fordern Sie zunächst eine Autorisierung an, indem Sie anrufen. CreateVpcPeeringAuthorization Amazon GameLift löscht die VPC-Peering-Verbindung beim Löschen der Flotte.

Sie können entweder die GameLift Amazon-Konsole oder das AWS CLI-Tool verwenden, um eine Flotte zu löschen.

Amazon GameLift console

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Flotten aus.
2. Wählen Sie die Flotte aus, die Sie löschen möchten. Sie können Flotten nur in unserem `ERROR` Status `ACTIVE` löschen.
3. Wählen Sie Löschen.
4. Bestätigen Sie im Dialogfeld Flotte löschen den Löschvorgang, indem Sie Folgendes eingeben**`delete`**.
5. Wählen Sie Löschen.

AWS CLI

Verwenden Sie den folgenden AWS CLI-Befehl, um eine Flotte zu löschen:

- delete-fleet

Einer GameLift Amazon-Flotte einen Alias hinzufügen

Ein GameLift Amazon-Alias wird verwendet, um eine Flottenbezeichnung zu abstrahieren. Flottenbezeichnungen teilen Amazon mit GameLift , wo nach verfügbaren Ressourcen gesucht werden muss, wenn neue Spielsitzungen für Spieler erstellt werden. Verwenden Sie Aliase anstelle bestimmter Flotten-IDs, um den Spielerverkehr nahtlos von einer Flotte auf eine andere umzuleiten, indem Sie den Zielstandort des Alias ändern.

Es gibt für Aliasnamen zwei Arten von Routing-Strategien:

- Einfach — Leitet Spieler-Traffic an eine bestimmte Flotten-ID weiter. Sie können die Flotten-ID für einen Alias jederzeit aktualisieren.
- Terminal — Leitet eine Nachricht zurück an den Client. Beispielsweise kannst du Spieler, die einen out-of-date Client verwenden, zu einem Ort weiterleiten, an dem sie ein Upgrade erhalten können.

Flotten haben eine begrenzte Lebensdauer, und es gibt mehrere Gründe, Flotten während der Laufzeit eines Spiels auszutauschen. Du kannst den Spielserver-Build einer Flotte nicht aktualisieren

oder bestimmte Eigenschaften von Rechenressourcen einer vorhandenen Flotte ändern. Erstelle stattdessen neue Flotten mit den Änderungen und wechsle dann die Spieler zu den neuen Flotten. Mit Alias wirkt sich der Flottenwechsel minimal auf Spiele aus und bleibt für die Spieler unsichtbar.

Aliase sind nützlich in Spielen, die keine Warteschlangen verwenden. Wenn Sie Flotten in einer Warteschlange wechseln möchten, können Sie einfach eine neue Flotte erstellen und der Warteschlange hinzufügen. Die alte Flotte entfernen Sie. Der Spieler bemerkt nichts davon. Im Gegensatz dazu müssen Spieleclients, die keine Warteschlangen verwenden, angeben, welche Flotte für die Kommunikation mit dem GameLift Amazon-Service verwendet werden soll. Ohne Aliase erfordert ein Flottenwechsel Aktualisierungen Ihres Spielcodes und möglicherweise die Verteilung eines aktualisierten Spielclients an die Spieler.

Bei der Aktualisierung der Flotten-ID, auf die ein Alias verweist, gibt es eine Übergangszeit von bis zu 2 Minuten, in der Spielsitzungen mit dem Alias auf der alten Flotte landen können.

Erstelle einen neuen Alias

Sie können einen Alias entweder mit der GameLift Amazon-Konsole, wie hier beschrieben, oder mit dem AWS CLI-Befehl create-alias erstellen.

1. Wählen Sie in der GameLift Amazon-Konsole im Navigationsbereich Aliases aus.
2. Wählen Sie auf der Seite Aliases Create Alias. Wir empfehlen, den Flottentyp in Ihre Aliasnamen aufzunehmen. Dies macht es viel einfacher, den Flottentyp zu identifizieren, wenn Sie sich eine Liste von Aliasnamen ansehen.
3. Gehen Sie auf der Seite Alias erstellen unter Aliasdetails wie folgt vor:
    a. Geben Sie unter Name einen Aliasnamen ein.
    b. Geben Sie als Beschreibung eine kurze Beschreibung zur Identifizierung ein.
    c. Wählen Sie den Routingtyp Einfach oder Terminal.
4. (Optional) Fügen Sie dem Alias unter Tags Tags Tags hinzu, indem Sie Schlüssel - und Wertepaare eingeben.
5. Wählen Sie Erstellen.

Bearbeiten Sie einen Alias

Sie können einen Alias mit der GameLift Amazon-Konsole oder mit dem AWS CLI-Befehl update-alias bearbeiten.

1. Wählen Sie in der GameLift Amazon-Konsole im Navigationsbereich Aliases aus.
2. Wählen Sie auf der Seite Aliase den Alias aus, den Sie bearbeiten möchten.
3. Wählen Sie auf der Alias-Seite Bearbeiten aus.
4. Auf der Seite Edit alias können Sie die folgenden Felder bearbeiten:

   - Aliasname — Benutzerfreundlicher Name für Ihren Alias.
   - Beschreibung — Kurzbeschreibung für Ihren Alias.
   - Typ — Routing-Strategie für den Spielerverkehr. Wählen Sie Simple aus, um die verknüpfte Flotte zu ändern, oder Terminal, um die Beendigungsnachricht zu bearbeiten.
5. Wählen Sie Save Changes.

Probleme mit der GameLift Amazon-Flotte beheben

Dieses Thema enthält Anleitungen zu Problemen mit der Flottenkonfiguration für eine von Amazon GameLift verwaltete Hosting-Lösung. Für eine zusätzliche Fehlerbehebung können Sie per Fernzugriff auf eine Flotten-Instance zugreifen, sobald die Flotte aktiv ist. Siehe Remote-Verbindung zu Amazon- GameLift Flotten-Instances.

Probleme bei der Erstellung von Flotten

Wenn eine Flotte erstellt wird, initiiert der GameLift Amazon-Dienst einen Workflow, der an jedem Standort der Flotte eine neue Instanz bereitstellt und diese für den Betrieb Ihrer Spieleserver vorbereitet. Eine ausführliche Beschreibung finden Sie unterSo funktioniert die GameLift Amazon-Flottenerstellung. Eine Flotte kann keine Spielsitzungen und Spieler hosten, bis sie den Status Aktiv erreicht hat. In diesem Abschnitt werden die häufigsten Probleme erörtert, die verhindern, dass Flotten aktiv werden.

Herunterladen und Validieren

In dieser Phase kann die Flottenerstellung fehlschlagen, wenn Probleme mit den extrahierten Build-Dateien auftreten, das Installationsskript nicht ausgeführt wird oder wenn die in der Laufzeitkonfiguration angegebenen ausführbaren Dateien nicht in den Build-Dateien enthalten sind. Amazon GameLift stellt zu jedem dieser Probleme Protokolle zur Verfügung.

Wenn die Protokolle kein Problem anzeigen, ist es möglich, dass das Problem auf einen internen Service-Fehler zurückzuführen ist. In diesem Fall sollten Sie versuchen, die Flotte erneut zu erstellen. Wenn das Problem weiterhin besteht, sollten Sie den Spiel-Build erneut hochladen (falls die Dateien

beschädigt sind). Sie können sich auch an den GameLift Amazon-Support wenden oder eine Frage im Forum stellen.

Erstellung

Probleme, die zu Fehlern während der Build-Phase führen, sind beinahe sicher auf Probleme mit den Dateien des Spiel-Builds und/oder dem Installationsskript zurückzuführen. Stellen Sie sicher, dass Ihre auf Amazon hochgeladenen Game-Build-Dateien auf einem Computer installiert werden könnenGameLift, auf dem das entsprechende Betriebssystem ausgeführt wird. Stellen Sie sicher, dass Sie eine saubere Betriebssysteminstallation und keine vorhandene Entwicklungsumgebung verwenden.

Aktivierung

Während der Aktivierungsphase treten Probleme mit der Flottenerstellung am häufigsten auf. In dieser Phase werden eine Reihe von Elementen getestet, darunter die Funktionsfähigkeit des Spielservers, die Einstellungen für die Laufzeitkonfiguration und die Fähigkeit des Spielservers, mithilfe des Server-SDK mit dem GameLift Amazon-Dienst zu interagieren. Zu den häufigsten Problemen, die bei der Aktivierung der Flotte auftreten, gehören:

Die Serverprozesse können nicht gestartet werden.

Überprüfen Sie zunächst, ob Sie den Startpfad und die optionalen Startparameter in der Laufzeitkonfiguration der Flotte korrekt festgelegt haben. Sie können die aktuelle Laufzeitkonfiguration der Flotte entweder auf der Flottendetailseite Details (,) oder indem AWS CLI Sie den Befehl aufrufen describe-runtime-configuration. Wenn die Laufzeitkonfiguration korrekt aussieht, überprüfen Sie, ob es Probleme im Zusammenhang mit den Build-Dateien Ihres Spiels und/oder dem Installationsskript gibt.

Die Serverprozesse werden gestartet, aber die Flotte wird nicht aktiviert.

Wenn Serverprozesse erfolgreich gestartet und ausgeführt werden, die Flotte jedoch nicht in den Status Aktiv wechselt, liegt eine wahrscheinliche Ursache darin, dass der Serverprozess Amazon nicht darüber informiertGameLift, dass er bereit ist, Spielesitzungen zu veranstalten. Stellen Sie sicher, dass Ihr Spiel-Server die Server-API-Aktion `ProcessReady()` ordnungsgemäß aufruft (siehe Initialisieren Sie den Serverprozess).

VPC-Peering-Verbindungsanforderung fehlgeschlagen.

Für Flotten, die mit einer VPC-Peering-Verbindung erstellt werden (siehe So richten Sie VPC-Peering mit einer neuen Flotte ein), erfolgt das VPC-Peering während dieser Activating

(Aktivierung) Phasen. Wenn ein VPC-Peering aus irgendeinem Grund fehlschlägt, kann die neue Flotte nicht in den Status Active (Aktiv) wechseln. Sie können den Erfolg oder Misserfolg der Peering-Anfrage nachverfolgen, indem Sie anrufen describe-vpc-peering-connections. Stellen Sie sicher, dass eine gültige VPC-Peering-Autorisierung vorhanden ist (describe-vpc-peering-authorizations, da Autorisierungen nur 24 Stunden gültig sind.

Probleme mit dem Serverprozess

Die Serverprozesse starten, werden jedoch nach kurzer Zeit wieder beendet oder laufen nicht richtig rund.

Ein solches Systemverhalten muss nicht zwangsläufig auf Probleme mit Ihrem Spiele-Build hinweisen, sondern es kann sich auch ergeben, wenn auf der Instance zu viele Serverprozesse gleichzeitig ausgeführt werden. Die optimale Anzahl an gleichzeitigen Prozessen hängt gleichermaßen von den Instance-Typ und den Ressourcenanforderungen Ihrer Spiel-Server ab. Verringern Sie die Anzahl der gleichzeitigen Prozesse, die in der Laufzeitkonfiguration der Flotte eingestellt wird, um zu prüfen, ob sich die Leistung verbessert. Sie können die Laufzeitkonfiguration einer Flotte entweder über die GameLift Amazon-Konsole (bearbeiten Sie die Kapazitätszuweisungseinstellungen der Flotte) oder indem Sie den AWS CLI Befehl aufrufen update-runtime-configuration.

Probleme bei der Flottenlöschung

Die Flotte kann aufgrund der maximalen Anzahl von Instances nicht beendet werden.

Die Fehlermeldung zeigt an, dass die zu löschende Flotte noch aktive Instances hat, was nicht zulässig ist. Sie müssen zunächst eine Flotte auf null aktive Instances reduzieren. Dies geschieht, indem Sie die gewünschte Anzahl der Instances der Flotte manuell auf "0" festlegen und dann darauf warten, dass die Skalierung greift. Stellen Sie sicher, dass Sie die automatische Skalierung deaktivieren, da dies den manuellen Einstellungen entgegenwirkt.

VPC-Aktionen sind nicht autorisiert.

Dieses Problem gilt nur für Flotten, für die Sie speziell VPC-Peering-Verbindungen erstellt haben (siehe. VPC-Peering für Amazon GameLift Dieses Szenario tritt auf, weil der Prozess des Löschens einer Flotte auch das Löschen der VPC der Flotte und aller VPC-Peering-Verbindungen beinhaltet. Sie müssen zuerst eine Autorisierung einholen, indem Sie die Amazon GameLift Service API CreateVpcPeeringAuthorization() aufrufen oder den AWS CLI-Befehl `create-vpc-`

`peering-authorization` verwenden. Sobald Sie die Berechtigung haben, können Sie die Flotte löschen.

Probleme mit der Serverflotte in Echtzeit

Zombie-Spielsitzungen: Sie haben ein Spiel gestartet und ausgeführt, aber sie enden nie.

Dieses Problem entsteht wahrscheinlich in einem der folgenden Szenarien:

- Skript-Updates werden nicht von den Echtzeitservern der Flotte empfangen.
- Die Flotte erreicht schnell die maximale Kapazität und verkleinert sich nicht, wenn die Spieleraktivitäten (z. B. neue Spielsitzungsanfragen) abnimmt.

Dies ist mit ziemlicher Sicherheit darauf zurückzuführen, dass Ihr Realtime-Skript nicht erfolgreich aufgerufen `processEnding` wurde. Obwohl die Flotte aktiv wird und Spielsitzungen gestartet werden, gibt es keine Möglichkeit, diese anzuhalten. Das hat zur Folge, dass der Realtime-Server, auf dem die Spielsitzung läuft, nie freigegeben wird, um eine neue zu starten, und neue Spielsitzungen können nur gestartet werden, wenn neue Realtime-Server eingerichtet werden. Darüber hinaus wirken sich Aktualisierungen des Realtime-Skripts nicht auf bereits laufende Spielsitzungen aus, sondern nur auf einzelne.

Um dies zu verhindern, müssen Scripts einen Mechanismus bereitstellen, der einen `processEnding`-Aufruf auslöst. Wie in der Beispiel für ein Realtime Server-Skript veranschaulicht, besteht eine Möglichkeit darin, einen Idle-Timeout zu programmieren, bei dem das Script die aktuelle Spielsitzung beendet, wenn für eine bestimmte Zeitspanne kein Spieler verbunden ist.

Wenn Sie jedoch in dieses Szenario geraten, gibt es einige Problemumgehungen, um Ihre Echtzeitserver zu lösen. Der Trick besteht darin, den Neustart der Echtzeit-Serverprozesse — oder der zugrunde liegenden Flotteninstanzen — auszulösen. In diesem Fall werden die Spielsitzungen GameLift automatisch für Sie geschlossen. Sobald die Echtzeit-Server frei sind, können sie mit der neuesten Version des Realtime-Skripts neue Spielsitzungen starten.

Es gibt eine Reihe von Methoden, um dies zu erreichen (abhängig davon, wie umfassend das Problem ist):

- Skalieren Sie die gesamte Flotte herunter. Diese Methode ist die einfachste, hat aber eine weitreichende Wirkung. Skalieren Sie die Flotte auf null Instances, warten Sie, bis die Flotte vollständig herunterskaliert ist, und skalieren Sie sie dann wieder hoch. Dadurch werden alle

bestehenden Spielsitzungen gelöscht und Sie können mit dem zuletzt aktualisierten Echtzeit-Skript von vorne beginnen.

- Remotezugriff auf die Instance und Neustart des Prozesses. Dies ist eine gute Option, wenn Sie nur wenige Prozesse reparieren müssen. Wenn Sie bereits an der Instance angemeldet sind (z. B. um Protokolle zu verfolgen oder zum debuggen), dann kann dies die schnellste Methode sein. Siehe Remote-Verbindung zu Amazon- GameLift Flotten-Instances.

Wenn du dich dafür entscheidest, `processEnding` in deinem Echtzeit-Skript keine Art zu rufen, gibt es einige knifflige Situationen, die auftreten können, selbst wenn die Flotte aktiv wird und Spielsitzungen gestartet werden. Erstens endet eine laufende Spielsitzung nicht. Infolgedessen ist der Serverprozess, der diese Spielsitzung ausführt, nie frei. Er kann somit keine neue Spielsitzung starten. Zweitens nimmt der Realtime-Server keine Skript-Updates auf.

Remote-Verbindung zu Amazon- GameLift Flotten-Instances

Sie können eine Verbindung zu jeder Instance in Ihren aktiven von Amazon GameLift verwalteten EC2-Flotten herstellen. Häufige Gründe für den Zugriff auf eine Instance sind:

- Beheben von Problemen mit Ihrer Spieleserver-Integration
- Optimieren Sie Ihre Laufzeitkonfiguration und andere flottenspezifische Einstellungen
- Erhalten Sie Echtzeit-Spielserveraktivitäten, z. B. Protokollverfolgung.
- Führen Sie Benchmarking-Tools mit tatsächlichem Spielerdatenverkehr aus.
- Untersuchen Sie bestimmte Probleme mit einer Spielsitzung oder einem Serverprozess.

Berücksichtigen Sie beim Herstellen einer Verbindung mit einer Instance die folgenden potenziellen Probleme:

- Sie können eine Verbindung zu Instances in aktiven Flotten herstellen. Nicht aktive Flotten, die aktivieren oder sich in einem Fehlerzustand befinden, sind möglicherweise für einen kurzen Zeitraum zugänglich. Hilfe zu Problemen bei der Flottenaktivierung finden Sie unter Probleme mit der GameLift Amazon-Flotte beheben.
- Das Herstellen einer Verbindung mit einer aktiven Instance wirkt sich nicht auf die Hosting-Aktivität der Instance aus. Die Instance startet und stoppt weiterhin Serverprozesse basierend auf der Laufzeitkonfiguration. Es aktiviert und hostet Spielsitzungen. Es kann als Reaktion auf ein Herunterskalierungsereignis oder ein anderes Ereignis heruntergefahren werden.

- Alle Änderungen, die Sie an Dateien oder Einstellungen auf der Instance vornehmen, können sich auf die aktiven Spielsitzungen und verbundenen Spieler der Instance auswirken.

In den folgenden Anweisungen wird beschrieben, wie Sie mithilfe der AWS Befehlszeilenschnittstelle (CLI) eine Remote-Verbindung zu einer Instance herstellen. Sie können auch programmgesteuerte Aufrufe mit dem AWS SDK tätigen, wie in der Amazon- GameLift Service-API-Referenz dokumentiert.

Erfassen von Instance-Daten

Sammeln Sie die folgenden Informationen:

- Die ID der Instance, mit der Sie eine Verbindung herstellen möchten. Sie können entweder die Instance-ID oder den ARN verwenden.
- Die Amazon GameLift Server SDK-Version, die auf der Instance verwendet wird. Das Server-SDK ist in den Spiele-Build integriert, der auf der Instance ausgeführt wird.

So rufen Sie Instance-Daten ab

Bei den folgenden Schritten wird davon ausgegangen, dass Sie über eine verwaltete EC2-Flotten-ID für die Instance verfügen, mit der Sie eine Verbindung herstellen möchten.

1. Rufen Sie den Datenverarbeitungsnamen ab.

   Rufen Sie list-compute für die verwaltete EC2-Flotte auf, um eine Liste aller aktiven Rechenvorgänge in der Flotte zu erhalten. Geben Sie für eine Flotte mit einem Standort die Flotten-ID oder den ARN an. Geben Sie für eine Flotte mit mehreren Standorten die Flotten-ID oder den ARN und einen Standort an. Für eine verwaltete EC2-Flotte sind Rechenvorgänge EC2-Instances und die zurückgegebene Eigenschaft `ComputeName` ist die Instance-ID. Beispielsweise:

   Anforderung

   ```
   aws gamelift list-compute \
     --fleet-id  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
     --location ""sa-east-1"
   ```

   Antwort

   ```
   {
   ```

```
  "ComputeList": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeName": "i-0abc12d3e45fa6b78",
      "IpAddress": "00.00.000.00",
      "DnsName":
 "b08444ki909kvqu6zpw3is24x5pyz4b6m05i3jbxvpk9craztu0lqrbbrbnbkks.uwp57060n1k6dnlnw49b78hg1
west-2.amazongamelift.com",
      "ComputeStatus": "Active",
      "Location": "sa-east-1",
      "CreationTime": "2023-07-09T22:51:45.931000-07:00",
      "OperatingSystem": "AMAZON_LINUX",
      "Type": "c4.large"
    }
  ]
}
```

2. Suchen Sie die Server-SDK-Version.

   Die Server-SDK-Version ist ein Attribut einer Build-Ressource.

   a. Rufen Sie describe-fleet-attributes mit einer Flotten-ID auf, um die Build-ID und den ARN der Flotte abzurufen.

   b. Rufen Sie describe-build mit der Build-ID oder dem ARN auf, um die Server-SDK-Version des Builds abzurufen.

      Beispielsweise:

      Anforderung

```
aws gamelift describe-fleet-attributes /
  --fleet-ids  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

   Antwort

```
{
  "FleetAttributes": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
```

```
      "ComputeType": "EC2",
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",
      . . .
    }
  ]
}
```

Anforderung

```
aws gamelift describe-build /
  --build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```

Antwort

```
"Build": {
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "Name": "My_Game_Server_Build_One",
  "OperatingSystem": "AMAZON_LINUX_2",
  "ServerSdkVersion": "5.1.1",
  . . .
}
```

Herstellen einer Verbindung mit einer Instance (Server-SDK 5)

Wenn auf der Instance, mit der Sie eine Verbindung herstellen möchten, ein Spiele-Build mit Server-SDK-Version 5.x ausgeführt wird, verwenden Sie die folgenden Anweisungen, um sich mit Amazon EC2 Systems Manager (SSM) mit der Instance zu verbinden. Sie können auf Remote-Instances zugreifen, die auf Windows oder Linux ausgeführt werden.

1. Fordern Sie Anmeldeinformationen für die Instance an. Wenn Sie einen Datenverarbeitungsnamen und eine Flotten-ID für die Instance haben, mit der Sie eine Verbindung herstellen möchten, rufen Sie aufget-compute-access. Bei Erfolg gibt Amazon einen Satz temporärer Anmeldeinformationen für den Zugriff auf die Instance GameLift zurück. Beispielsweise:

   Anforderung

   ```
   aws gamelift get-compute-access \
   --compute-name i-11111111a222b333c \
   ```

```
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa
--region us-west-2
```

Antwort

```
{
  "ComputeName": " i-11111111a222b333c ",
  "Credentials": {
    "AccessKeyId": " ASIAIOSFODNN7EXAMPLE ",
    "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY ",
    "SessionToken": " AQoDYXdzEJr...<remainder of session token>"
  },
  "FleetArn": " arn:aws:gamelift:us-west-2::fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa ",
  "FleetId": " fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa "
}
```

2. Exportieren Sie die Anmeldeinformationen für den Zugriff. Sie können die Anmeldeinformationen optional in Umgebungsvariablen exportieren und sie verwenden, um die AWS CLI für den Standardbenutzer zu konfigurieren. Weitere Informationen finden Sie unter Umgebungsvariablen zum Konfigurieren der AWS CLI im AWS Command Line Interface-Benutzerhandbuch.

```
export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
```

3. Stellen Sie eine Verbindung mit der Flotten-Instance her. Starten Sie eine SSM-Sitzung mit der Instance, mit der Sie eine Verbindung herstellen möchten. Geben Sie die AWS Region oder den Speicherort der Instance an. Weitere Informationen finden Sie unter Starten einer Sitzung (AWS-CLI) im Amazon EC2 Systems Manager-Benutzerhandbuch. Verwenden Sie die Anmeldeinformationen, die Sie in Schritt 1 erhalten haben. Beispielsweise:

```
aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2
```

Herstellen einer Verbindung mit einer Instance (Server-SDK 4.x oder früher)

Wenn auf der Instance, mit der Sie eine Verbindung herstellen möchten, ein Spiele-Build mit Server-SDK Version 4 oder früher ausgeführt wird, befolgen Sie die folgenden Anweisungen. Sie können

eine Verbindung zu Instances herstellen, auf denen Windows oder Linux ausgeführt wird. Stellen Sie über einen Remote Desktop Protocol (RDP)-Client eine Verbindung zu einer Windows-Instance her. Stellen Sie über einen SSH-Client eine Verbindung zu einer Linux-Instance her.

1. Fordern Sie Anmeldeinformationen für die Instance an. Wenn Sie über eine Instance-ID verfügen, verwenden Sie den Befehl , get-instance-access um Anmeldeinformationen für den Zugriff anzufordern. Bei Erfolg gibt Amazon das Betriebssystem, die IP-Adresse und einen Satz von Anmeldeinformationen (Benutzername und geheimer Schlüssel) der Instance GameLift zurück. Das Format der Anmeldeinformationen ist vom Betriebssystem der Instance abhängig. Verwenden Sie die folgenden Anweisungen, um Anmeldeinformationen für RDP oder SSH abzurufen.

   - Für Windows-Instances – Für die Verbindung mit einer Windows-Instance benötigt RDP einen Benutzernamen und ein Passwort. Die `get-instance-access`-Anforderung gibt diese Werte als einfache Zeichenfolgen zurück. Daher können Sie die zurückgegebenen Werte unverändert verwenden. Beispiel für Anmeldeinformationen:

     ```
     "Credentials": {
         "Secret": "aA1bBB2cCCd3EEE",
         "UserName": "gl-user-remote"
     }
     ```

   - Für Linux-Instances – Um eine Verbindung zu einer Linux-Instance herzustellen, benötigt SSH einen Benutzernamen und einen privaten Schlüssel. Amazon GameLift gibt private RSA-Schlüssel aus und gibt sie als einzelne Zeichenfolge zurück, wobei das Zeilenumbrüche durch das Zeilenumbruchzeichen (\n) angezeigt werden. Um den privaten Schlüssel verwendbar zu machen, führen Sie die folgenden Schritte aus: (1) Konvertieren der Zeichenfolge in eine `.pem` Datei und (2) Festlegen von Berechtigungen für die neue Datei. Beispiel für zurückgegebene Anmeldeinformationen:

     ```
     "Credentials": {
         "Secret": "-----BEGIN RSA PRIVATE KEY-----
     nEXAMPLEKEYKCAQEAy7WZhaDsrA1W3mRlQtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
     \nvBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7iO5dSrvC7dQkW2duV5QuUdE0QW
     \nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
     \nG50TCFeOzfl8dqqCP6GzbPaIjiU19xX/azOR9V+tpUOzEL+wmXnZt3/nHPQ5xvD2OJH67km6SuPW
     \noPzev/D8V+x4+bHthfSjR9Y7DvQFjfBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrqu
     \n/uler7vgIn5m7lN5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
     ufGxbL1\nmb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/
     GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
     ```

```
BOsShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA3OzdXzMQexXVJ1TLZVEH0E7bhlY9d8O1ozR
\noQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfql
+lIp1\nYkriL0DbLXlvRAH+yHPRit2hHOjtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh
+GTfbjsXsOxQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ5OBEKzet9XcKwONBYELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWVOEihvm+xTtmaZlSp//lkq75XDwnU
\nWA8gkn6O3QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
\ngYBjbO+OZk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\noMo7yykABY7Ozd5wQewBQ4AdSlWSX4nGDtsiFxWiI5sKuAAeOCbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vvwKBgF+09VI/1wJBirsDGz9whVWfFPrTkJNvJZzYt69qezxlsjgFKshy
\nWBhd4xHZtmCqpBPlAymEjr/TOlbxyARmXMnIOWIAnNXMGB4KGSyl1mzSVAoQ+fqR+cJ3d0dyPl1j
\njjb0Ed/NY8frlNDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0iOegLDa
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjkpUfVTrW0YFjXkfcrR/V+QFL5OndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
 RSA PRIVATE KEY-----",
    "UserName": "gl-user-remote"
}
```

Wenn Sie die AWS CLI verwenden, können Sie automatisch eine `.pem` Datei generieren, indem Sie die Parameter --query und ---output in Ihre `get-instance-access` Anfrage aufnehmen.

Um für die `.pem`-Datei Berechtigungen festzulegen, führen Sie den folgenden Befehl aus:

```
$ chmod 400 MyPrivateKey.pem
```

2. Öffnen Sie einen Port für die Remote-Verbindung. Sie können über jeden Port, der in der Flottenkonfiguration autorisiert ist, auf Instances in Amazon- GameLift Flotten zugreifen. Sie können die Port-Einstellungen einer Flotte mit dem Befehl `describe-fleet-port-settings` anzeigen.

   Als bewährte Methode wird empfohlen, Ports für den Remote-Zugriff nur zu öffnen, wenn Sie diese benötigen, und sie zu schließen, wenn Sie diese nicht mehr benötigen. Sie können die Porteinstellungen nicht aktualisieren, nachdem Sie eine Flotte erstellt haben, aber bevor sie aktiv ist. Wenn Sie hängen bleiben, erstellen Sie die Flotte mit den offenen Porteinstellungen neu.

   Verwenden Sie den Befehl `update-fleet-port-settings`, um eine Port-Einstellung für die Remote-Verbindung hinzuzufügen (z. B. 22 für SSH oder 3389 für RDP). Geben Sie als Wert für den IP-Adressbereich die IP-Adressen für die Geräte an, die Sie verwenden möchten, um eine Verbindung herzustellen (in das CIDR-Format konvertiert). Beispiel:

```
$ AWS gamelift update-fleet-port-settings
    --fleet-id  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
    --inbound-permission-authorizations
 "FromPort=22,ToPort=22,IpRange=54.186.139.221/32,Protocol=TCP"
```

Im folgenden Beispiel wird Port 3389 auf einer Windows-Flotte geöffnet

```
$ AWS gamelift update-fleet-port-settings
--fleet-id  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
    --inbound-permission-authorizations
 "FromPort=3389,ToPort=3389,IpRange=54.186.139.221/32,Protocol=TCP"
```

3. Öffnen Sie einen Remote-Verbindungs-Client. Verwenden Sie Remote Desktop für Windows-Instances oder SSH für Linux-Instances. Stellen Sie die Verbindung mit der Instance unter Verwendung von IP-Adresse, Port-Einstellung und Anmeldeinformationen her.

   SSH-Beispiel:

```
ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
```

Anzeigen von Dateien auf Remote-Instances

Wenn Sie remote mit einer Instance verbunden sind, verfügen Sie über einen vollständigen Benutzer- und Administratorzugriff. Das bedeutet, dass Sie auch Fehler und Ausfälle für das Spiele-Hosting verursachen können. Wenn die Instance Spiele mit aktiven Spielern hostet, besteht das Risiko, dass Spielsitzungen abstürzen und Spieler ablegen oder Prozesse zum Herunterfahren von Spielen unterbrechen und Fehler in gespeicherten Spieldaten und -protokollen verursachen.

Suchen Sie auf einer Hosting-Instance nach diesen Ressourcen:

- Build-Dateien von Spielen. Diese Dateien sind der Spiele-Build, den Sie in Amazon hochgeladen haben GameLift. Sie enthalten eine oder mehrere ausführbare Dateien, Komponenten und Abhängigkeiten des Spieleservers. Spiele-Build-Dateien befinden sich in einem Stammverzeichnis namens `game`:
  - Bei Windows: `c:\game`
  - Bei Linux: `/local/game`

- Protokolldateien von Spielen. Suchen Sie die Protokolldateien, die Ihr Spieleserver generiert, im `game` Stammverzeichnis unter dem von Ihnen angegebenen Verzeichnispfad.
- Amazon GameLift -Hosting-Ressourcen. Das Stammverzeichnis `Whitewater` enthält Dateien, die vom Amazon- GameLift Service zur Verwaltung von Spiel-Hosting-Aktivitäten verwendet werden. Ändern Sie diese Dateien nicht aus irgendeinem Grund.
- Laufzeitkonfiguration. Greifen Sie nicht auf die Laufzeitkonfiguration für einzelne Instances zu. Um Änderungen an einer Laufzeitkonfigurationseigenschaft vorzunehmen, aktualisieren Sie die Laufzeitkonfiguration der Flotte (siehe SDKAWS-Operation UpdateRuntimeConfiguration oder AWS CLI update-runtime-configuration).
- Flottendaten. Eine JSON-Datei enthält Informationen über die Flotte, zu der die Instance gehört, zur Verwendung durch Serverprozesse, die auf der Instance ausgeführt werden. Die JSON-Datei befindet sich am folgenden Speicherort:
  - Bei Windows: `C:\GameMetadata\gamelift-metadata.json`
  - Bei Linux: `/local/gamemetadata/gamelift-metadata.json`
- TLS-Zertifikate. Wenn sich die Instance auf einer Flotte befindet, für die die TLS-Zertifikatsgenerierung aktiviert ist, suchen Sie am folgenden Speicherort nach Zertifikatsdateien, einschließlich Zertifikat, Zertifikatkette, privatem Schlüssel und Stammzertifikat:
  - Bei Windows: `c:\\GameMetadata\Certificates`
  - Bei Linux: `/local/gamemetadata/certificates/`

Skalierung der GameLift Amazon-Hosting-Kapazität

Die Hosting-Kapazität, gemessen in Instanzen, gibt die Anzahl der Spielsitzungen an, die Amazon gleichzeitig hosten GameLift kann, und die Anzahl der gleichzeitigen Spieler, die diese Spielsitzungen aufnehmen können. Eine der schwierigsten Aufgaben beim Game-Hosting ist die Skalierung der Kapazität, um der Nachfrage der Spieler gerecht zu werden, ohne Geld für Ressourcen zu verschwenden, die Sie nicht benötigen. Weitere Informationen finden Sie unter Skalieren der Flottenkapazität.

Die Kapazität wird auf der Ebene des Flottenstandorts angepasst. Alle Flotten haben mindestens einen Standort: die AWS Heimatregion der Flotte. Bei der Anzeige oder Skalierung der Kapazität werden die Informationen nach Standort aufgelistet, einschließlich der Heimatregion der Flotte und aller weiteren entfernten Standorte.

Sie können die Anzahl der zu verwaltenden Instanzen manuell festlegen oder die automatische Skalierung einrichten, um die Kapazität dynamisch anzupassen, wenn sich die Anforderungen der Spieler ändern. Wir empfehlen, dass Sie zunächst die Option Target-based Auto Scaling aktivieren. Das Ziel der zielbasierten automatischen Skalierung besteht darin, genügend Hosting-Ressourcen bereitzustellen, um aktuellen Spielern gerecht zu werden, sowie ein wenig zusätzliche Ressourcen bereitzustellen, um unerwartete Nachfragespitzen der Spieler zu bewältigen. Für die meisten Spiele bietet die zielbasierte automatische Skalierung eine hocheffektive Skalierungslösung.

Die Themen in diesem Abschnitt bieten detaillierte Hilfe für die folgenden Aufgaben:

- Festlegen der unteren und oberen Grenzwerte für die Kapazitätsskalierung
- Manuelles Festlegen der Kapazitätsstufen
- Verwenden Sie zielgerichtete automatische Skalierung
- Regelbasierte automatische Skalierung verwalten (erweiterte Funktion)
- Deaktivieren Sie vorübergehend die automatische Skalierung

Sie können die meisten Aktivitäten zur Flottenskalierung mit der GameLift Amazon-Konsole durchführen. Sie können auch ein AWS SDK oder das AWS Command Line Interface (AWS CLI) mit der Amazon GameLift Service API verwenden.

Um die Flottenkapazität in der Konsole zu verwalten

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Detailseite der Flotte zu öffnen.
4. Wählen Sie den Tab Skalierung. Auf dieser Registerkarte können Sie:

    - Sehen Sie sich historische Skalierungsmetriken für die gesamte Flotte an.
    - Sehen und aktualisieren Sie die Kapazitätseinstellungen für jeden Flottenstandort, einschließlich Skalierungsgrenzen und aktuellen Kapazitätseinstellungen.
    - Aktualisieren Sie die zielbasierte automatische Skalierung, sehen Sie sich die regelbasierten Autoskalierungsrichtlinien an, die für die gesamte Flotte gelten, und setzen Sie die Auto-Scaling-Aktivitäten für jeden Standort aus.

Themen

Legen Sie GameLift Amazon-Kapazitätsgrenzen fest

Wenn Sie die Hosting-Kapazität für einen GameLift Amazon-Flottenstandort entweder manuell oder durch automatische Skalierung skalieren, sollten Sie die Skalierungsgrenzen des Standorts berücksichtigen. Alle Flottenstandorte haben eine Mindest- und Höchstgrenze, die die zulässige Reichweite für die Kapazität des Standorts definiert. Standardmäßig liegen die Limits für Flottenstandorte bei mindestens 0 Instances und maximal bei 1 Instance. Bevor Sie einen Flottenstandort skalieren können, passen Sie die Grenzwerte an.

Wenn Sie Auto Scaling verwenden, ermöglicht das maximale Limit Amazon die Skalierung eines Flottenstandorts, GameLift um der Nachfrage der Spieler gerecht zu werden. Dadurch werden jedoch außer Kontrolle geratene Hosting-Kosten vermieden, z. B. während eines DDOS-Angriffs. Richten Sie einen CloudWatchAmazon-Alarm ein, der Sie benachrichtigt, wenn sich die Kapazität dem Höchstwert nähert, sodass Sie die Situation beurteilen und bei Bedarf manuell anpassen können. (Sie können auch einen Abrechnungsalarm erstellen, um die AWS Kosten zu überwachen.) Das Mindestlimit ist nützlich, um die Verfügbarkeit des Hostings aufrechtzuerhalten, auch wenn die Nachfrage der Spieler gering ist.

Sie können Kapazitätslimits für die Standorte einer Flotte in der GameLiftAmazon-Konsole oder mithilfe der AWS Command Line Interface (AWS CLI) festlegen.

So legen Sie die Grenzwerte für die Flottenkapazität fest

Console

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Detailseite der Flotte zu öffnen.
4. Wählen Sie auf der Registerkarte Skalierung unter Skalierung der Kapazität einen Flottenstandort aus und wählen Sie dann Bearbeiten aus.
5. Legen Sie im Dialogfeld Skalierungskapazität bearbeiten die Anzahl der Instanzen für Minimale Größe, Gewünschte Instanzen und maximale Größe fest.

6. Wählen Sie Bestätigen aus.

AWS CLI

1. Überprüfen Sie die aktuellen Kapazitätseinstellungen. Verwenden Sie in einem Befehlszeilenfenster den describe-fleet-location-capacityBefehl mit der Flotten-ID und dem Standort, für den Sie die Kapazität ändern möchten. Dieser Befehl gibt ein FleetCapacityObjekt zurück, das die aktuellen Kapazitätseinstellungen des Standorts enthält. Stellen Sie fest, ob die neuen Instanzlimits der aktuellen Einstellung für die gewünschten Instanzen entsprechen.

```
aws gamelift describe-fleet-location-capacity \
    --fleet-id <fleet identifier> \
    --location <location name>
```

2. Aktualisieren der Limiteinstellungen. Verwenden Sie in einem Befehlszeilenfenster den update-fleet-capacityBefehl mit den folgenden Parametern. Sie können die Instance-Limits und die Anzahl der gewünschten Instances mit demselben Befehl anpassen.

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
--desired-instances <fleet capacity goal>
```

Beispiel:

```
aws gamelift update-fleet-capacity \
    --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --location us-west-2 \
    --max-size 10 \
    --min-size 1 \
    --desired-instances 10
```

Wenn Ihre Anfrage erfolgreich ist, GameLift gibt Amazon die Flottennummer zurück. Wenn der neue `max-size` oder `min-size` Wert mit der aktuellen `desired-instances` Einstellung in Konflikt steht, GameLift gibt Amazon einen Fehler zurück.

Manuelles Festlegen der Kapazität für eine GameLift Amazon-Flotte

Wenn Sie eine neue Flotte erstellen, legt Amazon die gewünschten Instances GameLift automatisch auf eine Instance an jedem Flottenstandort fest. Anschließend GameLift stellt Amazon an jedem Standort eine neue Instance bereit. Um die Flottenkapazität zu ändern, können Sie eine zielbasierte Auto-Scaling-Richtlinie hinzufügen oder die Anzahl der Instanzen, die Sie für einen Standort wünschen, manuell festlegen. Weitere Informationen finden Sie unter Skalieren der Flottenkapazität.

Das manuelle Festlegen der Kapazität einer Flotte kann nützlich sein, wenn Sie keine automatische Skalierung benötigen oder wenn Sie die Kapazität auf einem bestimmten Niveau halten müssen. Das manuelle Festlegen der Kapazität funktioniert nur, wenn Sie keine zielbasierte Auto-Scaling-Richtlinie verwenden. Wenn Sie über eine zielorientierte Auto-Scaling-Richtlinie verfügen, setzt diese auf der Grundlage ihrer eigenen Skalierungsregeln sofort die gewünschte Kapazität zurück.

Sie können die Kapazität manuell in der GameLift Amazon-Konsole oder mithilfe der AWS Command Line Interface (AWS CLI) festlegen. Der Status der Flotte muss aktiv sein.

Automatische Skalierung aussetzen

Sie können alle automatischen Skalierungsaktivitäten für jeden Flottenstandort aussetzen. Wenn die automatische Skalierung unterbrochen ist, bleibt die gewünschte Anzahl von Instanzen am Flottenstandort unverändert, sofern sie nicht manuell geändert wird. Wenn Sie die automatische Skalierung für einen Standort aussetzen, wirkt sich dies auf die aktuellen Richtlinien der Flotte und alle Richtlinien aus, die Sie möglicherweise in Zukunft definieren.

So legen Sie die Flottenkapazität manuell fest

Console

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Detailseite der Flotte zu öffnen.
4. Wählen Sie auf der Registerkarte Skalierung unter Gesperrte Standorte für die automatische Skalierung jeden Standort aus, für den Sie die automatische Skalierung aussetzen möchten, und wählen Sie dann Aussetzen aus.
5. Wählen Sie unter Skalierungskapazität einen Speicherort aus, den Sie manuell festlegen möchten, und wählen Sie dann Bearbeiten.

6. Geben Sie im Dialogfeld „Skalierungskapazität bearbeiten“ Ihren bevorzugten Wert für Gewünschte Instanzen ein und wählen Sie dann „Bestätigen“. Dadurch wird Amazon GameLift die Anzahl der Instances mitgeteilt, die in einem aktiven Zustand gehalten werden müssen, um Spielesitzungen zu hosten.

Amazon GameLift reagiert auf die Änderungen, indem es zusätzliche Instances bereitstellt oder nicht benötigte Instances herunterfährt. Sobald Amazon diesen Vorgang GameLift abgeschlossen hat, ändert sich die Anzahl der aktiven Instances am Standort, sodass sie dem aktualisierten Wert für die gewünschten Instances entspricht. Dieser Vorgang kann einige Zeit dauern.

AWS CLI

1. Überprüfen Sie die aktuellen Kapazitätseinstellungen. Verwenden Sie in einem Befehlszeilenfenster den describe-fleet-location-capacityBefehl mit der Flotten-ID und dem Standort, für den Sie die Kapazität ändern möchten. Dieser Befehl gibt ein FleetCapacityObjekt zurück, das die aktuellen Kapazitätseinstellungen des Standorts enthält. Stellen Sie fest, ob die Instanzlimits der neuen Einstellung für die gewünschten Instanzen entsprechen.

```
aws gamelift describe-fleet-location-capacity \
    --fleet-id <fleet identifier> \
    --location <location name>
```

2. Aktualisierung der gewünschten Kapazität. Verwenden Sie den update-fleet-capacityBefehl mit der Flotten-ID, dem Standort und einem neuen Wert für die gewünschten Instances. Wenn dieser Wert außerhalb des aktuellen Grenzbereichs liegt, können Sie die Grenzwerte im selben Befehl anpassen.

```
--fleet-id <fleet identifier>
--location <location name>
--desired-instances <fleet capacity as an integer>
--max-size <maximum capacity>    [Optional]
--min-size <minimum capacity>    [Optional]
```

Beispiel:

```
aws gamelift update-fleet-capacity \
    --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --location us-west-2 \
```

```
    --desired-instances 5 \
    --max-size 10 \
    --min-size 1
```

Wenn Ihre Anfrage erfolgreich ist, GameLift gibt Amazon die Flottennummer zurück. Wenn die neue Einstellung für die gewünschten Instances außerhalb der Mindest- und Höchstgrenzen liegt, GameLift gibt Amazon eine Fehlermeldung zurück.

Automatische Skalierung der Flottenkapazität mit Amazon GameLift

Verwenden Sie die automatische Skalierung in AmazonGameLift, um Ihre Flottenkapazität als Reaktion auf die Aktivität des Spielservers dynamisch zu skalieren. Wenn Spieler ankommen und Spielsitzungen beginnen, kann die automatische Skalierung weitere Instanzen hinzufügen. Wenn die Nachfrage der Spieler nachlässt, kann die automatische Skalierung nicht benötigte Instanzen beenden. Auto Scaling ist eine effektive Methode, um Ihre Hosting-Ressourcen und -Kosten zu minimieren und gleichzeitig ein reibungsloses, schnelles Spielererlebnis zu bieten.

Um Auto Scaling zu verwenden, erstellen Sie Skalierungsrichtlinien, die Amazon mitteilen, GameLift wann es nach oben oder unten skalieren soll. Es gibt zwei Arten von Skalierungsrichtlinien: zielgerichtete und regelbasierte. Der zielorientierte Ansatz — Zielverfolgung — ist eine Komplettlösung. Wir empfehlen es als die einfachste und effektivste Option. Bei regelbasierten Skalierungsrichtlinien müssen Sie jeden Aspekt des Entscheidungsprozesses für die automatische Skalierung definieren. Dies ist hilfreich, um bestimmte Probleme zu lösen. Diese Lösung eignet sich am besten als Ergänzung zur zielgerichteten automatischen Skalierung.

Sie können die zielbasierte automatische Skalierung mithilfe der GameLift Amazon-Konsole, der AWS Command Line Interface (AWS CLI) oder eines AWS SDK verwalten. Sie können die regelbasierte automatische Skalierung nur mit dem AWS CLI oder einem AWS SDK verwalten. Regelbasierte Skalierungsrichtlinien können Sie jedoch in der Konsole einsehen.

Themen

Zielgerichtete automatische Skalierung

Die zielorientierte automatische Skalierung für Amazon GameLift passt die Kapazitätsniveaus an die Flottenmetrik an. `PercentAvailableGameSessions` Diese Kennzahl stellt den verfügbaren Puffer der Flotte für einen plötzlichen Anstieg der Spielernachfrage dar.

Der primäre Grund für die Einrichtung eines Kapazitätspuffers ist die Wartezeit der Spieler. Wenn die Spielautomaten für Spielsitzungen bereit sind und warten, dauert es Sekunden, bis neue Spieler an Spielsitzungen teilnehmen. Wenn keine Ressourcen verfügbar sind, müssen die Spieler darauf warten, dass bestehende Spielsitzungen beendet werden, oder bis neue Ressourcen verfügbar sind. Das Starten neuer Instanzen und Serverprozesse kann Minuten dauern.

Wenn Sie die zielbasierte automatische Skalierung einrichten, geben Sie die Größe des Puffers an, den die Flotte verwalten soll. Da `PercentAvailableGameSessions` der Prozentsatz der verfügbaren Ressourcen gemessen wird, ist die tatsächliche Puffergröße ein Prozentsatz der gesamten Flottenkapazität. Amazon GameLift fügt Instances hinzu oder entfernt sie, um die Zielpuffergröße beizubehalten. Mit einem großen Puffer minimieren Sie die Wartezeit, zahlen aber auch für zusätzliche Ressourcen, die Sie möglicherweise nicht nutzen. Wenn Ihre Spieler toleranter gegenüber Wartezeiten sind, können Sie die Kosten senken, indem Sie einen kleinen Puffer einrichten.

Um eine zielgerichtete automatische Skalierung einzurichten

Console

1. Öffnen Sie die GameLiftAmazon-Konsole.
2. Wählen Sie im Navigationsbereich Hosting, Fleets aus.
3. Wählen Sie auf der Seite Flotten den Namen einer aktiven Flotte aus, um die Detailseite der Flotte zu öffnen.
4. Wählen Sie den Tab Skalierung. Diese Registerkarte zeigt die historischen Skalierungsmetriken der Flotte an und enthält Steuerelemente zum Anpassen der aktuellen Skalierungseinstellungen.
5. Überprüfen Sie unter Skalierung der Kapazität, ob die Grenzwerte für Mindestgröße und Maximalgröße für die Flotte angemessen sind. Wenn die automatische Skalierung aktiviert ist, passt sich die Kapazität zwischen diesen beiden Grenzwerten an.
6. Wählen Sie unter Target-based Auto Scaling Policy die Option Bearbeiten aus.
7. Geben Sie im Dialogfeld „Zielbasierte automatische Skalierungsrichtlinie bearbeiten“ unter Prozentuale verfügbare Spielsitzungen den Prozentsatz ein, den Sie beibehalten möchten,

und wählen Sie dann „Bestätigen“. Nachdem Sie die Einstellungen bestätigt haben, GameLift fügt Amazon unter Zielbasierte automatische Skalierungsrichtlinie eine neue zielbasierte Richtlinie hinzu.

AWS CLI

1. Festlegen der Grenzwerte für Flottenkapazität. Stellen Sie die Grenzwerte mit dem update-fleet-capacityBefehl ein. Weitere Informationen finden Sie unter Legen Sie GameLift Amazon-Kapazitätsgrenzen fest.
2. Eine neue Richtlinie erstellen. Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den put-scaling-policyBefehl mit den Parametereinstellungen Ihrer Richtlinie. Um eine bestehende Richtlinie zu aktualisieren, geben Sie den Namen der Richtlinie an und geben eine vollständige Version der aktualisierten Richtlinie an.

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--target-configuration <buffer size>
```

Beispiel:

```
aws gamelift put-scaling-policy \
    --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
    --name "My_Target_Policy_1" \
    --policy-type "TargetBased" \
    --metric-name "PercentAvailableGameSessions" \
    --target-configuration "TargetValue=5"
```

Automatische Skalierung mit regelbasierten Richtlinien

Regelbasierte Skalierungsrichtlinien in Amazon GameLift bieten eine detaillierte Kontrolle bei der automatischen Skalierung der Kapazität einer Flotte als Reaktion auf Spieleraktivitäten. Für jede Richtlinie können Sie die Skalierung mit einer von mehreren Flottenkennzahlen verknüpfen, einen Auslösepunkt identifizieren und das entsprechende Skalierungs- oder Scale-Down-Ereignis anpassen. Regelbasierte Richtlinien sind nützlich, um die zielgerichtete Skalierung zu ergänzen, um besonderen Umständen gerecht zu werden.

In einer regelbasierten Richtlinie heißt es wie folgt: „Wenn eine Flottenkennzahl für einen bestimmten Zeitraum einen Schwellenwert erreicht oder überschreitet, ändern Sie die Kapazität der Flotte um einen bestimmten Betrag.“ Dieses Thema beschreibt die Syntax zur Erstellung einer Richtlinienanweisung und bietet Hilfe bei der Erstellung und Verwaltung Ihrer regelbasierten Richtlinien.

Regelbasierte Richtlinien verwalten

Erstellen, aktualisieren oder löschen Sie regelbasierte Richtlinien mithilfe eines AWS SDK oder des AWS Command Line Interface (AWS CLI) mit der Amazon GameLift Service API. Sie können alle aktiven Richtlinien in der GameLift Amazon-Konsole einsehen.

Um vorübergehend alle Skalierungsrichtlinien für eine Flotte zu beenden, verwenden Sie den AWS CLI Befehl stop-fleet-actions.

So erstellen oder aktualisieren Sie eine regelbasierte Skalierungsrichtlinie ()AWS CLI:

1. Festlegen der Grenzwerte für Flottenkapazität. Stellen Sie mit dem update-fleet-capacityBefehl einen oder beide Grenzwerte ein. Weitere Informationen finden Sie unter Legen Sie GameLift Amazon-Kapazitätsgrenzen fest.
2. Eine neue Richtlinie erstellen. Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den put-scaling-policyBefehl mit den Parametereinstellungen Ihrer Richtlinie. Um eine bestehende Richtlinie zu aktualisieren, geben Sie den Namen der Richtlinie an und geben eine vollständige Version der aktualisierten Richtlinie an.

   ```
   --fleet-id <unique fleet identifier>
   --name "<unique policy name>"
   --policy-type <target- or rule-based policy>
   --metric-name <name of metric>
   --comparison-operator <comparison operator>
   --threshold <threshold integer value>
   --evaluation-periods <number of minutes>
   --scaling-adjustment-type <adjustment type>
   --scaling-adjustment <adjustment amount>
   ```

   Beispiel:

   ```
   aws gamelift put-scaling-policy \
       --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
       --name "Scale up when AGS<50" \
   ```

```
    --policy-type RuleBased \
    --metric-name AvailableGameSessions \
    --comparison-operator LessThanThreshold \
    --threshold 50 \
    --evaluation-periods 10 \
    --scaling-adjustment-type ChangeInCapacity \
    --scaling-adjustment 1
```

Um eine regelbasierte Skalierungsrichtlinie zu löschen, gehen Sie wie folgt vor: AWS CLI

- Öffnen Sie ein Befehlszeilenfenster und verwenden Sie den delete-scaling-policyBefehl mit der Flotten-ID und dem Richtliniennamen.

  Beispiel:

```
aws gamelift delete-scaling-policy \
    --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --name "Scale up when AGS<50"
```

Syntax für automatische Skalierungsregeln

Um eine regelbasierte Erklärung zur Skalierungspolitik zu erstellen, geben Sie sechs Variablen an:

Wenn für *<Metrikname>* gilt, dass der Wert *<Vergleichsoperator>* *<Schwellenwert>* für *<Testzeitraum>* unverändert bleibt, wird die Flottenkapazität unter Verwendung von *<Korrekturart>* auf/um *<Anpassungswert>* geändert.

Mit dieser Grundsatzerklärung wird beispielsweise immer dann ein Scale-up-Event gestartet, wenn die zusätzliche Kapazität einer Flotte unter dem liegt, was für 50 neue Spielsitzungen erforderlich ist:

If `AvailableGameSessions` remains at `less than 50` for `10 minutes`, then change fleet capacity using `ChangeInCapacity` by `1 instances`.

Metrikname

Um ein Skalierungsereignis zu starten, verknüpfen Sie eine Auto-Scaling-Richtlinie mit einer der folgenden flottenspezifischen Metriken. Vollständige Metrikbeschreibungen finden Sie unterGameLiftAmazon-Metriken für Flotten.

- Aktivieren der Spielsitzungen

- Aktive Spielsitzungen
- Verfügbare Spielsitzungen
- Verfügbare Spielsitzungen als Prozentwert
- Aktive Instances
- Verfügbare Spielersitzungen
- Aktuelle Spielersitzungen
- Instances im Leerlauf
- Instances im Leerlauf als Prozentwert

Befindet sich die Flotte in einer Warteschlange für Spielsitzungen, kannst du die folgenden Metriken verwenden:

- Warteschlangentiefe — Die Anzahl der ausstehenden Anfragen für Spielsitzungen, für die diese Flotte der beste verfügbare Hosting-Standort ist.
- Wartezeit — Flottenspezifische Wartezeit. Die Zeitspanne, wie lange die älteste ausstehende Spielsitzungsanfrage auf ihre Erfüllung gewartet hat. Die Wartezeit einer Flotte entspricht der Wartezeit der ältesten aktuellen Anforderung in der Warteschlange.

Vergleichsoperator

Teilt Amazon mit, GameLift wie die Metrikdaten mit dem Schwellenwert verglichen werden. Zu den gültigen Vergleichsoperatoren gehören größer als (**>**), kleiner als (**= <**), greater than or equal (**>**) und kleiner als oder gleich (**<=**).

Schwellenwert

Wenn der angegebene Metrikwert den Schwellenwert erreicht oder überschreitet, wird ein Skalierungsereignis ausgelöst. Dieser Wert ist immer eine positive Ganzzahl.

Auswertungszeitraum

Die Metrik muss den Schwellenwert für die gesamte Dauer des Bewertungszeitraums erreichen oder überschreiten, bevor ein Skalierungsereignis ausgelöst wird. Die Länge des Auswertungszeitraums ist fortlaufend; wenn die Metrik von dem Schwellenwert abweicht, beginnt der Auswertungszeitraum von neuem.

Anpassungstyp und -wert

Dieser Satz von Variablen legt zusammen fest, wie Amazon die Kapazität der Flotte anpassen GameLift soll, wenn ein Skalierungsereignis beginnt. Wählen Sie aus drei möglichen Anpassungsarten:

- Änderung der Kapazität — Erhöhen oder verringern Sie die aktuelle Kapazität um eine bestimmte Anzahl von Instanzen. Legen Sie den Anpassungswert für die Anzahl der Instances fest, um den die Flotte vergrößert oder verkleinert werden soll. Positive Werte fügen Instances hinzu, während negative Werte Instances entfernen. Ein Wert von „-10“ verkleinert die Flotte beispielsweise um 10 Instanzen, unabhängig von der Gesamtgröße der Flotte.
- Prozentuale Änderung der Kapazität — Erhöhen oder verringern Sie die aktuelle Kapazität um einen bestimmten Prozentsatz. Stellen Sie den Anpassungswert auf den Prozentsatz ein, um den Sie die Flottenkapazität erhöhen oder verringern möchten. Positive Werte fügen Instances hinzu, während negative Werte Instances entfernen. Beispielsweise werden bei einer Flotte mit 50 Instances durch eine prozentuale Änderung von „20" der Flotte 10 Instances hinzugefügt.
- Genaue Kapazität — Erhöhen oder verringern Sie die aktuelle Kapazität auf einen bestimmten Wert. Legen Sie den Anpassungswert auf die genaue Anzahl an Instances fest, die die Flotte beibehalten soll.

Tipps für regelbasiertes Auto-Scaling

Die folgenden Vorschläge können Ihnen helfen, das Beste aus der automatischen Skalierung mit regelbasierten Richtlinien herauszuholen.

Verwenden mehrerer Richtlinien

Sie können mehrere Auto-Scaling-Richtlinien für eine Flotte gleichzeitig verwenden. Das häufigste Szenario ist, dass eine zielorientierte Richtlinie die meisten Skalierungsanforderungen verwaltet und regelbasierte Richtlinien verwendet, um Randfälle zu behandeln. Der Verwendung mehrerer Richtlinien sind keine Grenzen gesetzt.

Bei mehreren Richtlinien verhält sich jede Richtlinie unabhängig. Es gibt keine Möglichkeit, die Reihenfolge der Skalierungsereignisse zu kontrollieren. Wenn Sie beispielsweise mehrere Richtlinien haben, die die Skalierung vorantreiben, ist es möglich, dass Spieleraktivitäten mehrere Skalierungsereignisse gleichzeitig auslösen. Vermeiden Sie Richtlinien, die sich gegenseitig beeinflussen. Sie könnten beispielsweise eine unendliche Schleife schaffen, wenn Sie Richtlinien für Skalierung und Reduzierung erstellen, die die Kapazität über den jeweiligen Schwellenwert hinaus festlegen.

Einstellen der maximalen und minimalen Kapazität

Jede Flotte hat eine Einstellung für das maximale und für das minimale Kapazitätslimit. Diese Funktion ist wichtig, wenn Sie Auto Scaling verwenden. Bei der automatischen Skalierung wird die Kapazität niemals auf einen Wert außerhalb dieses Bereichs festgelegt. Standardmäßig haben neu

erstellte Flotten einen Mindestwert von 0 und einen Höchstwert von 1. Erhöhen Sie den Maximalwert, damit sich Ihre Auto-Scaling-Richtlinie wie vorgesehen auf die Kapazität auswirkt.

Die Flottenkapazität wird auch durch Beschränkungen des Instance-Typs der Flotte und durch Servicekontingente in Ihrer Flotte eingeschränkt. AWS-Konto Außerhalb dieser Limits und Kontingente können Sie keine Mindest- und Höchstwerte festlegen.

Verfolgen von Metriken nach einer Kapazitätsänderung

Nachdem Amazon die Kapazität als Reaktion auf eine automatische Skalierungsrichtlinie geändert hat, GameLift wartet Amazon 10 Minuten, bevor es auf Auslöser derselben Richtlinie reagiert. Dieses Warten gibt Amazon GameLift Zeit, die neuen Instanzen hinzuzufügen, die Spielserver zu starten, Spieler zu verbinden und mit dem Sammeln von Daten von den neuen Instances zu beginnen. Während dieser Zeit GameLift bewertet Amazon die Richtlinie anhand der Metrik und verfolgt den Bewertungszeitraum der Richtlinie, der nach einem Skalierungsereignis erneut beginnt. Dies bedeutet, dass eine Skalierungsrichtlinie unmittelbar nach Ablauf der Wartezeit ein weiteres Skalierungsereignis auslösen könnte.

Es gibt keine Wartezeit zwischen den Skalierungsereignissen, in denen verschiedene Auto-Scaling-Richtlinien starten.

GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten

Eine Spielesitzungswarteschlange ist der Hauptmechanismus, um neue Spielsitzungsanfragen zu verarbeiten und verfügbare Spieleserver zu finden, auf denen sie gehostet werden können. Warteschlangen bieten Spieleentwicklern und Spielern erhebliche Vorteile. Dazu zählen:

- Warteschlangen sorgen für die bestmögliche Platzierung. Bei der Verarbeitung von Anfragen zur Platzierung von Spielsitzungen verwendet eine Warteschlange GameLift Amazon-Algorithmen, um Warteschlangenpositionen auf der Grundlage einer Reihe von definierten Einstellungen zu priorisieren.
- Hoste Spiele auf günstigeren Spot-Flotten. Verwenden Sie Warteschlangen, um die Nutzung von AWS Spot-Flotten zu optimieren, die deutlich niedrigere Hosting-Kosten bieten. Standardmäßig versuchen Warteschlangen immer, neue Spielsitzungen in Spot-Flotten zu platzieren.
- Platziere neue Spiele schneller, wenn die Nachfrage hoch ist. Warteschlangen verwenden mehrere mögliche Standorte für Platzierungen. Dies bedeutet, dass es immer Ersatzkapazität gibt, wenn der bevorzugte Platzierungsort nicht verfügbar ist.

- Machen Sie die Spielverfügbarkeit robuster. Ausfälle können passieren. Bei einer Warteschlange an mehreren Standorten muss eine Verlangsamung oder ein Ausfall den Zugriff der Spieler auf dein Spiel nicht beeinträchtigen.
- Verwenden Sie zusätzliche Flottenkapazität effizienter. Um einem unerwarteten Anstieg der Spielernachfrage gerecht zu werden, bieten Warteschlangen schnellen Zugriff auf zusätzliche Hosting-Kapazität. Die Flottenstandorte in einer Warteschlange stellen sich gegenseitig Reservekapazitäten zur Verfügung. Die Standorte werden je nach Nachfrage der Spieler nach oben oder unten skaliert.
- Erhalte Statistiken zu Platzierungen in Spielsitzungen und zur Leistung in der Warteschlange. Amazon gibt GameLift Queue-Metriken aus, darunter Statistiken über erfolgreiche und fehlgeschlagene Platzierungen, die Anzahl der Anfragen in der Warteschlange und die durchschnittliche Zeit, die Anfragen in der Warteschlange verbringen. Sie können diese Metriken in der GameLift Amazon-Konsole oder in einsehenCloudWatch.

Informationen zu den ersten Schritten mit Warteschlangen finden Sie unter. Entwerfen Sie eine Warteschlange für Spielsitzungen

Entwerfen Sie eine Warteschlange für Spielsitzungen

In diesem Thema wird beschrieben, wie eine Warteschlange entworfen wird, die ein Spielererlebnis mit minimaler Latenz bietet und Hosting-Ressourcen effizient nutzt. Weitere Informationen zu Warteschlangen für Spielesitzungen und ihrer Funktionsweise finden Sie unter. GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten

Für diese GameLift Amazon-Funktionen sind Warteschlangen erforderlich:

- Matchmaking mit FlexMatch
- Spot-Instances mit Amazon verwenden GameLift

Definieren Sie den Umfang Ihrer Warteschlange

Die Spielerpopulation Ihres Spiels besteht möglicherweise aus Spielergruppen, die nicht zusammen spielen sollten. Wenn du dein Spiel zum Beispiel in zwei Sprachen veröffentlichst, sollte jede Sprache ihre eigenen Spieleserver haben.

Um die Platzierung von Spielsitzungen für deine Spielerpopulation festzulegen, erstelle für jedes Spielersegment eine separate Warteschlange. Übersuche jede Warteschlange, um die Spieler

auf den richtigen Spielservern zu platzieren. Zu den gängigen Methoden zur Festlegung des Gültigkeitsbereichs von Warteschlangen gehören:

- Nach geografischen Standorten. Wenn Sie Ihre Spielserver in mehreren geografischen Gebieten einsetzen, können Sie Warteschlangen für Spieler an jedem Standort einrichten, um die Latenz der Spieler zu reduzieren.
- Durch Build- oder Skriptvariationen. Wenn du mehr als eine Variante deines Gameservers hast, unterstützt du möglicherweise Spielergruppen, die nicht an denselben Spielsitzungen teilnehmen können. Beispielsweise können Gameserver-Builds oder -Skripts verschiedene Sprachen oder Gerätetypen unterstützen.
- Nach Ereignistypen. Sie können eine spezielle Warteschlange einrichten, um Spiele für Teilnehmer an Turnieren oder anderen besonderen Veranstaltungen zu verwalten.

Erstellen Sie eine Latenzrichtlinie für Spieler

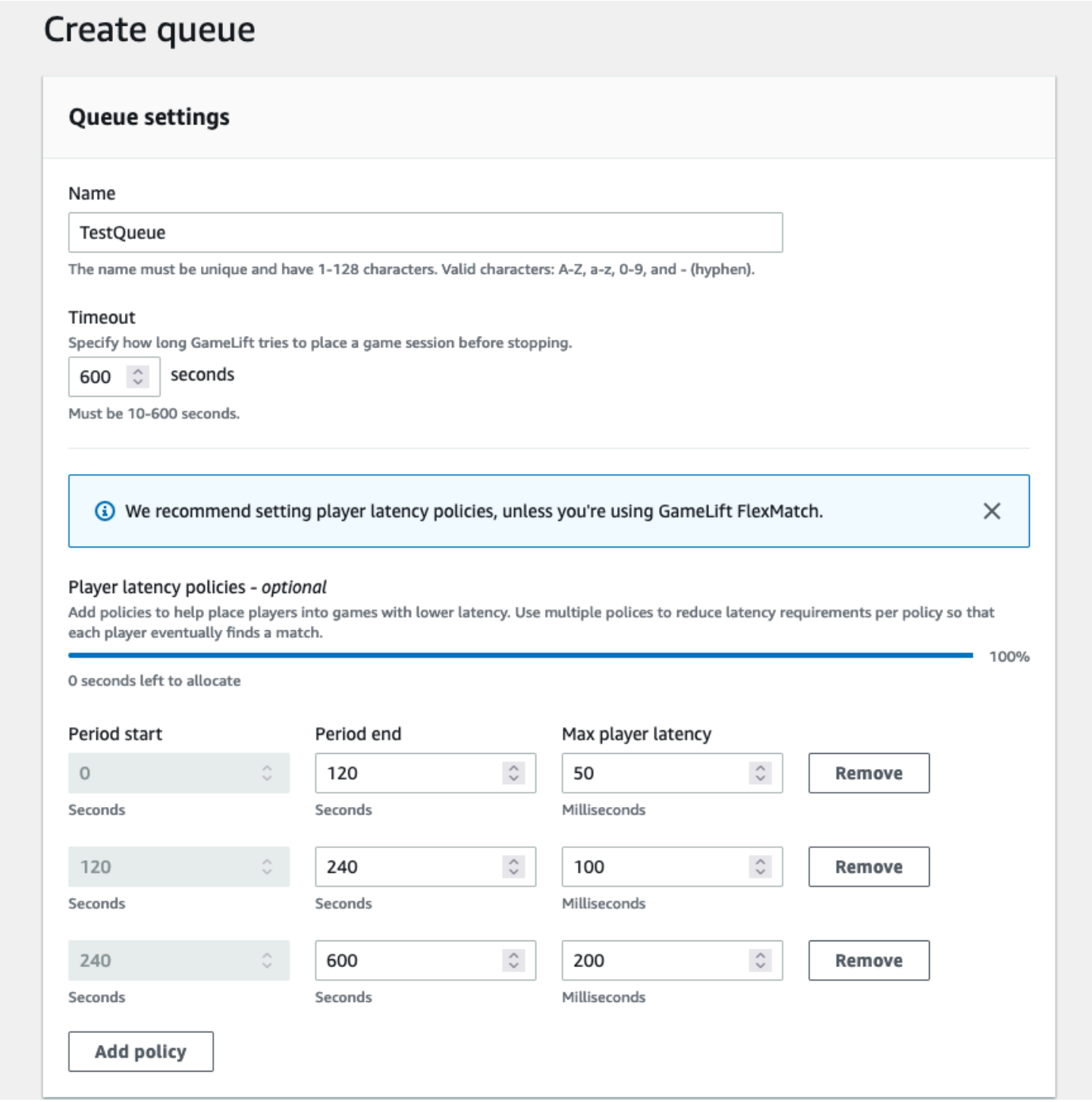
Wenn Ihre Platzierungsanfragen Daten zur Spielerlatenz enthalten, findet der Algorithmus Spielsitzungen an Orten mit der niedrigsten durchschnittlichen Latenz für alle Spieler. Das Platzieren GameLift von Spielsitzungen auf der Grundlage der durchschnittlichen Spielerlatenz verhindert, dass Amazon die meisten Spieler in Spiele mit hoher Latenz versetzt. Amazon platziert Spieler jedoch GameLift immer noch mit extremer Latenz. Um diesen Spielern gerecht zu werden, erstellen Sie Richtlinien für die Spielerlatenz.

Eine Richtlinie zur Spielerlatenz verhindert, dass Amazon GameLift eine angeforderte Spielsitzung an einer Stelle platziert, an der bei den angeforderten Spielern eine Latenz über dem Maximalwert auftreten würde. Richtlinien zur Spielerlatenz können Amazon GameLift auch daran hindern, Spielsitzungsanfragen mit Spielern mit höherer Latenz abzugleichen.

> **Tip**
>
> Um latenzspezifische Regeln zu verwalten, wie z. B. die Anforderung einer ähnlichen Latenz für alle Spieler in einer Gruppe, können Sie Amazon verwenden, um latenzbasierte Matchmaking-Regeln GameLift FlexMatch zu erstellen.

Stellen Sie sich zum Beispiel diese Warteschlange mit einem Timeout von 5 Minuten und den folgenden Richtlinien zur Spielerlatenz vor:

1. Verbringe 120 Sekunden damit, nach einem Ort zu suchen, an dem alle Spielerlatenzen unter 50 Millisekunden liegen.
2. Verbringe 120 Sekunden damit, nach einem Ort zu suchen, an dem alle Spielerlatenzen weniger als 100 Millisekunden betragen.
3. Verbringe die verbleibende Wartezeit bis zum Timeout damit, nach einem Ort zu suchen, an dem alle Spielerlatenzen unter 200 Millisekunden liegen.



Erstellen Sie eine Warteschlange mit mehreren Standorten

Wir empfehlen ein Design mit mehreren Standorten für alle Warteschlangen. Dieses Design kann die Platzierungsgeschwindigkeit und die Stabilität des Hostings verbessern. Ein Design mit mehreren Standorten ist erforderlich, um Spielerlatenzdaten zu verwenden, um Spieler mit minimaler Latenz in Spielsitzungen zu versetzen. Wenn Sie Warteschlangen mit mehreren Standorten einrichten, die

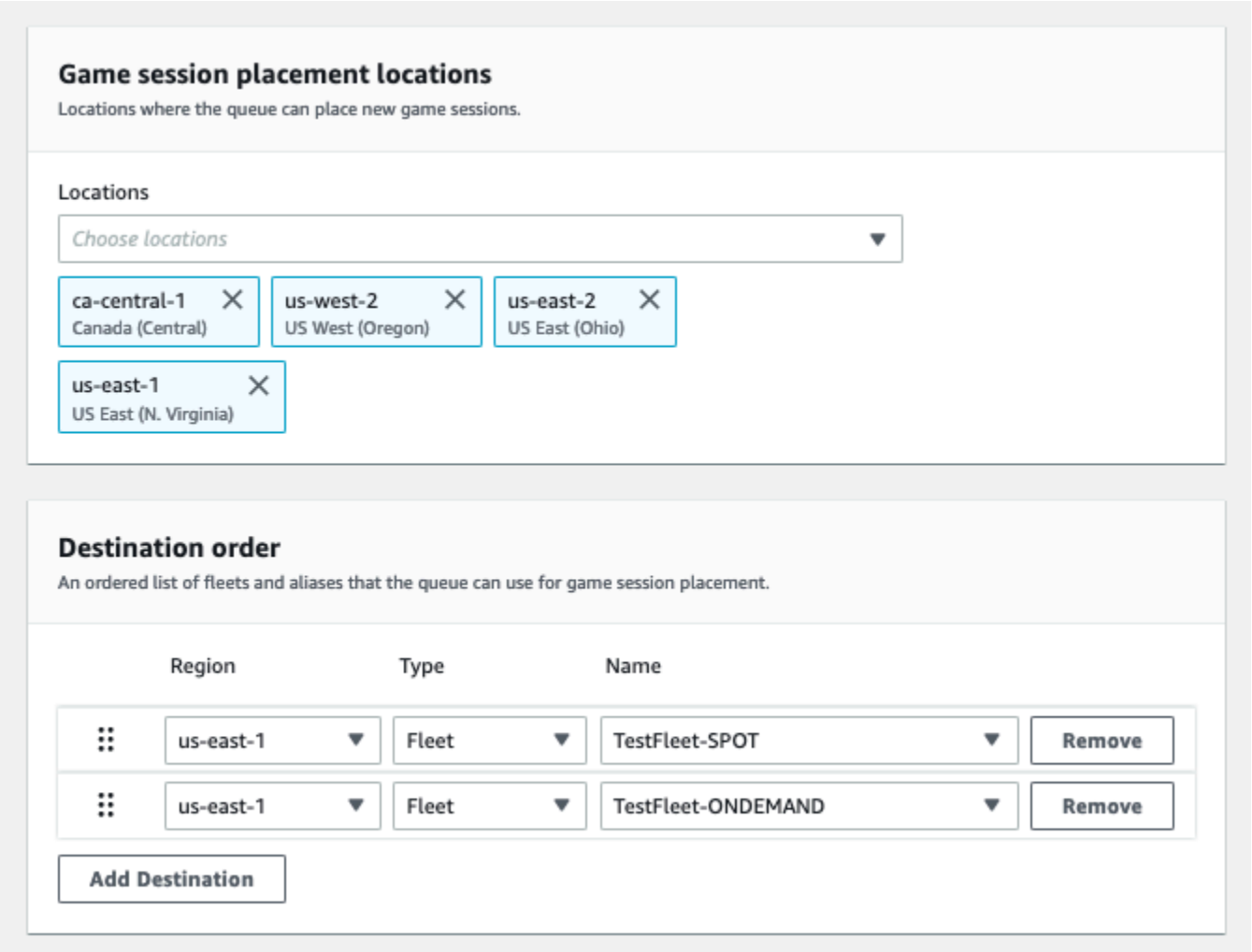
Spot-Instance-Flotten verwenden, folgen Sie den Anweisungen unter. Tutorial: Richten Sie eine Warteschlange für Spielsitzungen für Spot-Instances ein

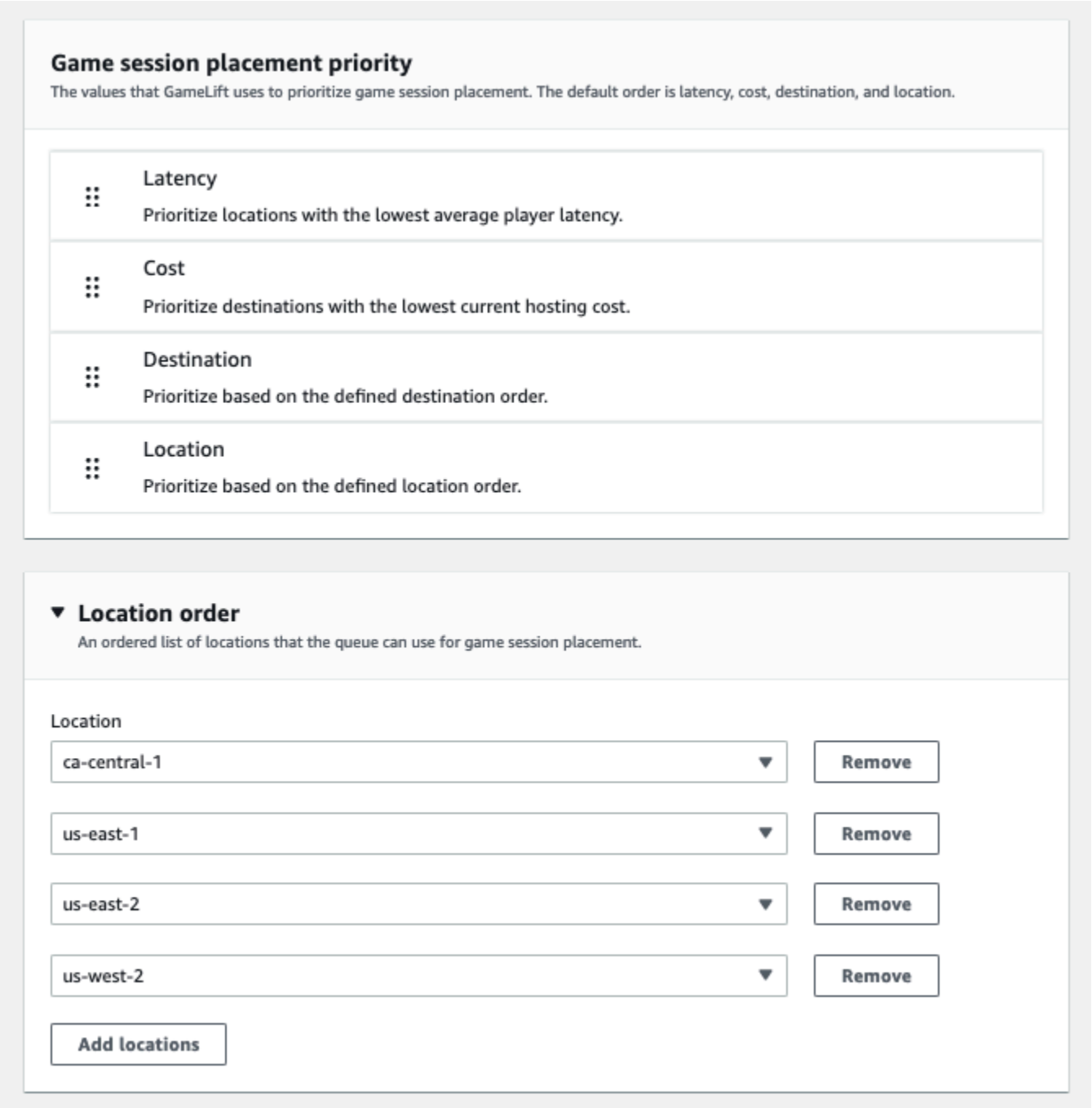
Eine Möglichkeit, eine Warteschlange mit mehreren Standorten zu erstellen, besteht darin, einer Warteschlange eine Flotte mit mehreren Standorten hinzuzufügen. Auf diese Weise kann die Warteschlange Spielsitzungen an jedem beliebigen Standort der Flotte veranstalten. Sie können aus Redundanzgründen auch andere Flotten mit unterschiedlichen Konfigurationen oder Heimatstandorten hinzufügen. Wenn Sie eine Spot-Instance-Flotte mit mehreren Standorten verwenden, befolgen Sie die bewährten Methoden und fügen Sie eine On-Demand-Instance-Flotte mit denselben Standorten hinzu.

Das folgende Beispiel beschreibt den Prozess des Entwurfs einer einfachen Warteschlange mit mehreren Standorten. In diesem Beispiel verwenden wir zwei Flotten: eine Spot-Instance-Flotte und eine On-Demand-Instance-Flotte. Jede Flotte hat die folgenden AWS-Regionen Platzierungsorte: `us-east-1`us-east-2,`ca-central-1`, und`us-west-2`.

Um eine einfache Warteschlange mit mehreren Standorten mit Flotten mit mehreren Standorten zu erstellen

1. Wählen Sie einen Ort, an dem die Warteschlange erstellt werden soll. Sie können die Anforderungslatenz minimieren, indem Sie die Warteschlange an einem Ort in der Nähe platzieren, an dem Sie den Client-Service bereitgestellt haben. In diesem Beispiel erstellen wir die Warteschlange in`us-east-1`.
2. Erstellen Sie eine neue Warteschlange und fügen Sie Ihre Flotten mit mehreren Standorten als Warteschlangenziele hinzu. Die Zielreihenfolge bestimmt, wie Amazon Spielsitzungen GameLift platziert. In diesem Beispiel listen wir zuerst die Spot-Instance-Flotte und dann die On-Demand-Instance-Flotte auf.
3. Definieren Sie die Prioritätsreihenfolge der Spielsitzungen in der Warteschlange. Diese Reihenfolge bestimmt, wo die Warteschlange zuerst nach einem verfügbaren Spielserver sucht. In diesem Beispiel verwenden wir die Standardprioritätsreihenfolge.
4. Definieren Sie die Reihenfolge der Standorte. Wenn Sie die Standortreihenfolge nicht definieren, GameLift verwendet Amazon die Standorte in alphabetischer Reihenfolge.


Game session placement locations
Locations where the queue can place new game sessions.
Locations
Choose locations
ca-central-1
Canada (Central)
us-west-2
US West (Oregon)
us-east-2
US East (Ohio)
us-east-1
US East (N. Virginia)
Destination order
An ordered list of fleets and aliases that the queue can use for game session placement.
Region
Type
Name
us-east-1
Fleet
TestFleet-SPOT
Remove
us-east-1
Fleet
TestFleet-ONDEMAND
Remove
Add Destination



Priorisieren Sie die Platzierung von Spielsitzungen

Amazon GameLift verwendet den FleetIQ-Algorithmus, um anhand einer Reihe von Kriterien zu bestimmen, wo eine neue Spielsitzung platziert werden soll. Sie können die Standardprioritätsreihenfolge verwenden oder die Reihenfolge anpassen.

Standardprioritätsreihenfolge

Bei Platzierungsanfragen, die Daten zur Spielerlatenz beinhalten, priorisiert FleetIQ die Platzierungskriterien für Spielsitzungen in der folgenden Standardreihenfolge:

1. Latenz — Niedrigste durchschnittliche Latenz für alle Spieler in der Anfrage.
2. Kosten — Niedrigste Hosting-Kosten, wenn die Latenz an mehreren Standorten gleich ist. Die Hosting-Kosten basieren in erster Linie auf einer Kombination aus Instance-Typ und Standort.
3. Ziel — Zielreihenfolge, wenn Latenz und Kosten an mehreren Standorten gleich sind. FleetIQ priorisiert Ziele in der Reihenfolge, die in der Warteschlangenkonfiguration aufgeführt ist.
4. Standort — Standortreihenfolge, wenn Latenz, Kosten und Ziel an mehreren Standorten gleich sind. FleetIQ priorisiert Standorte auf der Grundlage der Reihenfolge, die in der Warteschlangenkonfiguration aufgeführt ist.

Benutzerdefinierte Prioritätsreihenfolge

Um die Prioritätsreihenfolge einer Warteschlange in der GameLiftAmazon-Konsole anzupassen, ziehen Sie den Prioritätswert an die gewünschte Position. Um die Prioritätsreihenfolge einer Warteschlange mithilfe von AWS Command Line Interface (AWS CLI) anzupassen, verwenden Sie den create-game-session-queueBefehl mit der --priority-configuration Option. Sie können diesen Befehl verwenden, um eine neue Warteschlange zu erstellen oder eine bestehende Warteschlange zu aktualisieren.

Der FleetIQ-Algorithmus fügt alle Kriterien, die nicht ausdrücklich erwähnt wurden, an das Ende Ihrer Liste an, basierend auf der Standardreihenfolge. Wenn Sie das Standortkriterium in Ihre Prioritätskonfiguration aufnehmen, müssen Sie auch eine geordnete Liste der Standorte angeben.

Entwerfen Sie nach Bedarf mehrere Warteschlangen

Abhängig von deinem Spiel und deinen Spielern möchtest du vielleicht mehr als eine Warteschlange für Spielsitzungen erstellen. Wenn dein Game-Client-Dienst eine neue Spielsitzung anfordert, gibt er an, welche Spielesitzungswarteschlange verwendet werden soll. Beachten Sie Folgendes, um zu entscheiden, ob Sie mehrere Warteschlangen verwenden sollten:

- Variationen deines Gameservers. Du kannst für jede Variante deines Gameservers eine separate Warteschlange erstellen. Alle Flotten in einer Warteschlange müssen kompatible Spielserver bereitstellen. Dies liegt daran, dass Spieler, die die Warteschlange verwenden, um an Spielen teilzunehmen, auf einem der Spieleserver der Warteschlange spielen können müssen.
- Verschiedene Spielergruppen. Sie können je nach Spielergruppe anpassen, wie Amazon Spielsitzungen GameLift platziert. Beispielsweise benötigen Sie möglicherweise Warteschlangen,

die für bestimmte Spielmodi angepasst sind und für die ein besonderer Instanztyp oder eine spezielle Laufzeitkonfiguration erforderlich ist. Oder vielleicht möchtest du eine spezielle Warteschlange einrichten, um die Platzierungen für ein Turnier oder ein anderes Event zu verwalten.

- Metriken für Warteschlangen bei Spielsitzungen. Du kannst Warteschlangen einrichten, je nachdem, wie du die Platzierungsdaten für Spielsitzungen sammeln möchtest. Weitere Informationen finden Sie unter GameLiftAmazon-Metriken für Warteschlangen.

Bewerten Sie Warteschlangenmetriken

Verwenden Sie Metriken, um zu bewerten, wie gut Ihre Warteschlangen ausgeführt werden. Sie können Kennzahlen zu Warteschlangen in der GameLiftAmazon-Konsole oder in Amazon einsehen. CloudWatch Eine Liste und Beschreibungen der Warteschlangenmetriken finden Sie unterGameLiftAmazon-Metriken für Warteschlangen.

Warteschlangenmetriken können Aufschluss über Folgendes geben:

- Gesamtleistung der Warteschlange — Die Warteschlangenmetriken geben an, wie erfolgreich eine Warteschlange auf Platzierungsanfragen reagiert. Diese Kennzahlen können Ihnen auch dabei helfen, herauszufinden, wann und warum Platzierungen fehlschlagen. Bei Warteschlangen mit manuell skalierten Flotten können die `QueueDepth` Kennzahlen `AverageWaitTime` und angeben, wann Sie die Kapazität einer Warteschlange anpassen sollten.
- Leistung des FleetIQ-Algorithmus — Bei Platzierungsanfragen, die den FleetIQ-Algorithmus verwenden, zeigen die Metriken, wie oft der Algorithmus die ideale Platzierung der Spielsitzung findet. Bei der Platzierung kann der Einsatz von Ressourcen mit der niedrigsten Spielerlatenz oder Ressourcen mit den niedrigsten Kosten priorisiert werden. Es gibt auch Fehlerkennzahlen, die häufige Gründe aufzeigen, warum Amazon keine ideale Platzierung finden GameLift kann. Weitere Informationen zu den Metriken finden Sie unter Überwachen Sie Amazon GameLift mit Amazon CloudWatch.
- Standortspezifische Platzierungen — Bei Warteschlangen mit mehreren Standorten zeigen Kennzahlen erfolgreiche Platzierungen nach Standort. Für Warteschlangen, die den FleetIQ-Algorithmus verwenden, bieten diese Daten nützliche Informationen darüber, wo Spieleraktivitäten stattfinden.

Beachten Sie bei der Bewertung der Metriken für die Leistung des FleetIQ-Algorithmus die folgenden Tipps:

- Um zu verfolgen, wie schnell die Warteschlange eine ideale Platzierung findet, verwenden Sie die `PlacementsSucceeded` Metrik in Kombination mit den FleetIQ-Metriken für die niedrigste Latenz und den niedrigsten Preis.
- Um die Wahrscheinlichkeit zu erhöhen, dass eine Warteschlange eine ideale Platzierung findet, sollten Sie sich die folgenden Fehlerkennzahlen ansehen:
  - Wenn der hoch `FirstChoiceOutOfCapacity` ist, passen Sie die Kapazitätsskalierung für die Flotten der Warteschlange an.
  - Wenn die `FirstChoiceNotViable` Fehlermetrik hoch ist, schauen Sie sich Ihre Spot-Instance-Flotten an. Spot-Instance-Flotten gelten als nicht rentabel, wenn die Unterbrechungsrate für einen bestimmten Instance-Typ zu hoch ist. Um dieses Problem zu beheben, ändern Sie die Warteschlange, sodass Spot-Instance-Flotten mit unterschiedlichen Instance-Typen verwendet werden. Wir empfehlen, dass Sie Spot-Instance-Flotten mit unterschiedlichen Instance-Typen an jedem Standort einbeziehen.

Bewährte Methoden für Warteschlangen für GameLift Amazon-Spielesitzungen

Im Folgenden finden Sie einige bewährte Methoden, die Ihnen helfen können, effektive Warteschlangen für die Platzierung von Spielsitzungen zu erstellen.

Bewährte Methoden für Warteschlangen aller Flottentypen

Eine Warteschlange enthält eine Liste von Flottenzielen, an denen neue Spielsitzungen platziert werden können. Jede Flotte kann über Instanzen verfügen, die an mehreren geografischen Standorten bereitgestellt werden. Bei der Auswahl einer Platzierung wählt die Warteschlange eine Kombination aus einer Flotte und einem Flottenstandort aus. Sie geben eine Reihe von Prioritäten für die Warteschlange an, die bei der Auswahl einer Platzierung verwendet werden sollen.

Beachten Sie die folgenden Richtlinien und bewährten Verfahren:

- Füge Flotten an Orten hinzu, die deine Spieler abdecken. Sie können Flotten und Aliase an jedem verfügbaren Standort hinzufügen. Der Standort ist wichtig, wenn du Platzierungen anhand der gemeldeten Spielerlatenz vornimmst.
- Verwenden Sie Aliase für alle Flotten. Weisen Sie jeder Flotte in einer Warteschlange einen Alias zu und verwenden Sie die Aliasnamen, wenn Sie Ziele in Ihrer Warteschlange festlegen.

- Verwenden Sie für alle Flotten den gleichen oder einen ähnlichen Spielbuild oder ein ähnliches Skript. In der Warteschlange können Spieler mit einer beliebigen Flotte in der Warteschlange an Spielsitzungen teilnehmen. Spieler müssen in der Lage sein, an jeder Spielsitzung auf jeder Flotte teilzunehmen.
- Erstelle Flotten an mindestens zwei Standorten. Indem du Spielserver an mindestens einem anderen Standort hostest, milderst du die Auswirkungen regionaler Ausfälle auf deine Spieler. Sie können Ihre Backup-Flotten verkleinern und die automatische Skalierung verwenden, um die Kapazität zu erhöhen, falls die Nutzung steigt.
- Priorisiere die Platzierung deiner Spielsitzung. Eine Warteschlange priorisiert Platzierungsentscheidungen auf der Grundlage verschiedener Elemente, einschließlich der Reihenfolge der Zielliste.
- Erstellen Sie Ihre Warteschlange am selben Ort wie Ihr Kundenservice. Indem Sie Ihre Warteschlange an einem Ort in der Nähe Ihres Kundendienstes platzieren, können Sie die Kommunikationslatenz minimieren.
- Verwenden Sie Flotten mit mehreren Standorten. Verwenden Sie die Konfiguration des Warteschlangenfilters, um zu verhindern, dass die Warteschlange Spielsitzungen an bestimmten Orten platziert. Du kannst mindestens zwei Flotten an mehreren Standorten mit unterschiedlichen Heimatstandorten einsetzen, um die Auswirkungen von Spielplatzierungen während eines regionalen Ausfalls zu mildern.
- Verwenden Sie für alle Flotten dieselbe TLS-Zertifikatseinstellung. Spielclients, die sich mit Spielsitzungen in Ihren Flotten verbinden, müssen über kompatible Kommunikationsprotokolle verfügen.

Bewährte Methoden für Warteschlangen mit Spot-Flotten

Wenn Ihre Warteschlange Spot-Flotten umfasst, richten Sie eine stabile Warteschlange ein. Dadurch werden die Kosteneinsparungen mit Spot-Flotten genutzt und gleichzeitig die Auswirkungen von Spielsitzungsunterbrechungen minimiert. Hilfe beim korrekten Aufbau von Flotten und Warteschlangen für Spielsitzungen zur Verwendung mit Spot-Flotten findest du unter. Tutorial: Richten Sie eine Warteschlange für Spielsitzungen für Spot-Instances ein Weitere Informationen zu Spot-Instances finden Sie unterSpot-Instances mit Amazon verwenden GameLift.

Zusätzlich zu den allgemeinen bewährten Verfahren im vorherigen Abschnitt sollten Sie die folgenden SPOT-spezifischen bewährten Verfahren berücksichtigen:

- Erstellen Sie an jedem Standort mindestens eine On-Demand-Flotte. On-Demand-Flotten stellen Backup-Spieleserver für Ihre Spieler bereit. Sie können Ihre Backup-Flotten verkleinern, bis sie benötigt werden, und die automatische Skalierung verwenden, um die On-Demand-Kapazität zu erhöhen, wenn Spot-Flotten nicht verfügbar sind.
- Wählen Sie verschiedene Instance-Typen für mehrere Spot-Flotten an einem Standort aus. Wenn ein Spot-Instance-Typ vorübergehend nicht verfügbar ist, betrifft die Unterbrechung nur eine Spot-Flotte am Standort. Es empfiehlt sich, allgemein verfügbare Instance-Typen auszuwählen und Instance-Typen derselben Familie zu verwenden (z. B. m5.large, m5.xlarge, m5.2xlarge). Verwenden Sie die GameLiftAmazon-Konsole, um historische Preisdaten für Instance-Typen einzusehen.

Erstellen Sie eine Warteschlange für Spielsitzungen

Warteschlangen werden verwendet, um neue Spielsitzungen mit den besten verfügbaren Hosting-Ressourcen über mehrere Flotten und Regionen hinweg zu platzieren. Weitere Informationen zum Erstellen von Warteschlangen für Ihr Spiel finden Sie unter Entwerfen Sie eine Warteschlange für Spielsitzungen.

In einer Spiele-Client werden Spielsitzungen mit Warteschlangen unter Verwendung von Platzierungsanforderungen gestartet. Weitere Informationen zur Platzierung von Spielsitzungen findest du unterSpielsitzungen erstellen.

Beim Aktualisieren des Warteschlangenziels in einer Warteschlange gibt es eine kurze Übergangszeit (bis zu 30 Sekunden), in der Spielsitzungen, die an den Warteschlangenzielen platziert wurden, möglicherweise immer noch auf der alten Flotte landen.

Console

1. Wählen Sie in der GameLiftAmazon-Konsole auf der Navigationsseite Warteschlangen aus.
2. Wählen Sie auf der Seite Queues (Warteschlangen) Create queue (Neue Warteschlange erstellen) aus.
3. Gehen Sie auf der Seite Warteschlange erstellen unter Warteschlangeneinstellungen wie folgt vor:

    a. Geben Sie unter Name einen Namen für die Warteschlange ein.

b. Geben Sie für Timeout die Dauer ein, nach der Amazon versuchen GameLift soll, eine Spielsitzung zu starten, bevor es beendet wird. Amazon GameLift sucht in jeder Flotte nach verfügbaren Ressourcen, bis die Anfrage abläuft.

c. (Optional) Geben Sie für die Richtlinien zur Spielerlatenz ein, wie lange Amazon innerhalb der definierten maximalen Latenz nach Ressourcen suchen GameLift soll. Fügen Sie zusätzliche Richtlinien hinzu, um die maximale Latenz schrittweise zu verringern. Um weitere Richtlinien hinzuzufügen, wählen Sie Richtlinie hinzufügen.

4. Wählen Sie unter Platzierungsorte für Spielsitzungen die Orte aus, die in die Warteschlange aufgenommen werden sollen. Standardmäßig sind Alle Standorte enthalten. Alle Flotten in der Warteschlange müssen dieselbe Zertifikatskonfiguration haben. Auf allen Flotten sollten Spielversionen laufen, die mit den Spielclients kompatibel sind, die die Warteschlange verwenden.

5. Fügen Sie unter Zielreihenfolge ein oder mehrere Ziele zur Warteschlange hinzu.

   a. Wählen Sie Add destination (Ziel hinzufügen).

   b. Wählen Sie den Standort aus, an dem sich das Ziel befindet.

   c. Wählen Sie den Typ für Ihr Ziel aus.

   d. Wählen Sie aus der resultierenden Liste von Flotten- oder Aliasnamen die Flotte (Alias) aus, die Sie hinzufügen möchten.

   e. Wenn Sie mehrere Ziele haben, legen Sie die Standardreihenfolge fest, indem Sie das Symbol mit den sechs Punkten links neben dem Ziel ziehen. Amazon GameLift verwendet diese Reihenfolge bei der Suche nach Zielen nach verfügbaren Ressourcen, um eine neue Spielsitzung zu platzieren.

6. Fügen Sie für die Priorität der Platzierung von Spielsitzungen die Werte für Latenz, Kosten, Ziel und Standort hinzu und ziehen Sie sie an die gewünschte Stelle, um zu definieren, wie Amazon Flotten in GameLift Ihrer Warteschlange priorisiert. Weitere Informationen zur Priorisierung von Flotten finden Sie unter. Priorisieren Sie die Platzierung von Spielsitzungen

7. Fügen Sie Ihrer Standortreihenfolge Standorte hinzu und ziehen Sie sie auf die Priorität, die die Warteschlange verwenden soll. Wenn der Standort die letzte Priorität für die Platzierung der Spielsitzung ist, GameLift verwendet Amazon ihn als Tiebreaker.

8. (Optional) Gehen Sie unter Einstellungen für Ereignisbenachrichtigungen wie folgt vor:

   a. Wählen oder erstellen Sie ein SNS-Thema, um Benachrichtigungen zu platzierungsbezogenen Ereignissen zu erhalten. Weitere Informationen zu

Ereignisbenachrichtigungen finden Sie unterRichten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein.

b. Fügen Sie benutzerdefinierte Ereignisdaten hinzu, um sie an Ereignisse anzuhängen, die von dieser Warteschlange erstellt wurden.

9. (Optional) Fügen Sie Schlagworte hinzu. Weitere Informationen zum Taggen finden Sie unter Ressourcen zum Taggen AWS.
10. Wählen Sie Erstellen aus.

AWS CLI

Example Erstellen einer Warteschlange

Im folgenden Beispiel wird eine Warteschlange für Spielsitzungen mit diesen Konfigurationen erstellt:

- Eine fünfminütige Auszeit
- Zwei Flottenziele
- Filtert, um nur Standorte im`us-east-1`, zuzulassen`us-east-2`. `us-west-2`, und `ca-central-1`
- Priorisiert Ziele auf der Grundlage der Kosten und dann Standorte in der definierten Reihenfolge.

```
aws gamelift create-game-session-queue \
    --name "sample-test-queue" \
    --timeout-in-seconds 300 \
    --destinations DestinationArn="arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-772266ba-8c82-4a6e-b620-a74a62a93ff8" DestinationArn="arn:aws:gamelift:us-
east-1:111122223333:fleet/fleet-33f28fb6-aa8b-4867-85b4-ceb217bf5994" \
    --filter-configuration "AllowedLocations=us-east-1, ca-central-1, us-east-2, us-
west-2" \
    --priority-configuration
 PriorityOrder="LOCATION","DESTINATION",LocationOrder="us-east-1","us-east-2","ca-
central-1","us-west-2" \
    --notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

> **Note**
>
> Sie können die ARN-Werte für Flotte und Alias abrufen, indem Sie entweder describe-fleet-attributesoder describe-alias mit der Flotte- oder Alias-ID aufrufen.

Wenn die `create-game-session-queue` Anfrage erfolgreich ist, GameLift gibt Amazon ein GameSessionQueueObjekt mit der neuen Warteschlangenkonfiguration zurück. Sie können jetzt Anfragen an die Warteschlange senden, indem Sie StartGameSessionPlacement.

Example Erstellen Sie eine Warteschlange mit Richtlinien zur Spielerlatenz

Im folgenden Beispiel wird eine Warteschlange für Spielsitzungen mit diesen Konfigurationen erstellt:

- Eine zehnminütige Auszeit
- Drei Flottenziele
- Eine Reihe von Latenzrichtlinien für Spieler

```
aws gamelift create-game-session-queue \
    --name "matchmaker-queue" \
    --timeout-in-seconds 600 \
    --destinations DestinationArn=arn:aws:gamelift:us-east-1::alias/alias-a1234567-
b8c9-0d1e-2fa3-b45c6d7e8910 \
                DestinationArn=arn:aws:gamelift:us-west-2::alias/alias-b0234567-
c8d9-0e1f-2ab3-c45d6e7f8901 \
                DestinationArn=arn:aws:gamelift:us-west-2::fleet/fleet-f1234567-
b8c9-0d1e-2fa3-b45c6d7e8912 \
    --player-latency-policies
 "MaximumIndividualPlayerLatencyMilliseconds=50,PolicyDurationSeconds=120" \

 "MaximumIndividualPlayerLatencyMilliseconds=100,PolicyDurationSeconds=120" \
                "MaximumIndividualPlayerLatencyMilliseconds=150" \
```

Wenn die `create-game-session-queue` Anfrage erfolgreich ist, GameLift gibt Amazon ein GameSessionQueueObjekt mit der neuen Warteschlangenkonfiguration zurück.

Richten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein

Mithilfe von Veranstaltungsbenachrichtigungen können Sie den Status einzelner Platzierungsanfragen überwachen. Wir empfehlen, für alle Spiele mit hoher Platzierungsaktivität Event-Benachrichtigungen einzurichten.

Es gibt zwei Möglichkeiten, Ereignisbenachrichtigungen einzurichten.

- Lassen Sie Amazon mithilfe einer Warteschlange Eventbenachrichtigungen zu einem Amazon Simple Notification Service (Amazon SNS) -Thema GameLift veröffentlichen.
- Verwenden Sie automatisch veröffentlichte EventBridge Amazon-Events und die zugehörigen Tools zur Verwaltung von Veranstaltungen.

Eine Liste der von Amazon GameLift ausgestrahlten Platzierungsereignisse für Spielsitzungen finden Sie unterPlatzierungsveranstaltungen für Spielsitzungen.

Ein SNS-Thema einrichten

Damit Amazon GameLift alle von einer Spielesitzungswarteschlange generierten Ereignisse zu einem Thema veröffentlichen kann, legen Sie das Benachrichtigungszielfeld auf ein Thema fest.

So richten Sie ein SNS-Thema für die GameLift Amazon-Eventbenachrichtigung ein

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-SNS-Konsole unter https://console.aws.amazon.com/sns/v3/home.
2. Wählen Sie auf der Seite mit den SNS-Themen die Option Thema erstellen aus und folgen Sie den Anweisungen, um Ihr Thema zu erstellen.
3. Gehen Sie unter Zugriffsrichtlinie wie folgt vor:

    a. Wählen Sie die Methode „Erweitert“.

    b. Fügen Sie der vorhandenen Richtlinie den folgenden fett gedruckten Abschnitt des JSON-Objekts hinzu.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
```

```
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
        }
      }
    }
  ]
}
```

c. (Optional) Fügen Sie dem Thema zusätzliche Zugriffskontrolle hinzu, indem Sie der Ressourcenrichtlinie Bedingungen hinzufügen.

4. Wählen Sie Create topic (Thema erstellen) aus.

5. Nachdem Sie Ihr SNS-Thema erstellt haben, fügen Sie es bei der Erstellung der Warteschlange zu Warteschlangen hinzu oder bearbeiten Sie eine bestehende Warteschlange, um es hinzuzufügen.

Richten Sie ein SNS-Thema mit serverseitiger Verschlüsselung ein

Mit serverseitiger Verschlüsselung (SSE) können Sie vertrauliche Daten in verschlüsselten Themen speichern. SSE schützt den Inhalt von Nachrichten in Amazon SNS-Themen mithilfe von Schlüsseln, die in AWS Key Management Service (AWS KMS) verwaltet werden. Weitere Informationen zur serverseitigen Verschlüsselung mit Amazon SNS finden Sie unter Encryption at Rest im Amazon Simple Notification Service Developer Guide.

Lesen Sie die folgenden Themen, um ein SNS-Thema mit serverseitiger Verschlüsselung einzurichten:

- Schlüssel im AWS Key Management ServiceEntwicklerhandbuch erstellen
- SSE für ein Thema im Amazon Simple Notification Service Developer Guide aktivieren

Verwenden Sie beim Erstellen Ihres KMS-Schlüssels die folgende KMS-Schlüsselrichtlinie:

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "gamelift.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:your_account:your_sns_topic_name"
        }
    }
```

```
}
```

Einrichten von EventBridge

Amazon veröffentlicht GameLift automatisch alle Platzierungsereignisse für SpielsitzungenEventBridge. Mit können EventBridge Sie Regeln einrichten, damit Ereignisse zur Verarbeitung an Ziele weitergeleitet werden. Sie können beispielsweise eine Regel festlegen, um das Ereignis `PlacementFulfilled` an eine AWS Lambda Funktion weiterzuleiten, die Aufgaben bearbeitet, die vor dem Herstellen einer Verbindung zu einer Spielsitzung anfallen. Weitere Informationen zu EventBridge finden Sie unter Was ist AmazonEventBridge? im EventBridgeAmazon-Benutzerhandbuch.

Im Folgenden finden Sie einige Beispiele für EventBridge Regeln zur Verwendung mit GameLift Amazon-Warteschlangen:

Stimmt mit Ereignissen aus allen GameLift Amazon-Warteschlangen überein

```
{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ]
}
```

Entspricht Ereignissen aus einer bestimmten Warteschlange

```
{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ],
    "resources": [
        "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
    ]
}
```

Tutorial: Richten Sie eine Warteschlange für Spielsitzungen für Spot-Instances ein

Einführung

In diesem Tutorial wird beschrieben, wie Sie die Platzierung von Spielsitzungen für Spiele einrichten, die auf kostengünstigen Spot-Flotten eingesetzt werden. Spot-Flotten erfordern zusätzliche Schritte, um die kontinuierliche Verfügbarkeit des Spielservers für Ihre Spieler aufrechtzuerhalten.

Zielgruppe

Dieses Tutorial richtet sich an Spieleentwickler, die Spot-Flotten verwenden möchten, um benutzerdefinierte Spieleserver oder Echtzeitserver zu hosten.

Was du lernen wirst

- Definieren Sie die Gruppe von Spielern, die Ihre Spielesitzungswarteschlange bedient.
- Baue eine Flotteninfrastruktur auf, die den Umfang der Warteschlange für Spielsitzungen unterstützt.
- Weisen Sie jeder Flotte einen Alias zu, um die Flotten-ID zu abstrahieren.
- Erstellen Sie eine Warteschlange, fügen Sie Flotten hinzu und priorisieren Sie, wo Amazon GameLift Spielsitzungen platziert.
- Füge Latenzrichtlinien für Spieler hinzu, um Latenzprobleme zu minimieren.

Voraussetzungen

Bevor Sie Flotten und Warteschlangen für die Platzierung von Spielsitzungen erstellen, müssen Sie die folgenden Aufgaben erledigen:

- Sehen Sie sich So GameLift funktioniert Amazon an.
- Integrieren Sie Ihren Gameserver mit Amazon GameLift.
- Laden Sie Ihren Gameserver-Build oder Ihr Echtzeit-Skript auf Amazon GameLift hoch.
- Planen Sie Ihre Flottenkonfiguration.

Schritt 1: Definieren Sie den Umfang Ihrer Warteschlange

In diesem Tutorial entwerfen wir eine Warteschlange für ein Spiel, das eine Gameserver-Build-Variante hat. Zum Start veröffentlichen wir das Spiel an zwei Orten: Asien-Pazifik (Seoul) und Asien-

Pazifik (Singapur). Da diese Standorte nahe beieinander liegen, ist Latenz für unsere Spieler kein Problem.

In diesem Beispiel gibt es ein Spielersegment, was bedeutet, dass wir eine Warteschlange erstellen. In Zukunft, wenn wir das Spiel in Nordamerika veröffentlichen, können wir eine zweite Warteschlange einrichten, die ausschließlich nordamerikanischen Spielern vorbehalten ist.

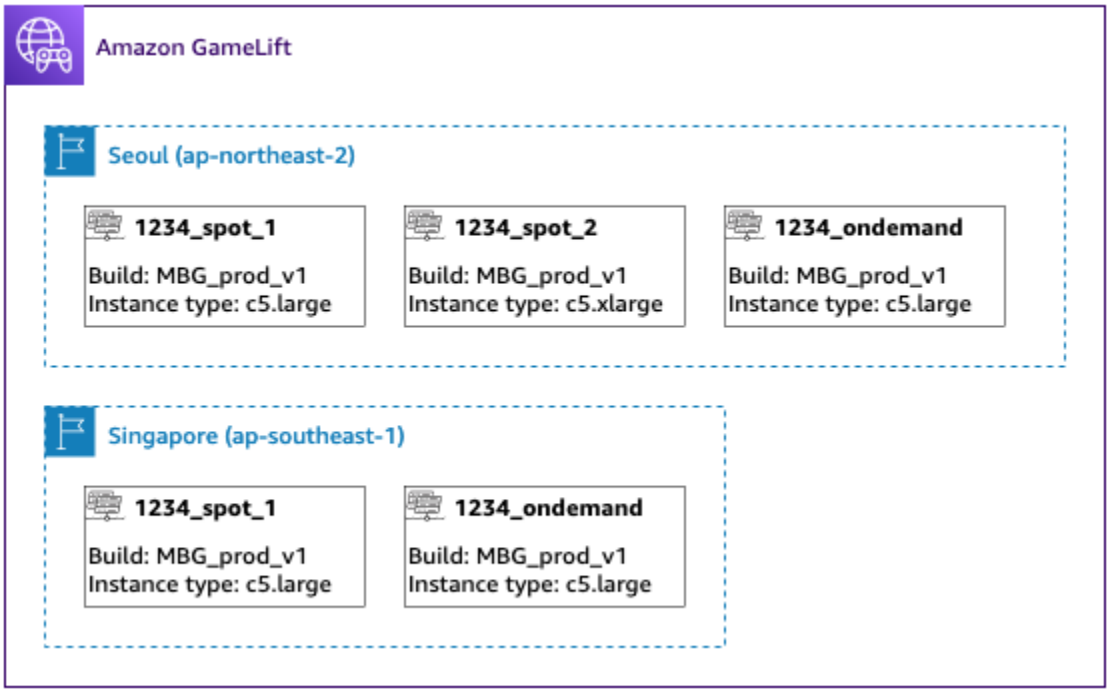
Weitere Informationen finden Sie unter Definieren Sie den Umfang Ihrer Warteschlange.

Schritt 2: Spot-Flotteninfrastruktur erstellen

Erstelle Flotten an Orten und mit Gameserver-Builds oder -Skripten, die dem von dir definierten Umfang entsprechen. Schritt 1: Definieren Sie den Umfang Ihrer Warteschlange

In diesem Tutorial erstellen wir eine Infrastruktur mit zwei Standorten mit mindestens einer Spot-Flotte und einer On-Demand-Flotte an jedem Standort. Jede Flotte verwendet den gleichen Gameserver-Build. Darüber hinaus gehen wir davon aus, dass der Spielerverkehr am Standort Seoul stärker sein wird, weshalb wir dort mehr Spot-Flotten hinzufügen werden.

Das folgende Diagramm zeigt das Beispiel einer Spot-Flotteninfrastruktur mit 3 Flotten am Standort ap-Northeast-2 (Seoul) und 2 Flotten am Standort ap-Southeast-1 (Singapur). Alle Instanzen in beiden Flotten verwenden den Build MBG_Prod_v1. Die Flotte in ap-northeast-2 enthält die folgenden Flottenkonfigurationen: fleet 1234_spot_1 mit dem Instance-Typ c5.large, fleet 1234_spot_2 mit dem Instance-Typ c5.xlarge und fleet 1234_ondemand mit dem Instance-Typ c5.large. Die Flotte in ap-southeast-1 enthält die folgenden Flottenkonfigurationen: fleet 1234_spot_1 mit dem Instance-Typ c5.large und fleet 1234_ondemand mit dem Instance-Typ c5.large.

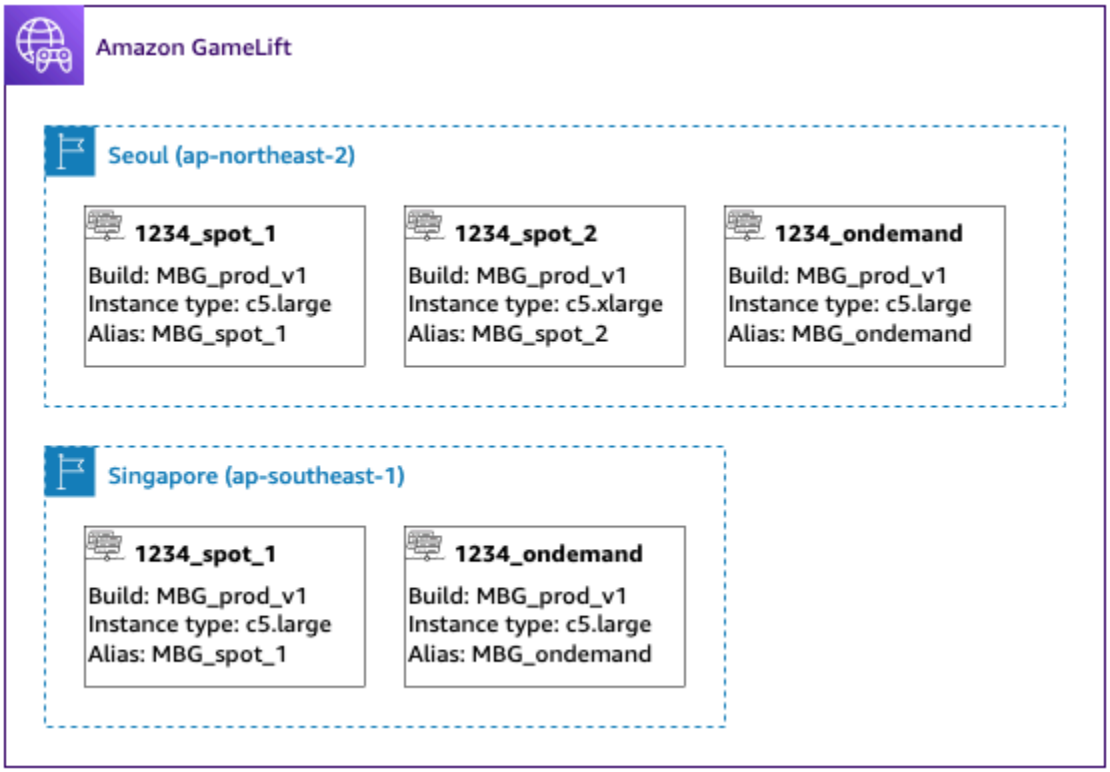


Schritt 3: Weisen Sie jeder Flotte Aliase zu

Erstellen Sie einen neuen Alias für jede Flotte in Ihrer Infrastruktur. Aliase abstrakten Flottenidentitäten, sodass ein regelmäßiger Flottenaustausch effizient ist. Weitere Hinweise zum Erstellen von Aliasen finden Sie unterEiner GameLift Amazon-Flotte einen Alias hinzufügen.

Unsere Flotteninfrastruktur hat fünf Flotten, daher erstellen wir mithilfe der Routing-Strategie fünf Aliase. Wir benötigen drei Aliase am Standort Asien-Pazifik (Seoul) und zwei Aliase am Standort Asien-Pazifik (Singapur).

Das folgende Diagramm zeigt die in Schritt zwei beschriebene Spot-Flotteninfrastruktur mit Aliasnamen, die jeder Flotte hinzugefügt wurden. Fleet 1234_spot_1 hat den Alias mbg_Spot_1, Fleet 1234_spot_2 hat den Alias mbg_Spot_2 und fleet 1234_ondemand hat den Alias mbg_OnDemand.



Weitere Informationen finden Sie unter Erstellen Sie eine Warteschlange mit mehreren Standorten.

Schritt 4: Erstellen Sie eine Warteschlange mit Zielen

Erstelle die Warteschlange für Spielsitzungen und füge deine Flottenziele hinzu. Weitere Hinweise zum Erstellen einer Warteschlange finden Sie unterErstellen Sie eine Warteschlange für Spielsitzungen.

Wenn du deine Warteschlange erstellst:

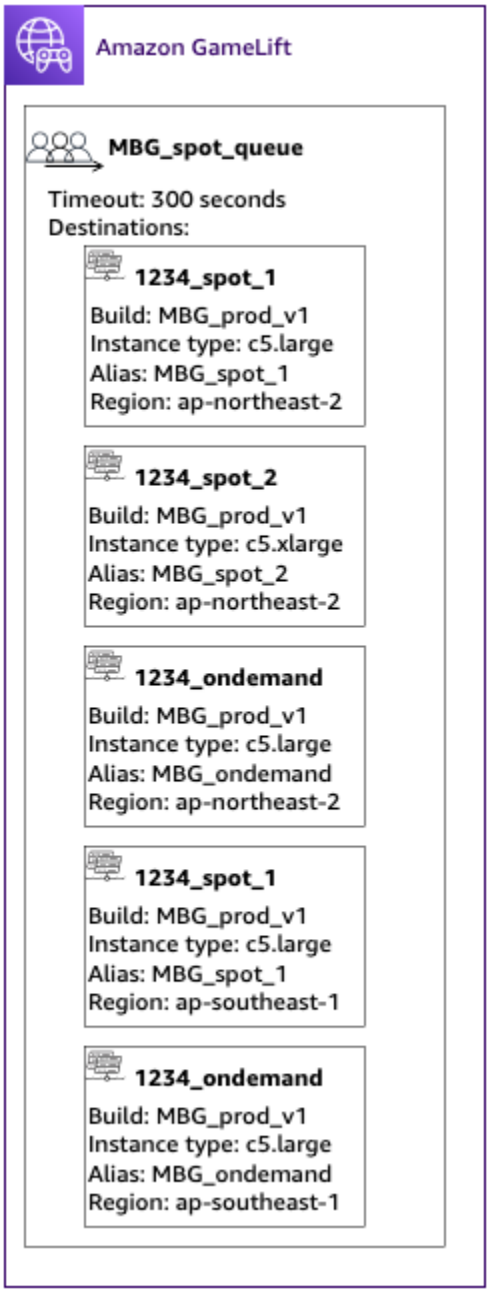
- Stellen Sie das Standard-Timeout auf 10 Minuten ein. Später kannst du testen, wie sich das Warteschlangen-Timeout auf die Wartezeiten deiner Spieler auswirkt, bis sie in die Spiele kommen.
- Überspringe vorerst den Abschnitt über die Latenzrichtlinien für Spieler. Wir werden dies im nächsten Schritt behandeln.
- Priorisieren Sie die Flotten in Ihrer Warteschlange. Bei der Arbeit mit Spot-Flotten empfehlen wir einen der folgenden Ansätze:
  - Wenn Ihre Infrastruktur einen primären Standort mit Flotten an einem zweiten Standort als Backup verwendet, priorisieren Sie Flotten zuerst nach Standort und dann nach Flottenart.

- Wenn Ihre Infrastruktur mehrere Standorte gleichmäßig nutzt, priorisieren Sie Flotten nach Flottentyp und platzieren Sie Spot-Flotten ganz oben in der Warteschlange.

Für dieses Tutorial erstellen wir eine neue Warteschlange mit dem Namen **`MBG_spot_queue`** und fügen die Aliase aller fünf unserer Flotten hinzu. Anschließend priorisieren wir die Platzierungen zunächst nach Standort und dann nach Flottenart.

Basierend auf dieser Konfiguration versucht diese Warteschlange immer, neue Spielsitzungen in einer Spot-Flotte in Seoul zu platzieren. Wenn diese Flotten voll sind, nutzt die Warteschlange die verfügbare Kapazität der Seoul On-Demand-Flotte als Backup. Wenn alle drei Flotten in Seoul nicht verfügbar sind, veranstaltet Amazon GameLift Spielsitzungen auf den Flotten von Singapur.

Das folgende Diagramm zeigt eine Warteschlange mit einem Timeout von 300 Sekunden und priorisierten Zielen. Die Ziele sind in der folgenden Reihenfolge: 1234_spot_1 in ap-northeast-2, 1234_spot_2 in ap-northeast-2, 1234_ondemand in ap-northeast-2, 1234_spot_1 in ap-Southeast-1 und 1234_ondemand in ap-Southeast-1.



Schritt 5: Fügen Sie der Warteschlange Latenzlimits hinzu

Unser Spiel enthält Informationen zur Latenz in Platzierungsanfragen für Spielsitzungen. Wir haben auch eine Funktion für Spielerpartys, die eine Spielsitzung für eine Gruppe von Spielern erstellt. Wir können die Spieler etwas länger warten lassen, bis sie in Spiele mit dem idealen Spielerlebnis einsteigen. Unsere Spieletests zeigen die folgenden Beobachtungen:

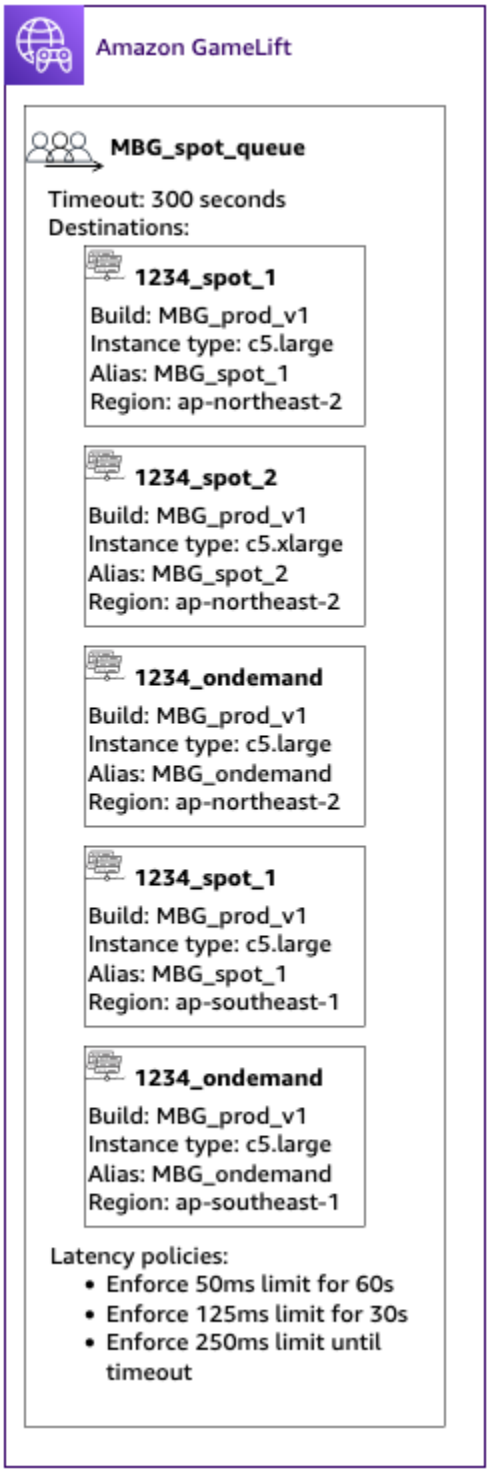
- Eine Latenz unter 50 Millisekunden ist ideal.

- Das Spiel ist bei Latenzen über 250 Millisekunden nicht spielbar.
- Die Spieler werden nach etwa einer Minute ungeduldig.

Für unsere Warteschlange mit einem Timeout von 300 Sekunden fügen wir Richtlinienanweisungen hinzu, die die zulässige Latenz einschränken. Die Richtlinienerklärungen ermöglichen schrittweise höhere Latenzwerte von bis zu 250 Millisekunden Latenz.

Mit dieser Richtlinie sucht unsere Warteschlange in der ersten Minute nach Platzierungen mit idealer Latenz (unter 50 Millisekunden) und lockert dann das Limit. Die Warteschlange führt keine Platzierungen durch, bei denen die Latenz der Spieler 250 Millisekunden oder höher beträgt.

Das folgende Diagramm zeigt die Warteschlange aus Schritt vier mit hinzugefügten Latenzrichtlinien für Spieler. Die Richtlinien für die Spielerlatenz besagen, dass das Limit von 50 ms für 60 Sekunden, das Limit von 125 ms für 30 Sekunden und das Limit von 250 ms bis zum Timeout durchgesetzt werden.



Übersicht

Herzlichen Glückwunsch! Hier sind die Dinge, die du erreicht hast:

- Sie haben eine Warteschlange für Spielsitzungen, die auf einen Teil Ihrer Spielerpopulation ausgerichtet ist.
- Ihre Warteschlange nutzt Spot-Flotten effektiv und ist widerstandsfähig, wenn Spot-Unterbrechungen auftreten.
- In deiner Warteschlange werden die Flotten priorisiert, um das beste Spielerlebnis zu bieten.
- Die Warteschlange hat Latenzbeschränkungen, um die Spieler vor schlechten Spielerlebnissen zu schützen.

Ihr könnt jetzt die Warteschlange verwenden, um Spielsitzungen für die Spieler zu veranstalten, denen sie dient. Wenn Sie Anfragen zur Platzierung von Spielsitzungen für diese Spieler stellen, geben Sie in der Anfrage den Namen der Warteschlange für die Spielsitzung an. Weitere Informationen dazu, wie du Platzierungsanfragen für Spielsitzungen stellstSpielsitzungen erstellen, findest du unter oderIntegration eines Game-Clients für Realtime Server.

Die nächsten Schritte:

- Gestalte deine eigene Warteschlange.
- Erstellen Sie eine Warteschlange.
- Benutze eine Warteschlange mit deinem Spielclient.

Ressourcen verwalten mit AWS CloudFormation

Sie können es verwendenAWS CloudFormation, um Ihre GameLift Amazon-Ressourcen zu verwalten. In AWS CloudFormation erstellen Sie eine Vorlage, die jede Ressource modelliert, und erstellen anhand der Vorlage dann Ihre Ressourcen. Um Ressourcen zu aktualisieren, nehmen Sie die Änderungen an der Vorlage vor und verwenden Sie AWS CloudFormation zum Implementieren der Updates. Sie können Ihre Ressourcen in logische Gruppen, sogenannte Stacks und Stack-Sets, organisieren.

Die Verwendung AWS CloudFormation zur Verwaltung Ihrer GameLift Amazon-Hosting-Ressourcen bietet eine effizientere Möglichkeit, AWS Ressourcengruppen zu verwalten. Sie können die Versionskontrolle verwenden, um Vorlagenänderungen im Zeitverlauf zu verfolgen und Aktualisierungen von mehreren Teammitgliedern zu koordinieren. Sie können Vorlagen auch wiederverwenden. Wenn Sie beispielsweise ein Spiel in mehreren Regionen bereitstellen, können Sie dieselbe Vorlage verwenden, um identische Ressourcen in jeder Region zu erstellen. Sie

können diese Vorlagen auch verwenden, um dieselben Ressourcensätze in einer anderen Partition bereitzustellen.

Weitere Informationen zu AWS CloudFormation finden Sie im AWS CloudFormation Benutzerhandbuch. Informationen zu Vorlagen für GameLift Amazon-Ressourcen finden Sie in der Referenz zu den GameLift Amazon-Ressourcentypen.

Bewährte Methoden

Eine ausführliche Anleitung zur Verwendung AWS CloudFormation finden Sie in den AWS CloudFormationBest Practices im AWS CloudFormationBenutzerhandbuch. Darüber hinaus sind diese Best Practices für Amazon von besonderer BedeutungGameLift.

- Verwalten Sie Ihre Ressourcen konsequent durchAWS CloudFormation. Wenn Sie Ihre Ressourcen ändern, werden AWS CloudFormation Ihre Ressourcen nicht mehr mit Ihren Ressourcenvorlagen synchronisiert.
- Verwenden Sie AWS CloudFormation-Stacks und Stack-Sets, um mehrere Ressourcen effizient zu verwalten.
  - Verwenden Sie Stacks, um Gruppen verbundener Ressourcen zu verwalten. Zum Beispiel ein Stack, der einen Build enthält, eine Flotte, die auf den Build verweist, und einen Alias, der auf die Flotte verweist. Wenn Sie Ihre Vorlage aktualisieren, um einen Build zu ersetzen, werden die mit dem Build verbundenen Flotten AWS CloudFormation ersetzt. AWS CloudFormationaktualisiert dann die vorhandenen Aliase so, dass sie auf die neuen Flotten verweisen. Weitere Informationen finden Sie im AWS CloudFormationBenutzerhandbuch unter Arbeiten mit Stacks.
  - Verwenden Sie AWS CloudFormation Stack-Sets, wenn Sie identische Stacks in mehreren Regionen oder AWS Konten bereitstellen. Weitere Informationen finden Sie im AWS CloudFormationBenutzerhandbuch unter Arbeiten mit Stacksets.
- Wenn Sie Spot-Instances verwenden, schließen Sie eine On-Demand-Flotte als Alternative ein. Wir empfehlen, Ihre Vorlagen mit zwei Flotten in jeder Region einzurichten, einer Flotte mit Spot-Instances und einer Flotte mit On-Demand-Instances.
- Gruppieren Sie Ihre standortspezifischen Ressourcen und globalen Ressourcen in separaten Stacks, wenn Sie Ressourcen an mehreren Standorten verwalten.
- Platzieren Sie Ihre globalen Ressourcen in der Nähe der Dienste, die sie nutzen. Ressourcen wie Warteschlangen und Matchmaking-Konfigurationen erhalten in der Regel ein hohes Volumen an Anfragen aus bestimmten Quellen. Indem Sie Ihre Ressourcen in der Nähe der Quelle dieser Anfragen platzieren, minimieren Sie die Reisezeit für Anfragen und können die Gesamtleistung verbessern.

- Platzieren Sie Ihre Matchmakingkonfiguration in derselben Region wie die Spielsitzungswarteschlange, die von ihr verwendet wird.
- Erstellen Sie für jede Flotte im Stack einen separaten Alias.

AWS CloudFormationStacks verwenden

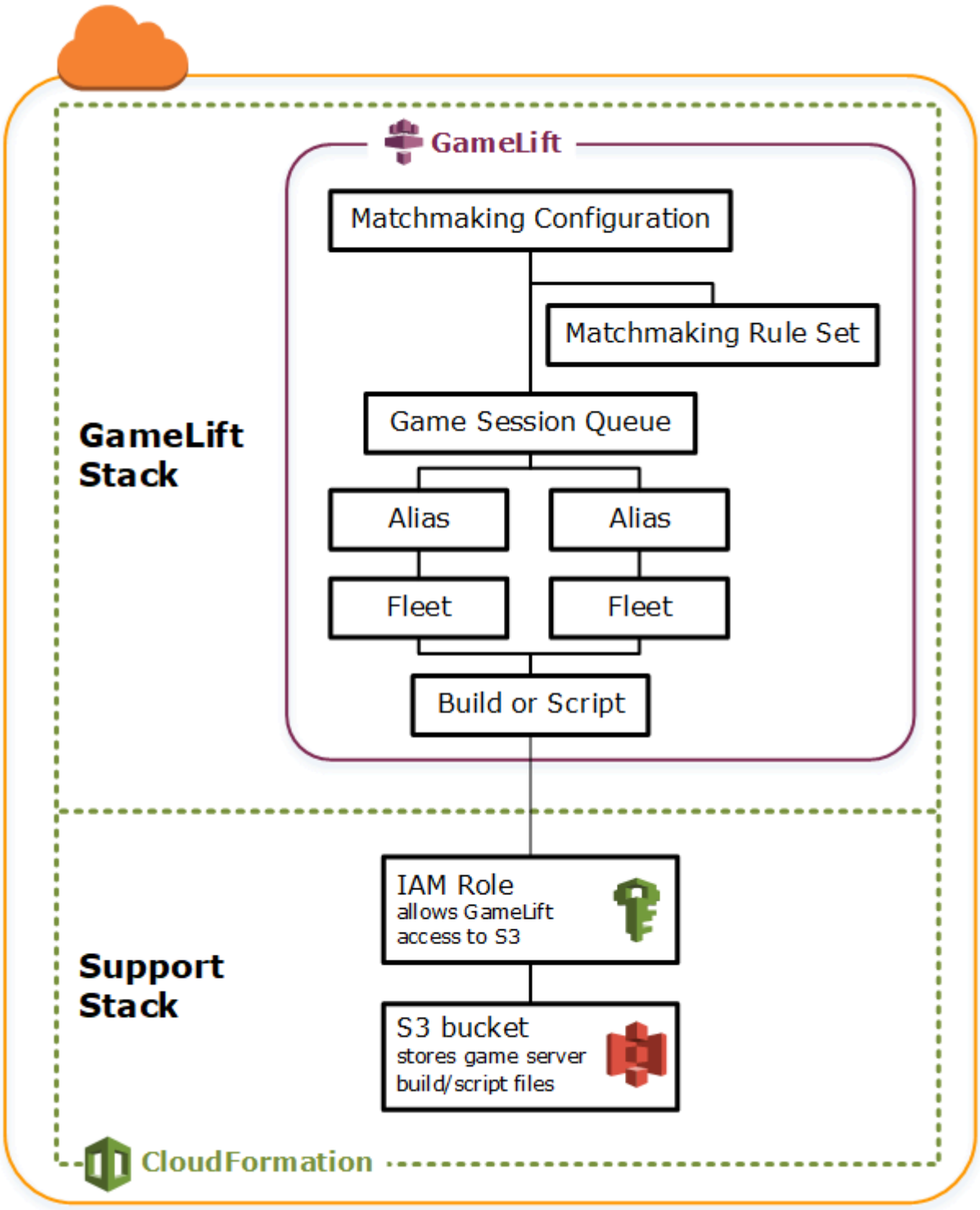
Wir empfehlen, die folgenden Strukturen für die Einrichtung von AWS CloudFormation Stacks für GameLift Amazon-Ressourcen zu verwenden. Ihre optimale Stackstruktur hängt davon ab, ob Sie Ihr Spiel an einem oder mehreren Standorten bereitstellen.

Stacks für einen einzelnen Standort

Um GameLift Amazon-Ressourcen an einem einzigen Standort zu verwalten, empfehlen wir eine zweistufige Struktur:

- Support-Stack — Dieser Stack enthält Ressourcen, von denen Ihre GameLift Amazon-Ressourcen abhängen. Dieser Stack sollte mindestens den S3-Bucket enthalten, in dem Sie Ihren benutzerdefinierten Spieleserver oder Echtzeit-Skriptdateien speichern. Der Stack sollte auch eine IAM-Rolle enthalten, die Amazon die GameLift Erlaubnis erteilt, Ihre Dateien aus dem S3-Bucket abzurufen, wenn eine GameLift Amazon-Build- oder Skriptressource erstellt wird. Dieser Stack kann auch andere AWS Ressourcen enthalten, die für Ihr Spiel verwendet werden, wie DynamoDB-Tabellen, Amazon Redshift-Cluster und Lambda-Funktionen.
- GameLiftAmazon-Stack — Dieser Stapel enthält all Ihre GameLift Amazon-Ressourcen, einschließlich des Builds oder Skripts, einer Reihe von Flotten, Aliasen und der Warteschlange für Spielsitzungen. AWS CloudFormationerstellt eine Build- oder Skriptressource mit Dateien, die im S3-Bucket-Speicherort gespeichert sind, und stellt den Build oder das Skript für eine oder mehrere Flottenressourcen bereit. Für jede Flotte sollte ein entsprechender Alias vorhanden sein. Die Spielsitzungswarteschlange verweist auf einige oder alle Flottenaliase. Wenn Sie es FlexMatch für Matchmaking verwenden, enthält dieser Stack auch eine Matchmaking-Konfiguration und einen Regelsatz.

Das folgende Diagramm zeigt eine Zwei-Stack-Struktur für die Bereitstellung von Ressourcen in einer einzelnen AWS-Region.



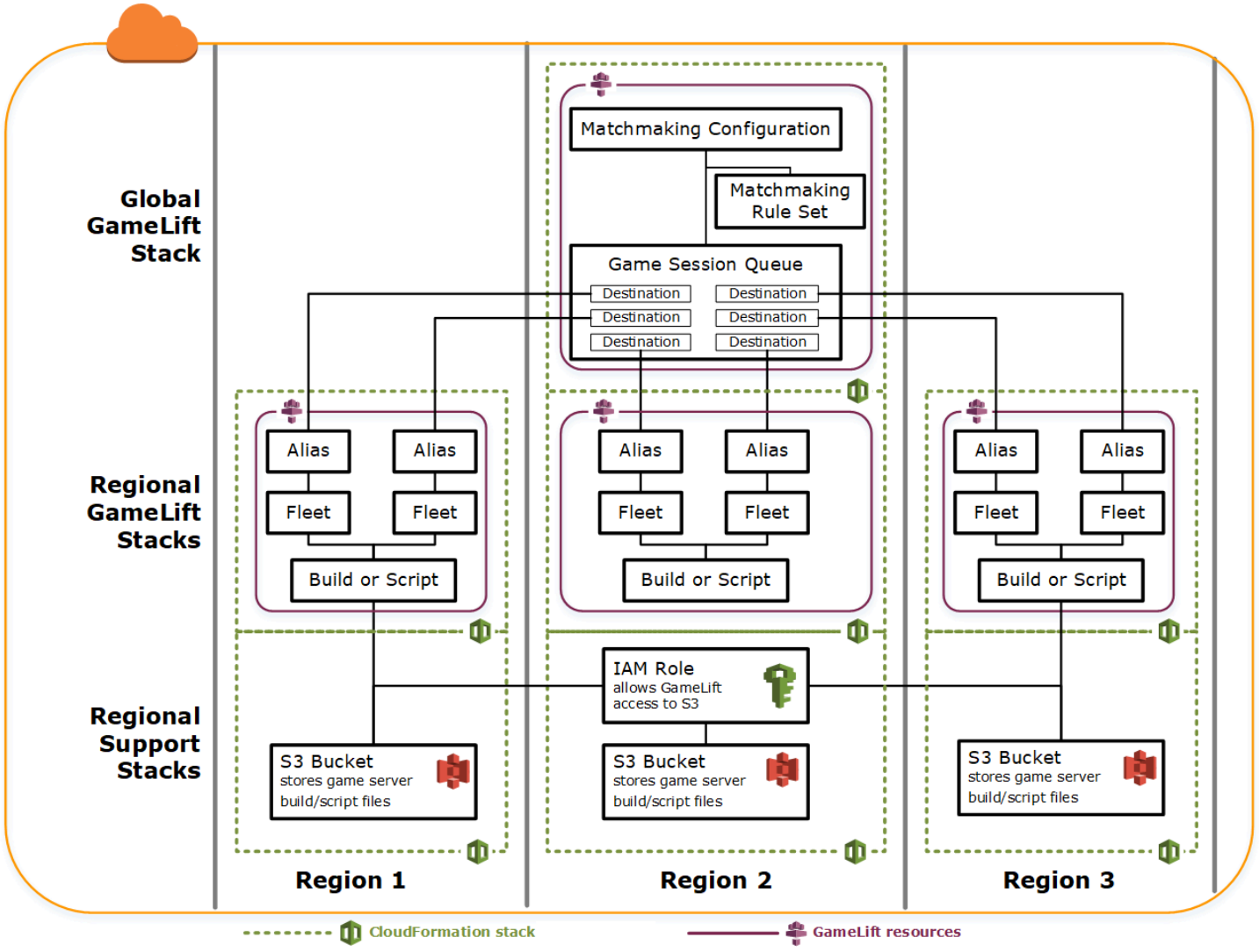
Stacks für mehrere Regionen

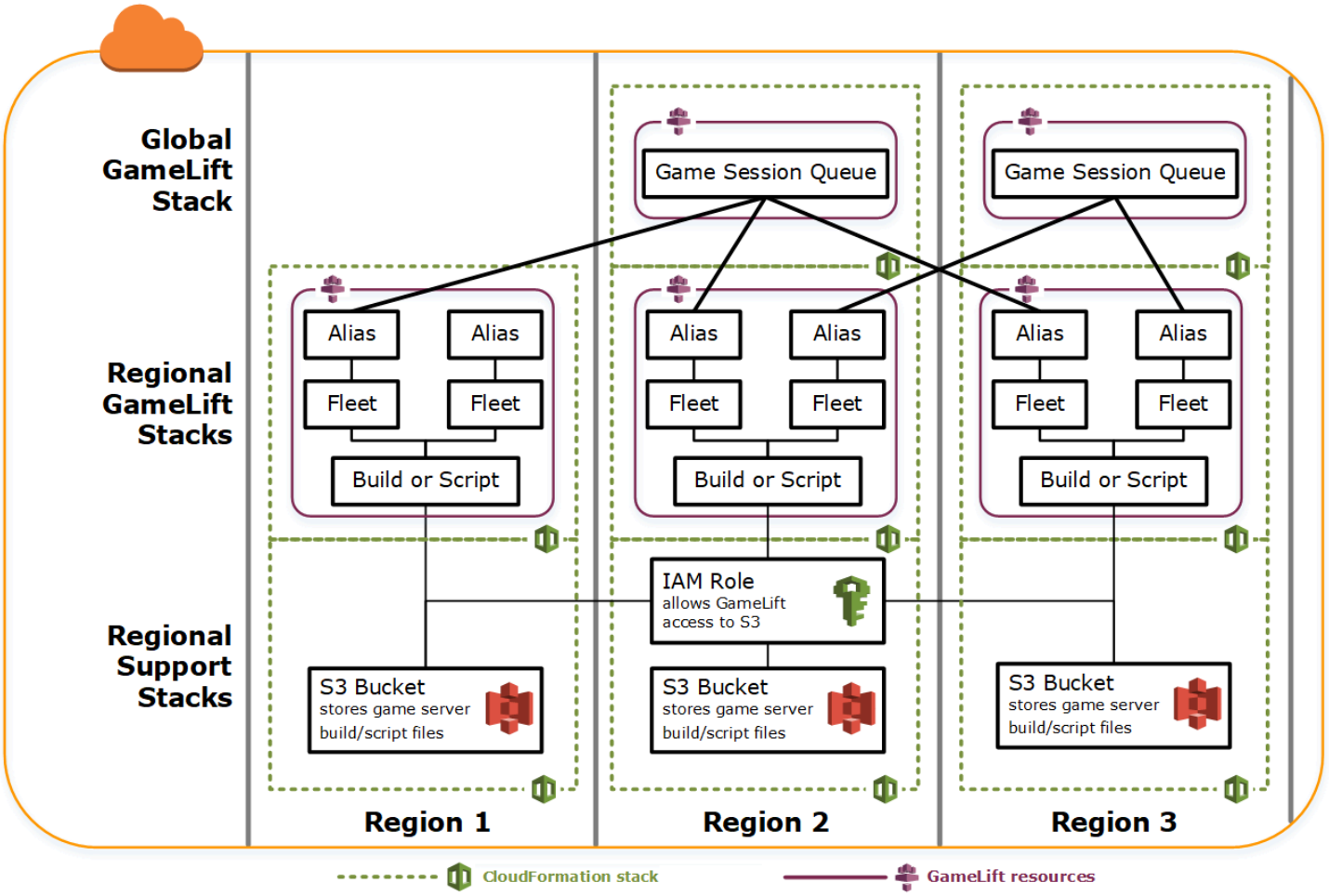
Beachten Sie bei der Bereitstellung Ihres Spiels in mehr als einer Region, wie Ressourcen in Regionen interagieren können. Einige Ressourcen, wie GameLift Amazon-Flotten, können nur auf andere Ressourcen in derselben Region verweisen. Andere Ressourcen, wie z. B. eine GameLift

Amazon-Warteschlange, sind regionsunabhängig. Um GameLift Amazon-Ressourcen in mehreren Regionen zu verwalten, empfehlen wir die folgende Struktur.

- Regionale Support-Stacks — Diese Stacks enthalten Ressourcen, von denen Ihre GameLift Amazon-Ressourcen abhängig sind. Dieser Stack muss den S3-Bucket enthalten, in dem Sie Ihren benutzerdefinierten Spieleserver oder Echtzeit-Skriptdateien speichern. Es kann auch andere AWS Ressourcen für Ihr Spiel enthalten, wie DynamoDB-Tabellen, Amazon Redshift-Cluster und Lambda-Funktionen. Viele dieser Ressourcen sind regionsspezifisch, du musst sie also in jeder Region erstellen. Amazon benötigt GameLift außerdem eine IAM-Rolle, die den Zugriff auf diese Support-Ressourcen ermöglicht. Da eine IAM-Rolle regionsunabhängig ist, benötigen Sie nur eine Rollenressource, die in einer beliebigen Region platziert und in allen anderen Support-Stacks referenziert wird.
- Regionale GameLift Amazon-Stacks — Dieser Stapel enthält die GameLift Amazon-Ressourcen, die in jeder Region vorhanden sein müssen, in der Ihr Spiel bereitgestellt wird, einschließlich des Builds oder Skripts, einer Reihe von Flotten und Aliasnamen. AWS CloudFormationerstellt eine Build- oder Skriptressource mit Dateien in einem S3-Bucket-Speicherort und stellt den Build oder das Skript für eine oder mehrere Flottenressourcen bereit. Für jede Flotte sollte ein entsprechender Alias vorhanden sein. Die Spielsitzungswarteschlange verweist auf einige oder alle Flottenaliase. Sie können eine Vorlage verwalten, um diese Art von Stack zu beschreiben und sie verwenden, um identische Sätze von Ressourcen in jeder Region zu erstellen.
- Globaler GameLift Amazon-Stack — Dieser Stapel enthält Ihre Warteschlange für Spielsitzungen und Matchmaking-Ressourcen. Diese Ressourcen können sich in jeder beliebigen Region befinden und werden normalerweise in derselben Region platziert. Die Warteschlange kann Flotten oder Aliase referenzieren, die sich in beliebigen Regionen befinden. Um zusätzliche Warteschlangen in verschiedenen Regionen zu platzieren, erstellen Sie zusätzliche globale Stacks.

Die folgenden Diagramme veranschaulichen eine Multistack-Struktur für die Bereitstellung von Ressourcen in mehreren AWS-Regionen. Das erste Diagramm zeigt eine Struktur für eine einzelne Spielsitzungs-Warteschlange. Das zweite Diagramm zeigt eine Struktur mit mehreren Warteschlangen.


Global GameLift Stack
Regional GameLift Stacks
Regional Support Stacks
Matchmaking Configuration
Matchmaking Rule Set
Game Session Queue
Destination
Alias
Fleet
Build or Script
IAM Role
allows GameLift access to S3
S3 Bucket
stores game server build/script files
Region 1
Region 2
Region 3
CloudFormation stack
GameLift resources



Aktualisierung von Builds

GameLiftAmazon-Builds sind unveränderlich, ebenso wie die Beziehung zwischen einem Build und einer Flotte. Wenn Sie Ihre Hosting-Ressourcen aktualisieren, um einen neuen Satz von Spiele-Build-Dateien zu verwenden, müssen Sie Folgendes tun:

- Erstellen Sie einen neuen Build mit dem neuen Satz von Dateien (Ersatz).
- Erstellen Sie einen neuen Satz von Flotten, um den neuen Spiel-Build (Ersatz) bereitzustellen.
- Leiten Sie Aliase um, um auf die neuen Flotten zu verweisen (Aktualisierung ohne Unterbrechung).

Weitere Informationen finden Sie im AWS CloudFormationBenutzerhandbuch unter Verhalten von Stack-Ressourcen aktualisieren.

Automatische Bereitstellung von Build-Updates

Beim Aktualisieren eines Stacks mit verwandten Build-, Flotten- und Aliasressourcen besteht das AWS CloudFormation-Standardverhalten darin, diese Schritte automatisch nacheinander auszuführen. Sie lösen dieses Update aus, indem Sie zuerst die neuen Build-Dateien an einen neuen S3-Speicherort hochladen. Anschließend ändern Sie Ihre AWS CloudFormation-Build-Vorlage so, dass sie auf den neuen S3-Standort verweist. Wenn Sie Ihren Stack mit dem neuen S3-Speicherot aktualisieren, löst dies die folgende AWS CloudFormation-Sequenz aus:

1. Ruft die neuen Dateien von S3 ab, validiert die Dateien und erstellt einen neuen GameLift Amazon-Build.
2. Aktualisiert die Build-Referenz in der Flottenvorlage, wodurch eine neue Flottenerstellung ausgelöst wird.
3. Wenn die neuen Flotten aktiv sind, wird der Flottenverweis im Alias so aktualisiert, dass der Alias auf die neuen Flotten verweist.
4. Löscht die alte Flotte.
5. Löscht den alten Build.

Wenn Ihre Spielsitzungs-Warteschlange Flottenaliase verwendet, wird der Spielerverkehr automatisch auf die neuen Flotten umgeschaltet, sobald die Aliase aktualisiert werden. Die alten Flotten werden schrittweise von Spielern entfernt, wenn Spielsitzungen enden. Auto Scaling übernimmt die Aufgabe, Instances aus jeder Gruppe von Flotten hinzuzufügen und zu entfernen, wenn der Spielerverkehr schwankt. Alternativ können Sie eine anfängliche Anzahl der gewünschten Instance angeben, um eine schnelle Umstellung und Auto Scaling zu einem späteren Zeitpunkt zu ermöglichen.

Sie können AWS CloudFormation auch dazu veranlassen, Ressourcen zu behalten, anstatt sie zu löschen. Weitere Informationen finden Sie unter RetainResources in der AWS CloudFormation-API-Referenz.

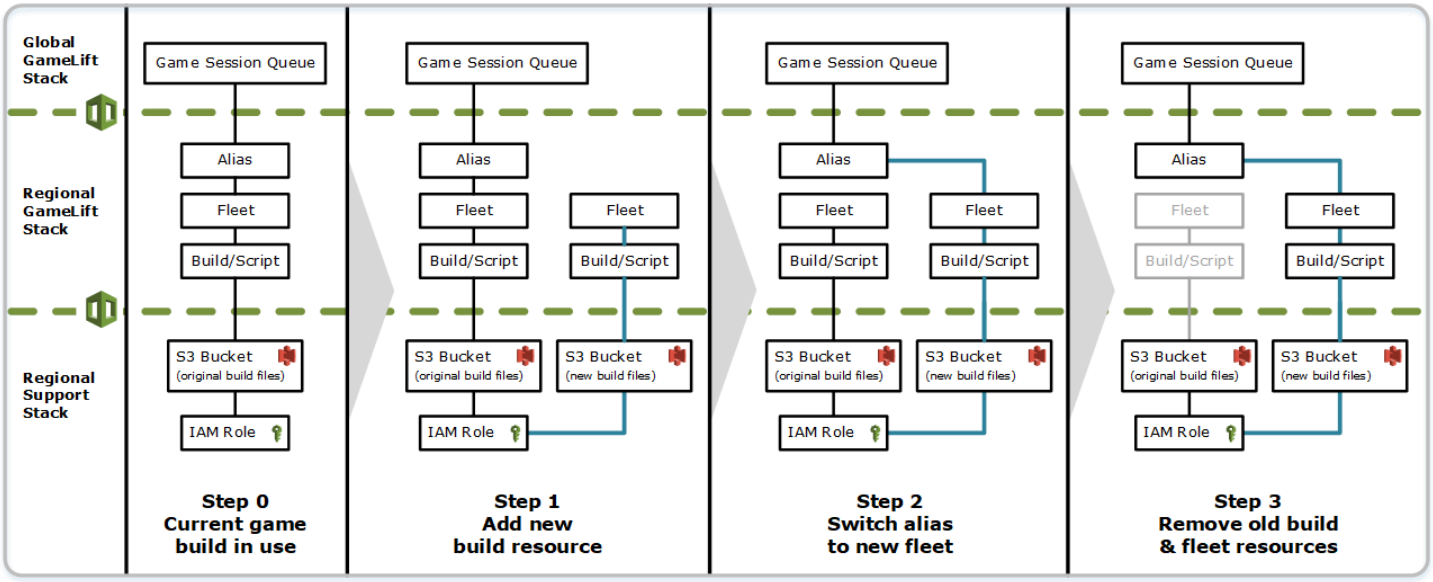
Manuelles Bereitstellen von Build-Updates

Wenn Sie mehr Kontrolle darüber haben möchten, wann neue Flotten für Spieler live gehen, bieten sich Ihnen einige Möglichkeiten. Sie können wählen, ob Sie Aliase manuell über die GameLift Amazon-Konsole oder die CLI verwalten möchten. Anstatt Ihre Build-Vorlage zu aktualisieren, um den Build und die Flotten zu ersetzen, können Sie auch einen zweiten Satz von Build- und Flottendefinitionen zur Vorlage hinzufügen. Wenn Sie die Vorlage aktualisieren, erstellt AWS

CloudFormation eine zweite Build-Ressource und entsprechende Flotten. Da die vorhandenen Ressourcen nicht ersetzt werden, werden sie nicht gelöscht und die Aliase verweisen weiterhin auf ursprüngliche Flotten.

Der Hauptvorteil bei diesem Ansatz ist, dass er Ihnen Flexibilität verleiht. Sie können separate Ressourcen für die neue Version Ihres Builds erstellen, die neuen Ressourcen testen und kontrollieren, wann die neuen Flotten für Spieler live gehen. Ein potenzieller Nachteil besteht darin, dass in jeder Region für einen kurzen Zeitraum doppelt so viele Ressourcen benötigt werden.

Das folgende Diagramm veranschaulicht diesen Prozess.



So funktionieren Rollbacks

Wenn bei einem Ressourcen-Update ein Schritt nicht erfolgreich durchgeführt wurde, löst AWS CloudFormation automatisch ein Rollback aus. Dieser Prozess kehrt jeden Schritt in Folge um und löscht die neu erstellten Ressourcen.

Wenn Sie ein Rollback manuell auslösen müssen, setzen Sie den S3-Speicherortschlüssel der Vorlage auf den ursprünglichen Speicherort zurück und aktualisieren Sie den Stack. Ein neuer GameLift Amazon-Build und eine neue Flotte werden erstellt, und der Alias wechselt auf die neue Flotte, sobald die Flotte aktiv ist. Wenn Sie Aliase separat verwalten, müssen Sie sie so umschalten, dass sie auf die neuen Flotten verweisen.

Weitere Informationen zum Umgang mit einem fehlgeschlagenen oder hängengebliebenen Rollback finden Sie im AWS CloudFormationBenutzerhandbuch unter Fortfahren des Rollbacks eines Updates.

VPC-Peering für Amazon GameLift

Dieses Thema enthält Anleitungen zum Einrichten einer VPC-Peering-Verbindung zwischen Ihren von Amazon GameLift gehosteten Spieleservern und Ihren anderen Ressourcen, die nicht von Amazon stammen. GameLift Verwenden Sie Amazon Virtual Private Cloud (VPC) -Peering-Verbindungen, damit Ihre Spieleserver direkt und privat mit Ihren anderen AWS Ressourcen wie einem Webservice oder einem Repository kommunizieren können. Sie können VPC-Peering mit allen Ressourcen einrichten, die auf einem AWS Konto ausgeführt werden AWS und von diesem verwaltet werden, auf das Sie Zugriff haben.

> **Note**
>
> VPC-Peering ist eine erweiterte Funktion. Informationen zu den bevorzugten Optionen, um deinen Spielservern die direkte und private Kommunikation mit deinen anderen AWS Ressourcen zu ermöglichen, findest du unterKommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Wenn Sie bereits mit Amazon VPCs und VPC-Peering vertraut sind, sollten Sie wissen, dass die Einrichtung von Peering mit GameLift Amazon-Spieleservern etwas anders ist. Sie haben keinen Zugriff auf die VPC, die Ihre Spieleserver enthält — sie wird vom GameLift Amazon-Dienst gesteuert —, sodass Sie VPC-Peering dafür nicht direkt anfordern können. Stattdessen autorisieren Sie die VPC zunächst mit Ihren GameLift Ressourcen, die nicht von Amazon stammen, um eine Peering-Anfrage vom Amazon-Service anzunehmen. GameLift Dann veranlassen Sie AmazonGameLift, das VPC-Peering anzufordern, das Sie gerade autorisiert haben. Amazon GameLift übernimmt die Aufgaben der Erstellung der Peering-Verbindung, der Einrichtung der Routing-Tabellen und der Konfiguration der Verbindung.

So richten Sie VPC-Peering für eine bestehende Flotte ein

1. Holen Sie sich AWS Konto-IDs und Anmeldeinformationen.

   Für die folgenden AWS Konten benötigen Sie eine ID und Anmeldeinformationen. Sie finden AWS Konto-IDs, indem Sie sich bei der anmelden AWS Management Consoleund Ihre Kontoeinstellungen aufrufen. Die Anmeldeinformationen können Sie von der IAM-Konsole aus abrufen.

   - AWSKonto, das Sie zur Verwaltung Ihrer GameLift Amazon-Spieleserver verwenden.

- AWSKonto, mit dem Sie Ihre GameLift Ressourcen verwalten, die nicht von Amazon stammen.

Wenn Sie dasselbe Konto für Amazon-Ressourcen GameLift und GameLift Ressourcen verwenden, die nicht von Amazon stammen, benötigen Sie nur eine ID und Anmeldeinformationen für dieses Konto.

2. Kennungen für jede VPC abrufen.

   Rufen Sie die folgenden Informationen für die beiden über Peering zu verbindenden VPCs ab:

   - VPC für Ihre GameLift Amazon-Spieleserver — Dies ist Ihre GameLift Amazon-Flotten-ID. Ihre Spieleserver werden in Amazon GameLift auf einer Flotte von EC2-Instances bereitgestellt. Eine Flotte wird automatisch in einer eigenen VPC platziert, die vom GameLift Amazon-Dienst verwaltet wird. Sie haben keinen direkten Zugriff auf die VPC, daher wird sie anhand der Flotten-ID identifiziert.
   - VPC für Ihre GameLift AWS Ressourcen, die nicht von Amazon stammen — Sie können ein VPC-Peering mit allen Ressourcen einrichten, die auf einem AWS Konto ausgeführt werden AWS und von diesem verwaltet werden, auf das Sie Zugriff haben. Falls Sie noch keine VPC für diese Ressourcen erstellt haben, finden Sie weitere Informationen unter Erste Schritte mit Amazon VPC. Nachdem Sie eine VPC erstellt haben, finden Sie die VPC-ID, indem Sie sich bei der AWS Management Console für Amazon VPC anmelden und Ihre VPCs anzeigen.

   > **Note**
   >
   > Beim Einrichten eines Peerings müssen beide VPCs in der selben Region existieren. Die VPC für Ihre Amazon GameLift Fleet-Spieleserver befindet sich in derselben Region wie die Flotte.

3. Autorisieren Sie ein VPC-Peering.

   In diesem Schritt autorisieren Sie eine zukünftige Anfrage von Amazon vorab, GameLift um die VPC mit Ihren Spielservern mit Ihrer VPC für Ressourcen zu verbinden, die nicht von Amazon stammen. GameLift Diese Aktion aktualisiert die Sicherheitsgruppe für Ihre VPC.

   Um das VPC-Peering zu autorisieren, rufen Sie die Amazon GameLift Service API CreateVpcPeeringAuthorization() auf oder verwenden Sie den AWS CLI-Befehl. `create-vpc-peering-authorization` Führen Sie diesen Anruf mit dem Konto durch, das Ihre GameLift

Ressourcen verwaltet, die nicht von Amazon stammen. Geben Sie die folgenden Informationen an:

- Peer-VPC-ID — Dies ist für die VPC mit Ihren Ressourcen, die nicht GameLift von Amazon stammen.
- GameLiftAWSAmazon-Konto-ID — Dies ist das Konto, mit dem Sie Ihre GameLift Amazon-Flotte verwalten.

Sobald Sie ein VPC-Peering autorisiert haben, bleibt die Autorisierung 24 Stunden lang gültig, sofern sie nicht widerrufen wird. Sie können Ihre VPC-Peering-Autorisierungen unter Verwendung der folgenden Operationen verwalten:

- DescribeVpcPeeringAuthorizations() (AWSCLI`describe-vpc-peering-authorizations`).
- DeleteVpcPeeringAuthorization() (AWSCLI`delete-vpc-peering-authorization`).

4. Fordern Sie eine Peering-Verbindung an.

   Mit einer gültigen Autorisierung können Sie Amazon GameLift auffordern, eine Peering-Verbindung herzustellen.

   Um ein VPC-Peering anzufordern, rufen Sie die Amazon GameLift Service API CreateVpcPeeringConnection() auf oder verwenden Sie den AWS CLI-Befehl. `create-vpc-peering-connection` Tätigen Sie diesen Anruf mit dem Konto, das Ihre GameLift Amazon-Spieleserver verwaltet. Verwenden Sie die folgenden Informationen, um die beiden VPCs zu identifizieren, zwischen denen Sie eine Peer-Verbindung herstellen möchten:

   - Peer-VPC-ID und AWS Konto-ID — Dies ist die VPC für Ihre GameLift Ressourcen, die nicht von Amazon stammen, und das Konto, mit dem Sie diese verwalten. Die VPC-ID muss mit der ID einer gültigen Peering-Autorisierung übereinstimmen.
   - Fleet ID — Dies identifiziert die VPC für Ihre GameLift Amazon-Spieleserver.

5. Verfolgen Sie den Status der Peering-Verbindung.

   Die Anforderung einer VPC-Peering-Verbindung ist ein asynchroner Vorgang. Um den Status einer Peering-Anfrage nachzuverfolgen und den Erfolg oder das Fehlschlagen zu verarbeiten, verwenden Sie eine der folgenden Optionen:

- Stetige Abfrage mit `DescribeVpcPeeringConnections()`. Diese Operation ruft den Datensatz für die VPC-Peering-Verbindung ab, einschließlich des Status der Anforderung. Wenn eine Peering-Verbindung erfolgreich erstellt wurde, enthält der Verbindungsdatensatz auch einen CIDR-Block mit privaten IP-Adressen, der der VPC zugewiesen wird.
- Behandeln Sie Flottenereignisse im Zusammenhang mit VPC-Peering-Verbindungen mit DescribeFleetEvents(), einschließlich Erfolgs- und Ausfallereignissen.

Sobald die Peering-Verbindung hergestellt wurde, können Sie sie mit den folgenden Vorgängen verwalten:

- DescribeVpcPeeringConnections() (AWSCLI`describe-vpc-peering-connections`).
- DeleteVpcPeeringConnection() (AWSCLI`delete-vpc-peering-connection`).

So richten Sie VPC-Peering mit einer neuen Flotte ein

Sie können eine neue GameLift Amazon-Flotte erstellen und gleichzeitig eine VPC-Peering-Verbindung anfordern.

1. Holen Sie sich AWS Konto-IDs und Anmeldeinformationen.

   Für die folgenden beiden AWS Konten benötigen Sie eine ID und Anmeldeinformationen. Sie finden AWS Konto-IDs, indem Sie sich bei der anmelden AWS Management Consoleund Ihre Kontoeinstellungen aufrufen. Die Anmeldeinformationen können Sie von der IAM-Konsole aus abrufen.

   - AWSKonto, das Sie zur Verwaltung Ihrer GameLift Amazon-Spieleserver verwenden.
   - AWSKonto, mit dem Sie Ihre GameLift Ressourcen verwalten, die nicht von Amazon stammen.

   Wenn Sie dasselbe Konto für Amazon-Ressourcen GameLift und GameLift Ressourcen verwenden, die nicht von Amazon stammen, benötigen Sie nur eine ID und Anmeldeinformationen für dieses Konto.
2. Holen Sie sich die VPC-ID für Ihre Ressourcen, die nicht von Amazon stammen GameLiftAWS.

   Wenn Sie noch keine VPC für diese Ressourcen erstellt haben, tun Sie dies jetzt (siehe Erste Schritte mit Amazon VPC). Stellen Sie sicher, dass Sie die neue VPC in derselben Region erstellen, in der Sie Ihre neue Flotte erstellen möchten. Wenn Ihre GameLift Ressourcen,

die nicht von Amazon stammen, unter einem anderen AWS Konto oder einer anderen Benutzergruppe verwaltet werden als dem, das Sie bei Amazon verwendenGameLift, müssen Sie diese Kontoanmeldeinformationen verwenden, wenn Sie im nächsten Schritt die Autorisierung beantragen.

Nachdem Sie eine VPC erstellt haben, finden Sie die VPC-ID in der Amazon-VPC-Konsole, indem Sie Ihre VPCs anzeigen.

3. Autorisieren Sie ein VPC-Peering mit Ressourcen, die nicht von Amazon stammen. GameLift

   Wenn Amazon die neue Flotte und eine entsprechende VPC GameLift erstellt, sendet es auch eine Anfrage zum Peering mit der VPC für Ihre Ressourcen, die nicht GameLift von Amazon stammen. Sie müssen diese Anfrage vorab autorisieren. Dieser Schritt aktualisiert die Sicherheitsgruppe für Ihre VPC.

   Rufen Sie mithilfe der Kontoanmeldeinformationen, mit denen Ihre GameLift Ressourcen außerhalb von Amazon verwaltet werden, die Amazon GameLift Service API CreateVpcPeeringAuthorization() auf oder verwenden Sie den AWS CLI-Befehl. `create-vpc-peering-authorization` Geben Sie die folgenden Informationen an:

   - Peer-VPC-ID — ID der VPC mit Ihren Ressourcen, die nicht GameLift von Amazon stammen.
   - GameLiftAWSAmazon-Konto-ID — ID des Kontos, das Sie zur Verwaltung Ihrer GameLift Amazon-Flotte verwenden.

   Sobald Sie ein VPC-Peering autorisiert haben, bleibt die Autorisierung 24 Stunden lang gültig, sofern sie nicht widerrufen wird. Sie können Ihre VPC-Peering-Autorisierungen unter Verwendung der folgenden Operationen verwalten:

   - DescribeVpcPeeringAuthorizations() (AWSCLI`describe-vpc-peering-authorizations`).
   - DeleteVpcPeeringAuthorization() (AWSCLI`delete-vpc-peering-authorization`).
4. Folgen Sie den Anweisungen zum Erstellen einer neuen Flotte mithilfe der AWS CLI. Fügen Sie die folgenden zusätzlichen Parameter ein:

   - peer-vpc-aws-account-id — ID für das Konto, mit dem Sie die VPC mit Ihren Ressourcen verwalten, die nicht GameLift von Amazon stammen.
   - peer-vpc-id— ID der VPC mit Ihrem GameLift Nicht-Konto.

Ein erfolgreicher Aufruf von create-fleet mit den VPC-Peering-Parametern generiert sowohl eine neue Flotte als auch eine neue VPC-Peering-Anfrage. Der Status der Flotte wird auf New gesetzt, und der Flottenaktivierungsprozess wird veranlasst. Der Status der Peering-Verbindungsanforderung wird auf initiating-request gesetzt. Sie können den Erfolg oder Misserfolg der Peering-Anfrage verfolgen, indem Sie anrufen describe-vpc-peering-connections.

Beim Anfordern einer neuen Flotte und einer VPC-Peering-Verbindung sind beide Aktionen erfolgreich oder schlagen fehl. Wenn eine Flotte während des Erstellungsprozesses fehlschlägt, wird die VPC-Peering-Verbindung nicht hergestellt. Analog dazu gilt, wenn eine VPC-Peering-Verbindung aus irgendeinem Grund fehlschlägt, kann auch die neue Flotte nicht vom Status Activating in den Status Active übergehen.

> **Note**
>
> Die neue VPC-Peering-Verbindung wird erst hergestellt, wenn die Flotte zur Aktivierung bereit ist. Dies bedeutet, dass die Verbindung nicht verfügbar ist und während der Installation des Spielservers nicht verwendet werden kann.

Im folgenden Beispiel wird sowohl eine neue Flotte als auch eine Peering-Verbindung zwischen einer vordefinierten VPC und der VPC für die neue Flotte erstellt. Die vorab eingerichtete VPC wird anhand der Kombination Ihrer GameLift AWS Konto-ID, die nicht von Amazon stammt, und der VPC-ID eindeutig identifiziert.

```
$ AWS gamelift create-fleet
    --name "My_Fleet_1"
    --description "The sample test fleet"
    --ec2-instance-type "c5.large"
    --fleet-type "ON_DEMAND"
    --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
    --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
                             MaxConcurrentGameSessionActivations=2,
                             ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated
\MultiplayerSampleProjectLauncher_Server.exe,
                                               Parameters=+sv_port 33435 +start_lobby,
                                               ConcurrentExecutions=10}]"
    --new-game-session-protection-policy "FullProtection"
    --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
                                      PolicyPeriodInMinutes=15"
```

```
    --ec2-inbound-permissions
 "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"

 "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP"
    --metric-groups  "EMEAfleets"
    --peer-vpc-aws-account-id "111122223333"
    --peer-vpc-id "vpc-a11a11a"
```

Kopierbare Version:

```
AWS gamelift create-fleet --name "My_Fleet_1" --description "The
 sample test fleet" --fleet-type "ON_DEMAND" --metric-groups
 "EMEAfleets" --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
 --ec2-instance-type "c5.large" --runtime-configuration
 "GameSessionActivationTimeoutSeconds=300,MaxConcurrentGameSessionActivations=2,ServerProcesses
\game\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe,Parameters=
+sv_port 33435 +start_lobby,ConcurrentExecutions=10}]" --new-game-session-
protection-policy "FullProtection" --resource-creation-limit-policy
 "NewGameSessionsPerCreator=3,PolicyPeriodInMinutes=15" --ec2-inbound-
permissions "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
 "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" --peer-vpc-aws-account-id
 "111122223333" --peer-vpc-id "vpc-a11a11a"
```

Beheben von VPC-Peering-Problemen

Wenn Sie Probleme haben, eine VPC-Peering-Verbindung für Ihre GameLift Amazon-Spieleserver herzustellen, sollten Sie die folgenden häufigen Ursachen in Betracht ziehen:

- Die Autorisierung für die angeforderte Verbindung wurde nicht gefunden.
  - Überprüfen Sie den Status einer VPC-Autorisierung für eine VPC, die nicht von Amazon stammtGameLift. Sie ist möglicherweise nicht vorhanden oder abgelaufen.
  - Überprüfen Sie die Regionen der beiden VPCs, zwischen denen Sie eine Peer-Verbindung herstellen möchten. Wenn sie sich nicht in der gleichen Region befinden, kann keine Peer-Verbindung hergestellt werden.
- Die CIDR-Blöcke (siehe Ungültige VPC-Peering-Verbindungskonfigurationen) Ihrer beiden VPCs überschneiden sich. Die IPv4-CIDR-Blöcke, die über Peering verbundenen VPCs zugewiesen sind, dürfen sich nicht überschneiden. Der CIDR-Block der VPC für Ihre GameLift Amazon-Flotte wird automatisch zugewiesen und kann nicht geändert werden. Daher müssen Sie den CIDR-Block für die VPC für Ihre Nicht-Amazon-Ressourcen ändern. GameLift So beheben Sie dieses Problem

  - Suchen Sie diesen CIDR-Block für Ihre GameLift Amazon-Flotte, indem Sie anrufen`DescribeVpcPeeringConnections()`.
  - Gehen Sie zur Amazon VPC-Konsole, suchen Sie die VPC für Ihre GameLift Nicht-Amazon-Ressourcen und ändern Sie den CIDR-Block, damit sie sich nicht überschneiden.
- Die neue Flotte wurde nicht aktiviert (wenn VPC-Peering mit einer neuen Flotte angefordert wurde). Wenn die neue Flotte nicht auf den Status „Aktiv“ gewechselt hat, ist keine VPC vorhanden, mit der eine Peering-Verbindung hergestellt werden kann.

Deine Spieldaten auf der Konsole ansehen

Der verwaltete GameLift Amazon-Dienst sammelt kontinuierlich Daten für aktive Spiele, damit Sie das Verhalten und die Leistung der Spieler besser verstehen können. Mit der GameLift Amazon-Konsole können Sie diese Informationen für Ihre Builds, Flotten, Spielsitzungen und Spielersitzungen anzeigen, verwalten und analysieren.

Themen

Sehen Sie sich Ihren aktuellen GameLift Amazon-Status an

Das GameLift Amazon-Dashboard bietet einen Überblick über Folgendes:

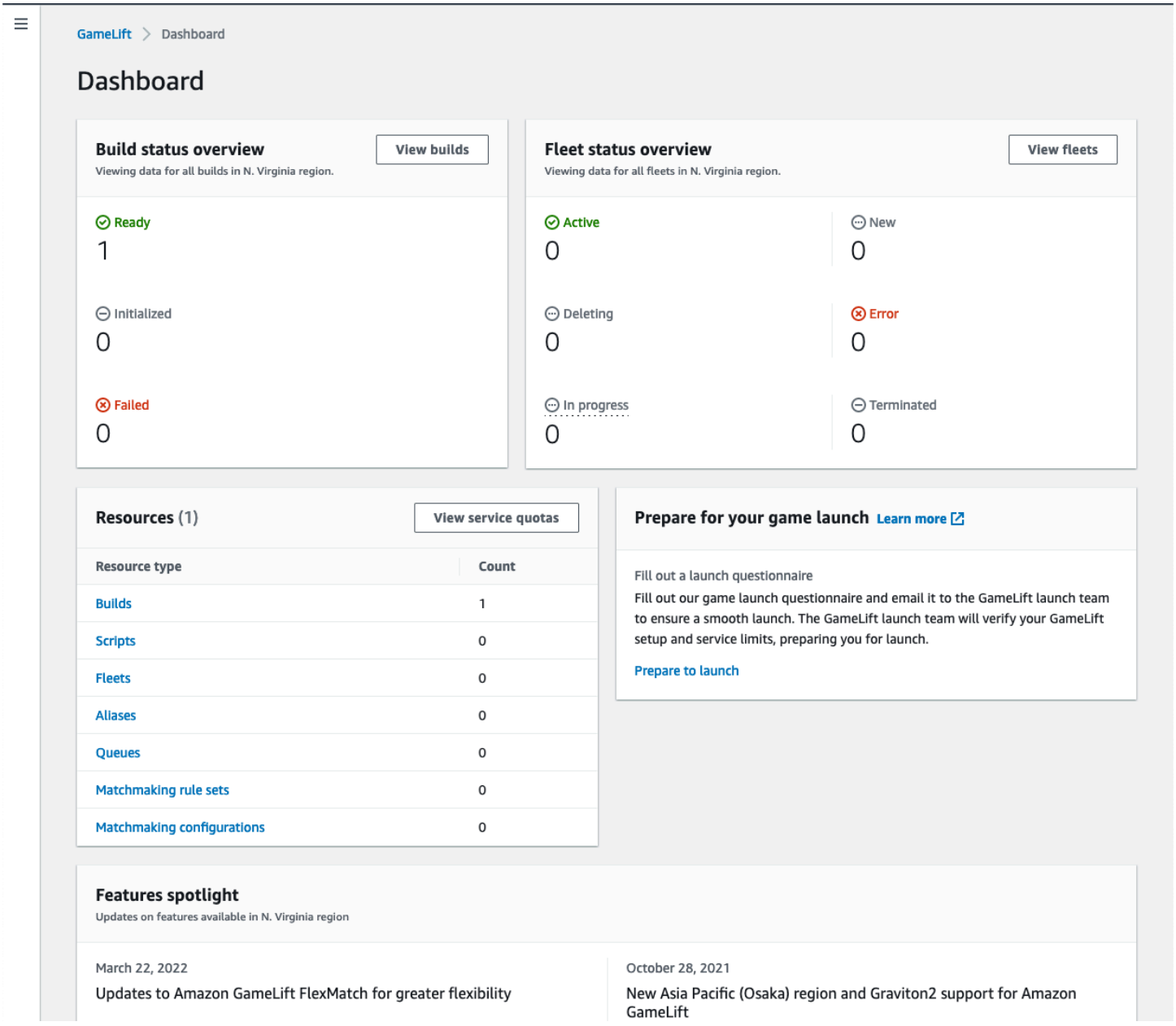
- Die Anzahl der Builds mit den Status Bereit, Initialisiert und Fehlgeschlagen. Wählen Sie Builds anzeigen aus, um weitere Informationen zu Builds in Ihrer aktuellen Region zu erhalten.
- Die Anzahl der Flotten in allen Status. Wählen Sie „Flotten anzeigen“, um weitere Informationen zu Flotten in Ihrer aktuellen Region zu erhalten.
- Ihre aktuellen Ressourcen.
- Ankündigungen neuer Funktionen und Dienste.

Um das GameLift Amazon-Dashboard zu öffnen

- Wählen Sie in der GameLift Amazon-Konsole im Navigationsbereich Dashboard aus.

Vom Dashboard aus können Sie:

- Bereite dein Spiel für den Start vor, indem du „Für den Start vorbereiten“ auswählst und den entsprechenden Fragebogen zur Veröffentlichung ausfüllst.
- Fordere als Vorbereitung oder als Reaktion auf Produkteinführungen eine Erhöhung der Servicekontingenten an, indem du „Servicekontingente anzeigen“ auswählst.
- Um Blogbeiträge und detaillierte Informationen zu neuen Funktionen anzuzeigen, klicken Sie auf den Link im Feature-Spotlight.



Sehen Sie sich Ihre Builds an

Auf der Build-Seite der GameLiftAmazon-Konsole können Sie Informationen zu allen Gameserver-Builds einsehen und diese verwalten, die Sie auf Amazon hochgeladen habenGameLift. Wählen Sie im Navigationsbereich Hosting, Builds aus.

Auf der Seite Builds werden für jeden Build die folgenden Informationen angezeigt:

> **Note**
>
> Auf der Seite Builds werden nur Builds in Ihrer aktuellen AWS Region angezeigt.

- Name — Der Name, der dem hochgeladenen Build zugeordnet ist.
- Status — Der Status des Builds. Zeigt eine von drei Statusmeldungen an:
  - Initialisiert — Der Upload wurde nicht gestartet oder ist noch im Gange.
  - Bereit — Der Build ist bereit für die Flottenerstellung.
  - Fehlgeschlagen — Beim Build ist ein Timeout aufgetreten, bevor Amazon die Binärdateien GameLift erhalten hat.
- Erstellungszeit — Datum und Uhrzeit, zu denen Sie den Build auf Amazon hochgeladen habenGameLift.
- Build-ID — Die eindeutige ID, die dem Build beim Upload zugewiesen wurde.
- Version — Die Versionsbezeichnung, die dem hochgeladenen Build zugeordnet ist.
- Betriebssystem — Das Betriebssystem, auf dem der Build ausgeführt wird. Das Build-Betriebssystem bestimmt, welches Betriebssystem Amazon auf den GameLift Instances einer Flotte installiert.
- Größe — Die Größe der auf Amazon hochgeladenen Build-Datei in Megabyte (MB). GameLift
- Flotten — Die Anzahl der Flotten, die mit dem Build eingesetzt wurden.

Auf dieser Seite können Sie die folgenden Schritte ausführen:

- Zeigen Sie die Build-Details an. Wählen Sie den Namen eines Builds, um die zugehörige Build-Detailseite zu öffnen.
- Erstellen Sie eine neue Flotte von einem Build. Wählen Sie einen Build aus und wählen Sie dann Flotte erstellen aus.

- Filtern und sortieren Sie die Build-Liste. Verwenden Sie die Steuerelemente oben in der Tabelle.
- Löschen Sie einen Build. Wählen Sie einen Build aus und wählen Sie dann Löschen.

Einzelheiten zum Bau

Wählen Sie auf der Seite Builds den Namen eines Builds aus, um die zugehörige Detailseite zu öffnen. Im Abschnitt „Übersicht“ der Detailseite werden dieselben Build-Zusammenfassungsinformationen angezeigt wie auf der Seite „Builds“. Der Abschnitt Flotten enthält eine Liste der Flotten, die mit dem Build erstellt wurden, einschließlich derselben zusammenfassenden Informationen wie auf der Seite Flotten.

Sehen Sie sich Ihre Skripte an

Auf der Seite „Skripts“ der GameLiftAmazon-Konsole können Sie Informationen über alle Realtime Server-Skripts, die Sie auf Amazon GameLift hochgeladen haben, einsehen und verwalten. Wählen Sie im Navigationsbereich Hosting, Scripts aus.

Auf der Seite „Skripte“ werden für jedes Skript die folgenden Informationen angezeigt:

> **Note**
>
> Auf der Seite „Skripte“ werden nur Skripte in Ihrer aktuellen AWS Region angezeigt.

- Name — Der Name, der dem hochgeladenen Skript zugeordnet ist.
- ID — Die eindeutige ID, die dem Skript beim Upload zugewiesen wurde.
- Version — Die Versionsbezeichnung, die dem hochgeladenen Skript zugeordnet ist.
- Größe — Die Größe der auf Amazon hochgeladenen Skriptdatei in Megabyte (MB). GameLift
- Erstellungszeit — Datum und Uhrzeit, zu denen Sie das Skript auf Amazon hochgeladen habenGameLift.
- Flotten — Die Anzahl der Flotten, die mit dem Skript eingesetzt wurden.

Auf dieser Seite können Sie die folgenden Schritte ausführen:

- Sehen Sie sich die Skriptdetails an. Wählen Sie den Namen eines Builds, um die zugehörige Skriptdetailseite zu öffnen.

- Erstellen Sie eine neue Flotte anhand eines Skripts. Wählen Sie ein Skript aus und wählen Sie dann Create fleet aus.
- Filtern und sortieren Sie die Skriptliste. Verwenden Sie die Steuerelemente oben in der Tabelle.
- Löscht ein Skript. Wählen Sie ein Skript aus und wählen Sie dann Löschen.

Einzelheiten zum Skript

Wählen Sie auf der Seite Scripts den Namen eines Skripts aus, um dessen Detailseite zu öffnen. Im Abschnitt „Übersicht“ der Detailseite werden dieselben Skriptzusammenfassungsinformationen angezeigt wie auf der Seite „Builds“. Der Abschnitt Flotten enthält eine Liste von Flotten, die mit dem Skript erstellt wurden, einschließlich derselben zusammenfassenden Informationen wie auf der Seite Flotten.

Sehen Sie sich Ihre Flotten an

In Ihrem AWS Konto können Sie Informationen zu allen Flotten einsehen, die für das Hosten Ihrer Spiele bei GameLift Amazon eingerichtet wurden. Die Liste zeigt Flotten, die in Ihrer aktuellen Region erstellt wurden. Auf der Seite Flotten können Sie eine neue Flotte erstellen oder zusätzliche Details zu einer Flotte anzeigen. Die Detailseite einer Flotte enthält Nutzungsinformationen, Kennzahlen, Spielsitzungsdaten und Spielersitzungsdaten. Sie können auch einen Flottendatensatz bearbeiten oder eine Flotte löschen.

Um die Seite Flotten aufzurufen, wählen Sie im Navigationsbereich Flotten aus.

Die Seite Fleets (Flotten) zeigt standardmäßig die folgenden zusammenfassenden Informationen an. Sie können die angezeigten Informationen anpassen, indem Sie auf die Schaltfläche Einstellungen (Zahnrad) klicken.

- Name — Freundlicher Name, der der Flotte gegeben wurde.
- Status — Der Status der Flotte. Dabei kann es sich um einen der folgenden Zustände handeln: Neu, Wird heruntergeladen, In Aufbau und Aktiv.
- Erstellungszeit — Datum und Uhrzeit der Erstellung der Flotte.
- Berechnungstyp — Die Art der Rechenleistung, die zum Hosten deiner Spiele verwendet wird. Eine Flotte kann eine verwaltete EC2-Flotte oder eine Anywhere-Flotte sein.
- Instance-Typ — Der Amazon EC2-Instance-Typ, der die Rechenkapazität der Instances der Flotte bestimmt.

- Aktive Instances — Die Anzahl der EC2-Instances, die für die Flotte verwendet werden.
- Gewünschte Instances — Die Anzahl der EC2-Instances, die aktiv bleiben sollen.
- Spielsitzungen — Die Anzahl der aktiven Spielsitzungen, die in der Flotte laufen. Die Daten werden um fünf Minuten verzögert.

Flottendetails anzeigen

Greifen Sie über das Dashboard oder die Flottenseite auf eine Flottendetailseite zu, indem Sie den Flottennamen auswählen.

Auf der Seite mit den Flottendetails können Sie die folgenden Maßnahmen ergreifen:

- Aktualisieren Sie die Attribute, Porteinstellungen und Laufzeitkonfiguration einer Flotte.
- Fügen Sie Flottenstandorte hinzu oder entfernen Sie sie.
- Ändern Sie die Flottenkapazitätseinstellungen.
- Stellen Sie die automatische Skalierung für die Zielverfolgung ein oder ändern Sie sie.
- Löschen einer Flotte.

Details

Flotteneinstellungen

- Flotten-ID — Der Flotte zugewiesene eindeutige Kennung.
- Name — Der Name der Flotte.
- ARN — Die dieser Flotte zugewiesene Kennung. Der ARN einer Flotte identifiziert sie als GameLift Amazon-Ressource und gibt die Region und das AWS Konto an.
- Beschreibung — Eine kurze, identifizierbare Beschreibung der Flotte.
- Status — Aktueller Status der Flotte. Dieser kann „Neu“, „Wird heruntergeladen“, „In Aufbau“ und „Aktiv“ lauten.
- Erstellungszeit — Datum und Uhrzeit der Erstellung der Flotte.
- Kündigungszeit — Datum und Uhrzeit der Einstellung der Flotte. Dieses Feld ist leer, wenn die Flotte noch aktiv ist.
- Flottentyp — Gibt an, ob die Flotte On-Demand-Instances oder Spot-Instances verwendet.
- EC2-Typ — Amazon EC2-Instance-Typ, der bei der Erstellung für die Flotte ausgewählt wurde.

- Instanzrolle — Eine AWS IAM-Rolle, die den Zugriff auf Ihre anderen AWS Ressourcen verwaltet, sofern eine bei der Flottenerstellung bereitgestellt wurde.
- TLS-Zertifikat — Gibt an, ob die Flotte aktiviert oder deaktiviert ist, um ein TLS-Zertifikat zur Authentifizierung eines Spieleservers und zur Verschlüsselung der gesamten Client/Server-Kommunikation zu verwenden.
- Metrische Gruppe — Die Gruppe, die verwendet wird, um Metriken für mehrere Flotten zu aggregieren.
- Richtlinie zum Schutz vor Spieleskalierung — Aktuelle Einstellung für den Schutz der Spielesitzungen für die Flotte.
- Maximale Spielsitzungen pro Spieler — Die maximale Anzahl an Spielsitzungen, die ein Spieler während des Richtlinienzeitraums erstellen kann.
- Richtlinienzeitraum — Gibt an, wie lange man warten muss, bis die Anzahl der Sessions, die ein Spieler erstellt hat, zurückgesetzt wird.

Einzelheiten zum Bau

Im Abschnitt Build-Details wird der Build angezeigt, der auf der Flotte gehostet wird. Wählen Sie den Build-Namen aus, um die vollständige Build-Detailseite anzuzeigen.

Laufzeitkonfiguration

Im Abschnitt Laufzeitkonfiguration werden die Serverprozesse angezeigt, die auf jeder Instance gestartet werden sollen. Dazu gehören der Pfad für die ausführbare Datei des Spiel-Servers sowie optionale Startparameter.

Aktivierung der Spielsitzung

Im Abschnitt Aktivierung der Spielsitzung wird die Anzahl der Serverprozesse angezeigt, die gleichzeitig gestartet werden, und wie lange es dauert, bis der Prozess aktiviert ist, bevor er beendet wird.

EC2-Porteinstellungen

Im Abschnitt Ports werden die Verbindungsberechtigungen der Flotte angezeigt, einschließlich der IP-Adressen und der Porteinstellungsbereiche.

Metriken

Die Registerkarte Metrics (Metriken) zeigt eine grafische Darstellung der Flottenmetriken im Zeitverlauf. Weitere Informationen zur Verwendung von Metriken in Amazon GameLift finden Sie unterÜberwachen Sie Amazon GameLift mit Amazon CloudWatch.

Ereignisse

Die Registerkarte Events bietet ein Protokoll aller Ereignisse für die Flotte, einschließlich des Ereigniscodes, der Nachricht und des Zeitstempels. Weitere Informationen finden Sie in der GameLift Amazon-API-Referenz.

Skalierung

Die Registerkarte Skalierung enthält Informationen zur Flottenkapazität, einschließlich des aktuellen Status und der Kapazitätsänderungen im Laufe der Zeit. Dazu kommen Tools für die Aktualisierung der Kapazitätslimits und zur Verwaltung der automatischen Skalierung

Kapazität skalieren

Sehen Sie sich die aktuellen Flottenkapazitätseinstellungen für jeden Flottenstandort an. Weitere Informationen zum Ändern von Grenzwerten und Kapazitäten finden Sie unterSkalierung der GameLift Amazon-Hosting-Kapazität.

- AWSStandort — Name eines Standorts, an dem Flotteninstanzen bereitgestellt werden.
- Status — Hosting-Status des Flottenstandorts. Der Standortstatus muss `ACTIVE` lauten, um Spiele veranstalten zu können.
- Mindestgröße — Die kleinste Anzahl von Instanzen, die am Standort bereitgestellt werden müssen.
- Gewünschte Instanzen — Die Zielanzahl der aktiven Instanzen zur Aufrechterhaltung des Standorts. Wenn aktive Instanzen und gewünschte Instanzen nicht identisch sind, wird ein Skalierungsereignis gestartet, um Instanzen nach Bedarf zu starten oder herunterzufahren, bis aktive Instanzen den gewünschten Instanzen entsprechen.
- Max. Größe — Die meisten Instanzen, die am Standort bereitgestellt werden können.
- Verfügbar — Das Service-Limit für Instanzen abzüglich der Anzahl der verwendeten Instances. Dieser Wert gibt die maximale Anzahl von Instanzen an, die Sie dem Standort hinzufügen können.

Richtlinien zur automatischen Skalierung

In diesem Abschnitt finden Sie Informationen zu den Richtlinien für die automatische Skalierung, die auf die Flotte angewendet werden. Sie können eine zielorientierte Richtlinie einrichten oder aktualisieren. Die regelbasierten Richtlinien der Flotte, die mithilfe des AWS SDK oder der CLI definiert werden müssen, werden hier angezeigt. Weitere Hinweise zur Skalierung finden Sie unterAutomatische Skalierung der Flottenkapazität mit Amazon GameLift.

Verlauf skalieren

Sehen Sie sich Diagramme der Kapazitätsänderungen im Laufe der Zeit an.

Locations

Die Registerkarte Standorte listet alle Standorte auf, an denen Flotteninstanzen bereitgestellt werden. Zu den Standorten gehören die Heimatregion der Flotte und alle hinzugefügten entfernten Standorte. Auf dieser Registerkarte können Sie Standorte direkt hinzufügen oder entfernen.

- Standort — Name eines Standorts, an dem Flotteninstanzen bereitgestellt werden.
- Status — Hosting-Status des Flottenstandorts. Der Standortstatus verfolgt den Prozess der Aktivierung der ersten Instanzen am Standort. Der Standortstatus muss `ACTIVE` lauten, um Spiele veranstalten zu können.
- Aktive Instances — Die Anzahl der Instances mit Serverprozessen, die am Flottenstandort ausgeführt werden.
- Aktive Server — Die Anzahl der Gameserverprozesse, die Spielesitzungen am Flottenstandort hosten können.
- Spielsitzungen — Die Anzahl der aktiven Spielsitzungen auf Instanzen am Flottenstandort.
- Spielersitzungen — Die Anzahl der Spielersitzungen, die einzelne Spieler repräsentieren, die an Spielsitzungen teilnehmen, die am Flottenstandort aktiv sind.

Spielsitzungen

Die Registerkarte Game sessions führt aktuelle und frühere auf der Flotte gehostete Spielsitzungen zusammen mit einigen Detailinformationen auf. Wähle eine Spielsitzungs-ID, um auf zusätzliche Informationen zur Spielsitzung zuzugreifen, einschließlich Spielersitzungen. Weitere Informationen zu Spielersitzungen finden Sie unterDaten zu Spiel- und Spielersitzungen anzeigen.

Daten zu Spiel- und Spielersitzungen anzeigen

Sie können Informationen zu den Spielsitzungen und einzelnen Spielern einsehen. Weitere Informationen zu Spielsitzungen und Spielersitzungen finden Sie unter So verbinden sich Spieler mit Spielen.

Um die Spielsitzung und die Spielerdaten einzusehen

1. Wählen Sie in der GameLiftAmazon-Konsole im Navigationsbereich Flotten aus.
2. Wähle aus der Flottenliste die Flotte aus, die deine Spielsitzungen veranstaltet hat.
3. Wähle den Tab Spielsitzungen. In dieser Registerkarte werden alle auf der Flotte gehosteten Spielsitzungen zusammen mit zusammenfassenden Informationen angezeigt.
4. Wähle eine Spielsitzung aus, um zusätzliche Informationen zur Spielsitzung und eine Liste der Spieler zu sehen, die sich mit dem Spiel verbunden haben.

Details

Übersicht

In diesem Abschnitt finden Sie eine Zusammenfassung der Informationen zu Ihrer Spielsitzung.

- Status — Status der Spielsitzung.
  - Aktivierung — Die Instanz initiiert eine Spielsitzung.
  - Aktiv — Eine Spielsitzung läuft und steht Spielern zur Verfügung, je nach den Richtlinien der Sitzung zur Spielererstellung.
  - Abgebrochen — Die Spielsitzung ist beendet.
- ARN — Der Amazon-Ressourcenname der Spielsitzung.
- Name — Name, der für die Spielsitzung generiert wurde.
- Ort — Der Ort, an dem Amazon die Spielsitzung GameLift veranstaltet hat.
- Erstellungszeit — Datum und Uhrzeit, zu denen Amazon die Stream-Sitzung GameLift erstellt hat.
- Endzeit — Datum und Uhrzeit des Endes der Spielsitzung.
- DNS-Name — Der Hostname der Spielsitzung.
- IP-Adresse — Die für die Spielsitzung angegebene IP-Adresse.
- Port — Portnummer, die für die Verbindung mit der Spielsitzung verwendet wird.

- Creator-ID — Eine eindeutige Kennung des Spielers, der die Spielsitzung initiiert hat.
- Richtlinie zur Erstellung von Spielersitzungen — Gibt an, ob die Spielsitzung neue Spieler akzeptiert.
- Richtlinie zum Schutz vor Spieleskalierung — Die Art des Schutzes von Spielesitzungen, der auf allen neuen Instances eingerichtet werden soll, die Amazon in der Flotte GameLift startet.

Daten zum Spiel

Gut formatierte Daten, die Sie beim Start an Ihre Spielsitzung senden können.

Eigenschaften des Spiels

Eigenschaften von Schlüssel- und Wertpaaren, die deine Spielsitzung beeinflussen.

Matchmaking-Daten

Der FlexMatch Matchmaker JSON. Um den Matchmaker zu überprüfen und zu bearbeiten, wählen Sie Matchmaking-Konfiguration anzeigen. Weitere Informationen über FlexMatch Matchmaking findest du unter Einen Matchmaker erstellen.

Spielersitzungen

Es werden für jede Spiele-Sitzung die folgenden Daten zu Spielersitzungen erfasst:

- Spielersitzungs-ID — Die der Spielersitzung zugewiesene Kennung.
- Spieler-ID — Eine eindeutige Kennung für den Spieler. Wähle diese ID, um zusätzliche Spielerinformationen zu erhalten.
- Status — Der Status der Spielersitzung. Die folgenden Statuswerte sind möglich:
  - Reserviert — Die Spielersitzung wurde reserviert, aber die Spieler sind nicht verbunden.
  - Aktiv — Die Spielersitzung ist mit dem Spielserver verbunden.
  - Abgeschlossen — Die Spielersitzung wurde beendet; der Spieler ist nicht mehr verbunden.
  - Timeout — Der Spieler konnte keine Verbindung herstellen.
- Erstellungszeit — Die Zeit, zu der der Spieler eine Verbindung zur Spielsitzung hergestellt hat.
- Endzeit — Die Zeit, zu der der Spieler die Verbindung zur Spielsitzung getrennt hat.
- Spielerdaten — Informationen über den Spieler, die bei der Erstellung der Spielersitzung bereitgestellt wurden.

Spielerinformationen

Zeigen Sie zusätzliche Informationen für einen ausgewählten Spieler an, einschließlich einer flottenübergreifenden Liste aller Spiele, mit denen sich der Spieler in der aktuellen Region verbunden hat. Zu diesen Informationen gehören der Status, die Startzeiten, die Endzeiten und die gesamte Verbindungszeit für jede Spielersitzung. Du kannst wählen, ob du dir Daten für die jeweiligen Spielsitzungen und Flotten ansehen möchtest.

Sehen Sie sich Ihre Aliase an

Auf der Alias-Seite werden Informationen zu den Flottenaliasen angezeigt, die Sie in Ihrer aktuellen Region erstellt haben. Um die Alias-Seite aufzurufen, wählen Sie im Navigationsbereich Aliase aus.

Auf der Alias-Seite können Sie Folgendes tun:

- Erstellen Sie einen neuen Alias. Wählen Sie Alias erstellen aus.
- Filtern und sortieren Sie die Alias-Tabelle. Verwenden Sie die Steuerelemente oben in der Tabelle.
- Details zu Aliasnamen anzeigen. Wählen Sie einen Aliasnamen, um die Alias-Detailseite zu öffnen.
- Einen Alias löschen. Wählen Sie einen Alias und dann Löschen.

Angaben zum Alias

Auf der Alias-Detailseite werden Informationen über den Alias angezeigt.

Auf dieser Seite können Sie:

- Bearbeiten Sie einen Alias. Wählen Sie Edit (Bearbeiten) aus.
- Sehen Sie sich die Flotten an, die Sie dem Alias zugeordnet haben.
- Einen Alias löschen. Wählen Sie Löschen.

Die Detailinformationen zu Aliasnamen schließen Folgendes ein:

- ID — Die eindeutige Nummer, die zur Identifizierung des Alias verwendet wird.
- Beschreibung — Die Beschreibung des Alias.
- ARN — Der Amazon-Ressourcenname des Alias.
- Erstellung — Datum und Uhrzeit der Erstellung des Alias.

- Letzte Aktualisierung — Datum und Uhrzeit der letzten Aktualisierung des Alias.
- Routingtyp — Der Routing-Typ für den Alias. Dabei kann es sich um einen der folgenden Typen handeln:
  - Einfach — Leitet den Spielerverkehr an eine bestimmte Flotten-ID weiter. Sie können die Flotten-ID für einen Alias jederzeit aktualisieren.
  - Terminal — Leitet eine Nachricht an den Client zurück. Du kannst zum Beispiel Spieler, die einen out-of-date Client verwenden, an einen Ort weiterleiten, an dem sie ein Upgrade erhalten können.
- Tags — Schlüssel- und Wertepaare, die zur Identifizierung des Alias verwendet werden.

Sehen Sie sich Ihre Warteschlangen an

Sie können Informationen zu allen vorhandenen Warteschlangen zur Platzierung von Spielsitzungen anzeigen. Auf der Seite mit den Warteschlangen werden Warteschlangen angezeigt, die in Ihrer aktuellen Region erstellt wurden. Auf der Seite Warteschlangen können Sie eine neue Warteschlange erstellen, vorhandene Warteschlangen löschen oder eine Detailseite für eine ausgewählte Warteschlange öffnen. Jede Warteschlangendetailseite enthält die Konfigurations- und Metrikdaten der Warteschlange. Weitere Informationen zu Warteschlangen finden Sie unter GameLiftAmazon-Warteschlangen für die Platzierung von Spielesitzungen einrichten.

Auf der Seite mit den Warteschlangen werden für jede Warteschlange die folgenden zusammenfassenden Informationen angezeigt:

- Warteschlangenname — Der Name, der der Warteschlange zugewiesen wurde. Dieser Name wird beispielsweise verwendet, wenn Anforderungen auf eine neue Spielsitzung eine Warteschlange referenzieren.
- Warteschlangen-Timeout — Die maximale Zeitspanne in Sekunden, in der eine Anfrage zur Platzierung einer Spielsitzung in der Warteschlange verbleibt, bevor das Timeout abläuft.
- Ziele in der Warteschlange — Anzahl der Flotten, die in der Warteschlangenkonfiguration aufgeführt sind. Amazon GameLift platziert neue Spielsitzungen für jede Flotte in der Warteschlange.

Warteschlangendetails anzeigen

Sie können auf detaillierte Informationen zu einer Warteschlange zugreifen, z. B. die Konfiguration und die Metriken der Warteschlange. Um eine Warteschlangendetailseite zu öffnen, rufen Sie die Seite Warteschlangen auf und wählen Sie einen Warteschlangennamen.

Die Detailseite zu einer Warteschlange zeigt eine Übersichtstabelle und Registerkarten mit zusätzlichen Informationen an. Auf dieser Seite können Sie Folgendes machen:

- Sie können die Konfiguration der Warteschlange, die Liste der Ziele und Latenz-Richtlinien für Spieler aktualisieren. Wählen Sie Edit (Bearbeiten) aus.
- Löscht eine Warteschlange. Nachdem du eine Warteschlange gelöscht hast, schlagen alle Anfragen für neue Spielsitzungen fehl, die auf diesen Warteschlangennamen verweisen. Wählen Sie Löschen.

> **Note**
>
> Um eine gelöschte Warteschlange wiederherzustellen, erstellen Sie eine neue Warteschlange mit dem Namen der gelöschten Warteschlange.

Details

Übersicht

Im Abschnitt „Übersicht“ werden der Amazon-Ressourcenname (ARN) der Warteschlange und das Timeout angezeigt. Sie können den ARN verwenden, wenn Sie in anderen Aktionen oder Bereichen von Amazon GameLift auf die Warteschlange verweisen. Das Timeout ist die maximale Zeitspanne in Sekunden, für die eine Anfrage zur Platzierung einer Spielsitzung in der Warteschlange verbleibt, bevor das Timeout abläuft.

Ereignisbenachrichtigung

Im Abschnitt „Ereignisbenachrichtigung“ werden das SNS-Thema aufgeführt, für das Amazon Ereignisbenachrichtigungen GameLift veröffentlicht, sowie die Ereignisdaten, die zu allen Ereignissen hinzugefügt werden, die von dieser Warteschlange erstellt wurden.

Tags (Markierungen)

In der Tags-Tabelle werden die Schlüssel und Werte angezeigt, die zum Taggen der Ressource verwendet wurden. Weitere Informationen zum Taggen finden Sie unter Ressourcen zum Taggen AWS.

Metriken

Die Registerkarte Metrics zeigt eine grafische Darstellung der Metriken für die Warteschlange im Zeitverlauf.

Zu den Kennzahlen für die Warteschlangen gehören eine Reihe von Informationen zur Beschreibung der Platzierungsaktivitäten in der Warteschlange, einschließlich erfolgreicher Platzierungen, die nach Regionen organisiert wurden. Du kannst Regionsdaten verwenden, um zu verstehen, wo du deine Spiele hostest. Metriken zur regionalen Platzierung können dabei helfen, Probleme mit der allgemeinen Warteschlangengestaltung zu erkennen.

Warteschlangenmetriken sind auch in Amazon verfügbarCloudWatch. Eine Beschreibung der verfügbaren Metriken finden Sie unterGameLiftAmazon-Metriken für Warteschlangen.

Ziele

In der Registerkarte Destinations werden alle Ziele und Aliase für die Warteschlange angezeigt.

Wenn Amazon die Ziele GameLift nach verfügbaren Ressourcen für das Hosten einer neuen Spielsitzung durchsucht, durchsucht es die hier aufgeführte Standardreihenfolge. Solange am ersten aufgelisteten Ziel Kapazität vorhanden ist, GameLift platziert Amazon dort neue Spielesitzungen. Sie können veranlassen, dass individuelle Anfragen für Spielsitzungsplatzierungen die Standardreihenfolge überschreiben, indem Latenzdaten für den Spieler bereitgestellt werden. Diese Daten weisen Amazon anGameLift, nach einem verfügbaren Ziel mit der niedrigsten durchschnittlichen Spielerlatenz zu suchen. Weitere Informationen zum Entwerfen Ihrer Warteschlangen finden Sie unter. Entwerfen Sie eine Warteschlange für Spielsitzungen

Platzierung der Sitzung

Richtlinien für die Latenz von Spielern

Der Abschnitt Richtlinien für die Spielerlatenz enthält alle Richtlinien, die von der Warteschlange verwendet werden. In der Tabelle sind die Richtlinien in der Reihenfolge aufgeführt, in der sie durchgesetzt werden.

Locations

Im Abschnitt Standorte werden die Orte angezeigt, an denen diese Warteschlange eine Spielsitzung ablegen kann.

Priorität

Im Abschnitt „Priorität“ wird die Reihenfolge angezeigt, in der die Warteschlange die Details einer Spielsitzung bewertet.

Reihenfolge der Standorte

Der Abschnitt „Reihenfolge der Standorte“ zeigt die Standardreihenfolge, die die Warteschlange beim Platzieren von Spielsitzungen verwendet. Die Warteschlange verwendet diese Reihenfolge, wenn Sie keine anderen Prioritätstypen definiert haben.

Überwachung von Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon GameLift und Ihren anderen AWS Lösungen. Bei Amazon gibt es drei Hauptanwendungen für Metriken GameLift: zur Überwachung des Systemzustands und zur Einrichtung von Alarmen, zur Nachverfolgung der Leistung und Nutzung von Spieleservern und zur Kapazitätsverwaltung mithilfe manueller oder auto-scaling.

AWS bietet die folgenden Überwachungstools, um Amazon zu beobachten GameLift, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

- GameLift Amazon-Konsole
- Amazon CloudWatch — Sie können GameLift Amazon-Metriken in Echtzeit überwachen, ebenso wie Metriken für andere AWS Ressourcen und Anwendungen, die Sie auf AWS Services ausführen. CloudWatch bietet eine Reihe von Überwachungsfunktionen, darunter Tools zur Erstellung benutzerdefinierter Dashboards und die Möglichkeit, Alarme einzurichten, die benachrichtigen oder Maßnahmen ergreifen, wenn eine Metrik einen bestimmten Schwellenwert erreicht.
- AWS CloudTrail— erfasst alle API-Aufrufe und damit verbundene Ereignisse, die von oder im Namen Ihres AWS Kontos für Amazon GameLift und andere AWS Dienste getätigt wurden. Daten werden als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket übermittelt. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Anrufe erfolgten.
- Spielsitzungsprotokolle — Sie können benutzerdefinierte Servermeldungen für Ihre Spielsitzungen in Protokolldateien ausgeben, die in Amazon S3 gespeichert sind.

Themen

Überwachen Sie Amazon GameLift mit Amazon CloudWatch

Sie können Amazon GameLift mithilfe von Amazon überwachenCloudWatch, einem AWS Dienst, der Rohdaten sammelt und sie in lesbare Kennzahlen nahezu in Echtzeit verarbeitet. Diese Statistiken werden 15 Monate lang aufbewahrt, um einen historischen Überblick über die Leistung Ihres Gameservers zu bieten, der bei Amazon gehostet GameLift wird. Sie können auch Alarme einrichten, die auf bestimmte Grenzwerte prüfen und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im CloudWatchAmazon-Benutzerhandbuch.

In den folgenden Tabellen sind die Metriken und Dimensionen für Amazon aufgeführtGameLift. Alle Metriken, die in verfügbar CloudWatch sind, sind auch in der GameLift Amazon-Konsole verfügbar, die die Daten in Form von anpassbaren Grafiken bereitstellt. Für den Zugriff auf CloudWatch-Metriken für Ihre Spiele können Sie die AWS Management Console, die AWS CLI oder die CloudWatch-API verwenden.

Wenn eine Metrik keinen Standort hat, verwendet sie den Heimatstandort.

Dimensionen für GameLift Amazon-Metriken

Amazon GameLift unterstützt das Filtern von Metriken nach den folgenden Dimensionen.

Dimension	Beschreibung
`Location`	Filtern Sie Metriken für einen Flotteneinsatzort. Wenn eine Metrik keinen Standort hat, verwendet sie den Heimatstandort.
`FleetId`	Filtermetriken für eine einzelne Flotte. Diese Dimension kann für alle Instance-, Serverpro zess-, Spielsitzungs- und Spielersitzungsmetriken verwendet werden.
`MetricGroup`	Filtermetriken für eine Sammlung von Flotten. Fügen Sie einer Metrikgruppe eine Flotte hinzu, indem Sie den Namen der Metrikgruppe zu den Flottenat tributen hinzufügen UpdateFleetAttributes(siehe ()). Diese Dimension kann für alle Instance-, Serverpro

Dimension	Beschreibung
	zess-, Spielsitzungs- und Spielersitzungsmetriken verwendet werden.
`QueueName`	Filtermetriken für eine einzelne Warteschlange. Die Dimension wird nur für Metriken für Spielsitz ungswarteschlangen verwendet.
`ConfigurationName`	Filtermetriken für eine einzelne Matchmaking-Konfig uration. Diese Dimension wird für Metriken für Matchmaking-Konfigurationen verwendet.
`ConfigurationName-RuleName`	Filtermetriken für einen Schnittpunkt einer Matchmaki ng-Konfiguration und Matchmaking-Regel. Diese Dimension wird nur für Metriken für Matchmaking-Regeln verwendet.
`InstanceType`	Filtermetriken für eine EC2-Instance-Typ-Zuordnung (z. B. „c4.large2“). Diese Dimension wird für Spot-Inst ance-Metriken verwendet.
`OperatingSystem`	Filtermetriken für das Betriebssystem einer Instance. Diese Dimension wird mit Metriken für Spot-Inst ances verwendet.
`GameServerGroup`	Filtern Sie FleetIQ-Metriken für eine Spieleser vergruppe.

GameLiftAmazon-Metriken für Flotten

Der `AWS/GameLift`-Namespace enthält die folgenden Metriken hinsichtlich der Aktivitäten einer Flotte oder Flottengruppe. Flotten werden mit einer verwalteten GameLift Amazon-Lösung verwendet. Der GameLift Amazon-Service sendet Metriken zu CloudWatch jeder Minute.

Instances

Metrik	Beschreibung
`ActiveInstances`	Instances mit dem Status ACTIVE, d. h. solche, auf denen aktive Serverprozesse ausgeführt werden. Die Anzahl umfasst die nicht verwendeten Instances sowie jene, die mehr als eine Spielsitzung hosten. Diese Metrik misst die aktuelle Gesamtkapazität der Instance. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`DesiredInstances`	Zielanzahl der aktiven Instances, an GameLift deren Wartung Amazon in der Flotte arbeitet. Mit Auto Scaling wird dieser Wert anhand der aktuell gültigen Skalierungsrichtlinien bestimmt. Ohne Auto Scaling wird dieser Wert manuell festgelegt. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`IdleInstances`	Aktive Instances, die gegenwärtig null (0) Spielsitzungen hosten. Diese Metrik misst die verfügbare, aber nicht verwendete Kapazität. Diese Metrik kann mit Auto Scaling verwendet werden.

Metrik	Beschreibung
	Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`MaxInstances`	Maximale Anzahl der Instances, die für die Flotte zugelassen sind. Der Maximalwert einer Flotten-Instance bestimmt die Kapazitätsgrenze während der manuellen oder automatischen Aufwärtsskalierung. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`MinInstances`	Minimale Anzahl der Instances, die für die Flotte zugelassen sind. Der Minimalwert einer Flotten-Instance bestimmt die Kapazitätsgrenze während der manuellen oder automatischen Abwärtsskalierung. Diese Metrik ist nicht verfügbar, wenn Metrik-Daten für Flottengruppen angezeigt werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Beschreibung
`PercentIdleInstances`	Prozentsatz der ungenutzten aktiven Instances (berechnet als `IdleInstances / ActiveInstances`). Diese Metrik kann für Auto Scaling verwendet werden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`RecycledInstances`	Anzahl der Spot-Instances, die recycelt und ersetzt wurden. Amazon GameLift recycelt Spot-Instances, die derzeit keine Spielsitzungen hosten und bei denen die Wahrscheinlichkeit einer Unterbrechung hoch ist. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
`InstanceInterruptions`	Anzahl der Spot-Instances, die unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Beschreibung
`CPUUtilization`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Der Prozentsatz der physischen CPU-Zeit, die Amazon EC2 für die Ausführung der Instance verwendet, einschließlich der Zeit, die für die Ausführung des Benutzercodes und des Amazon EC2-Codes aufgewendet wird. Tools in Ihrem Betriebssystem können CloudWatc h aufgrund von Faktoren wie der Simulation älterer Geräte, der Konfiguration nicht veralteter Geräte, unterbrechungsintensiven Workloads, Live-Migr ation und Live-Update einen anderen Prozentsatz anzeigen als dies der Fall ist. Einheiten: Prozent
`NetworkIn`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Anzahl der von der Instance auf allen Netzwerkschnittstellen empfangenen Byte. Diese Metrik gibt das an eine Anwendung auf einer einzelnen Instance eingehende Netzwerkdatenvolum en an. Einheiten: Byte
`NetworkOut`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Anzahl der von der Instance auf allen Netzwerkschnittstellen gesendeten Byte. Diese Metrik gibt das an eine Anwendung auf einer einzelnen Instance ausgehende Netzwerkd atenvolumen an. Einheiten: Byte

Metrik	Beschreibung
`DiskReadBytes`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes gelesene Byte. Diese Metrik wird verwendet, um das von der Festplatte der Instance gelesene Datenvolu men der Anwendung zu ermitteln. Sie können damit die Geschwindigkeit der Anwendung bestimmen. Einheiten: Byte
`DiskWriteBytes`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. In alle der Instance zu Verfügung stehenden Instance-Speicher-Volumes geschriebene Byte. Diese Metrik wird verwendet, um das auf die Festplatte der Instance geschrieb ene Datenvolumen der Anwendung zu ermitteln. Sie können damit die Geschwindigkeit der Anwendung bestimmen. Einheiten: Byte
`DiskReadOps`	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Abgeschlossene Lesevorgä nge von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes in einem angegebenen Zeitraum. Zum Berechnen der durchschnittlichen I/O-Operationen pro Sekunde (IOPS) für einen Zeitraum dividieren Sie die Gesamtvorgänge im Zeitraum durch die Anzahl der Sekunden in dem Zeitraum. Einheiten: Anzahl

Metrik	Beschreibung
DiskWriteOps	EC2-Metrik. Für Amazon steht GameLift diese Metrik für die Hardwareleistung aller aktiven Instances an einem Flottenstandort. Abgeschlossene Schreibvorgänge von allen der Instance zu Verfügung stehenden Instance-Speicher-Volumes in einem angegebenen Zeitraum. Zum Berechnen der durchschnittlichen I/O operationen pro Sekunde (IOPS) für einen Zeitraum dividieren Sie die Gesamtvorgänge im Zeitraum durch die Anzahl der Sekunden in dem Zeitraum. Einheiten: Anzahl

Serverprozesse

Metrik	Beschreibung
ActiveServerProcesses	Serverprozesse mit dem Status ACTIVE, d. h. solche, die laufen und Spielsitzungen hosten können. Die Anzahl umfasst die ungenutzten Serverprozesse und jene, die Spielsitzungen hosten. Diese Metrik misst die aktuelle Gesamt-Serverprozesskapazität. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
HealthyServerProcesses	Aktive stabile Serverprozesse. Diese Metrik ist beim Nachverfolgen des übergreifenden Zustands der Spielserver der Flotte nützlich. Einheiten: Anzahl

Metrik	Beschreibung
	Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`PercentHealthyServerProcesses`	Prozentsatz aller aktiven stabilen Serverprozesse (berechnet als `HealthyServerProcesses / ActiveServerProcesses`). Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`ServerProcessAbnormalTerminations`	Serverprozesse, die seit dem letzten Bericht infolge anormaler Umstände heruntergefahren wurden. Diese Metrik beinhaltet Kündigungen, die vom GameLift Amazon-Service initiiert wurden. Dies tritt auf, wenn ein Serverprozess nicht mehr reagiert, konsistent fehlgeschlagene Integritätsprüfungen meldet oder nicht ordnungsgemäß beendet wird (durch den Aufruf von ProcessEnding()). Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Beschreibung
`ServerProcessActivations`	Serverprozesse, die seit dem letzten Bericht erfolgrei ch vom Status ACTIVATING in den Status ACTIVE gewechselt sind. Um Spielsitzungen hosten zu können, müssen die Serverprozesse aktiv sein. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort
`ServerProcessTerminations`	Serverprozesse, die seit dem letzten Bericht heruntergefahren wurden. Dies umfasst alle Serverprozesse, die aus irgendeinem Grund in den Status TERMINATED gewechselt sind, einschlie ßlich ordnungsgemäße und nicht ordnungsgemäße Prozessbeendigungen. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Spielsitzungen

Metrik	Beschreibung
`ActivatingGameSessions`	Spielsitzungen mit dem Status ACTIVATING, d. h. solche, die sich im Hochfahrprozess befinden. Spielsitzungen können Spieler erst dann hosten, wenn sie aktiv sind. Über einen längeren Zeitraum anliegende hohe Werte weisen darauf hin, dass die Spielsitzungen nicht vom Status ACTIVATING in den

Metrik	Beschreibung
	Status ACTIVE wechseln. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`ActiveGameSessions`	Spielsitzungen mit dem Status ACTIVE, d. h. solche, die Spieler hosten können und null oder mehr Spieler hosten. Diese Metrik misst die Gesamtzahl der Spielsitzungen, die gegenwärtig gehostet werden. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Beschreibung
`AvailableGameSessions`	Aktive, fehlerfreie Serverprozesse, die derzeit nicht zum Hosten einer Spielsitzung verwendet werden und ohne Verzögerung eine neue Spielsitzung starten können, um neue Serverprozesse oder -instanzen hochzufahren. Diese Metrik kann mit Auto Scaling verwendet werden. **Note** Verwenden Sie für Flotten, die die Aktivierung gleichzeitiger Spielsitzungen einschränken, die Metrik. `ConcurrentActivatableGameSessions` Diese Metrik gibt die Anzahl der neuen Spielsitzungen genauer an, die ohne jegliche Verzögerung beginnen können. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort

Metrik	Beschreibung
`ConcurrentActivatableGameSessions`	Aktive, fehlerfreie Serverprozesse, die derzeit nicht zum Hosten einer Spielsitzung verwendet werden und sofort eine neue Spielsitzung starten können. Diese Metrik unterscheidet `AvailableGameSessions` sich wie folgt: Serverprozesse, die derzeit aufgrund von Beschränkungen bei der Aktivierung von Spielsitzungen keine neue Spielsitzung aktivieren können, werden nicht mitgezählt. (Siehe RuntimeConfigurationoptionale Flotteneinstellung `MaxConcurrentGameSessionActivations`). Für Flotten, die die Aktivierungen von Spielsitzungen nicht einschränken, ist diese Metrik identisch mit. `AvailableGameSessions` Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Abmessungen: Standort
`PercentAvailableGameSessions`	Spielsitzungs-Slots auf allen aktiven (stabilen oder unstabilen) Servern, die gegenwärtig nicht verwendet werden (berechnet als `AvailableGameSessions / [ActiveGameSessions + AvailableGameSessions + unhealthy server processes]`). Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt Abmessungen: Standort

Metrik	Beschreibung
`GameSessionInterruptions`	Anzahl der Spielsitzungen auf Spot-Instances, die unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum Abmessungen: Standort

Spielersitzungen

Metrik	Beschreibung
`CurrentPlayerSessions`	Spielersitzungen, die sich im Status ACTIVE (der Spieler ist mit einer aktiven Spielsitzung verbunden) oder RESERVED (dem Spieler ist ein Slot in einer Spielsitzung zugewiesen worden, hat sich aber noch nicht verbunden) befinden. Diese Metrik kann mit Auto Scaling verwendet werden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum
`PlayerSessionActivations`	Spielersitzungen, die seit dem letzten Bericht vom Status RESERVED in den Status ACTIVE gewechsel t sind. Dies tritt ein, wenn sich ein Spieler erfolgreich mit einer aktiven Spielsitzung verbindet. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe, Durchschnitt, Minimum, Maximum

GameLiftAmazon-Metriken für Warteschlangen

Der `Amazon GameLift`-Namespace enthält die folgenden Metriken hinsichtlich der Aktivitäten in einer Spielsitzungsplatzierungs-Warteschlange. Warteschlangen werden mit einer verwalteten GameLift Amazon-Lösung verwendet. Der GameLift Amazon-Service sendet Metriken zu CloudWatch jeder Minute.

Metrik	Beschreibung
`AverageWaitTime`	Durchschnittliche Zeit, die sich die Anforderungen zur Spielsitzungsplatzierung in der Warteschlange mit dem Status PENDING befunden haben, bis sie erfüllt worden sind. Einheiten: Sekunden Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe Abmessungen: Standort
`FirstChoiceNotViable`	Spielesitzungen, die erfolgreich platziert wurden, jedoch NICHT in der Flotte der ersten Wahl, da diese Flotte als nicht geeignet betrachtet wurde (da es sich beispielsweise um eine Spot-Flotte mit einer hohen Unterbrechungsrate handelt). Diese Metrik basiert auf den Kosten, nicht auf der Latenz. Die Flotte der ersten Wahl ist entweder die erste Flotte, die in der Warteschlange aufgeführt ist, oder — wenn eine Platzierungsanfrage Daten zur Spielerlatenz enthält — die erste Flotte, die durch die FleetIQ-Priorisierung ausgewählt wurde. Wenn es keine realisierbare Spot-Flotten gibt, kann eine beliebige Flotte in dieser Region ausgewählt werden. Einheiten: Anzahl

Metrik	Beschreibung
	Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
`FirstChoiceOutOfCapacity`	Spielesitzungen, die erfolgreich platziert wurden, jedoch NICHT in der Flotte der ersten Wahl, da in dieser Flotte keine Ressourcen verfügbar waren. Die Flotte der ersten Wahl ist entweder die erste Flotte, die in der Warteschlange aufgeführt ist, oder — wenn eine Platzierungsanfrage Daten zur Spielerlatenz enthält — die erste Flotte, die anhand Ihrer definiert en FleetIQ-Priorisierung ausgewählt wurde. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
`LowestLatencyPlacement`	Spielesitzungen, die erfolgreich in einer Region platziert wurden, die den Spielern die niedrigste Latenz in der Warteschlange bietet, die möglich ist. Diese Metrik wird nur ausgegeben, wenn in der Platzierungsanforderung Spielerlatenzdaten enthalten sind. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Beschreibung
`LowestPricePlacement`	Spielsitzungen, die erfolgreich in einer Flotte platziert wurden, deren Preis in der Warteschlange für die gewählte Region so niedrig wie möglich war. Diese Flotte kann entweder eine Spot-Flotte oder eine On-Demand-Instance sein, wenn die Warteschlange keine Spot-Instances enthält. Diese Metrik wird nur ausgegeben, wenn in der Platzierungsanforderung Spielerlatenzdaten enthalten sind. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
`Placement <region name>`	Spielesitzungen, die erfolgreich in Flotten platziert wurden, die sich in der angegebenen Region befinden. Diese Metrik analysiert die `PlacementsSucceeded` -Metrik nach Region. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`PlacementsCanceled`	Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht vor Erreichen des Zeitlimits abgebrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Beschreibung
PlacementsFailed	Nicht erfolgreiche Anforderungen für Spielesit zungsplatzierungen seit dem letzten Bericht. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsStarted	Neue Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht der Warteschlange hinzugefügt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsSucceeded	Neue Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht zu einer neuen Spielsitz ung geführt haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
PlacementsTimedOut	Anforderungen für Spielsitzungsplatzierungen, die seit dem letzten Bericht das Zeitlimit der Warteschl ange erreicht haben, ohne erfüllt worden zu sein. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe

Metrik	Beschreibung
`QueueDepth`	Anzahl der Anforderungen für Spielsitzungsplatz ierungen in der Warteschlange mit dem Status PENDING. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe Abmessungen: Standort

GameLiftAmazon-Metriken für Matchmaking

Der `Amazon GameLift` Namespace enthält FlexMatch Aktivitätsmetriken für Matchmaking-Konfigurationen und Matchmaking-Regeln. FlexMatchMatchmaking wird mit einer verwalteten GameLift Amazon-Lösung verwendet. Der GameLift Amazon-Service sendet Metriken zu CloudWatch jeder Minute.

Weitere Informationen zur Reihenfolge der Matchmaking-Aktivitäten finden Sie unter So GameLift FlexMatch funktioniert Amazon.

Matchmaking-Konfigurationen

Metrik	Beschreibung
`CurrentTickets`	Aktuell verarbeitete oder auf die Verarbeitung wartende Matchmaking-Anforderungen. Einheiten: Anzahl Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum, Summe
`MatchAcceptancesTimedOut`	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die potenziellen Matches, für die seit dem letzten Bericht während des Akzeptier ungsvorgangs ein Timeout aufgetreten ist.

Metrik	Beschreibung
	Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`MatchesAccepted`	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die seit dem letzten Bericht akzeptierten potenziellen Matches. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`MatchesCreated`	Potenzielle Matches, die seit dem letzten Bericht erstellt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`MatchesPlaced`	Matches, die seit dem letzten Bericht in einer Spielsitzung platziert wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`MatchesRejected`	Für Matchmaking-Konfigurationen, die eine Akzeptanz bedingen, die potenziellen Matches, für die seit dem letzten Bericht von mindestens einem Spieler eine Ablehnung erteilt wurde. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe

Metrik	Beschreibung
PlayersStarted	Spieler in Matchmaking-Tickets, die seit dem letzten Bericht hinzugefügt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsFailed	Matchmaking-Anforderungen, die seit dem letzten Bericht zu einem Fehler geführt haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsStarted	Neue Matchmaking-Anforderungen, die seit dem letzten Bericht erstellt wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TicketsTimedOut	Matchmaking-Anforderungen, die seit dem letzten Bericht das Timeout-Limit erreicht haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
TimeToMatch	Für Matchmaking-Anforderungen, die vor dem letzten Bericht in einem potenziellen Match angeordne t wurden, die Dauer zwischen der Erstellung des Tickets und des potenziellen Matchs. Einheiten: Sekunden Relevante CloudWatch Statistiken: Stichproben, Durchschnitt, Minimum, Maximum

Metrik	Beschreibung
`TimeToTicketCancel`	Für Matchmaking-Anforderungen, die vor dem letzten Bericht storniert wurden, die Zeit zwischen der Ticketerstellung und der Stornierung. Einheiten: Sekunden Relevante CloudWatch Statistiken: Stichproben, Durchschnitt, Minimum, Maximum
`TimeToTicketSuccess`	Für Matchmaking-Anforderungen, die vor dem letzten Bericht erfolgreich waren, die Zeit zwischen der Ticketerstellung und der erfolgreichen Match-Pla tzierung. Einheiten: Sekunden Relevante CloudWatch Statistiken: Stichproben, Durchschnitt, Minimum, Maximum

Matchmaking-Regeln

Metrik	Beschreibung
`RuleEvaluationsPassed`	Regelauswertungen während des Matchmaking-Prozesses, die seit dem letzten Bericht erfolgreich waren. Diese Metrik ist auf die obersten 50 Regeln begrenzt. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe
`RuleEvaluationsFailed`	Regelauswertungen während des Matchmakings, die seit dem letzten Bericht fehlgeschlagen sind. Diese Metrik ist auf die obersten 50 Regeln begrenzt. Einheiten: Anzahl

Metrik	Beschreibung
	Relevante CloudWatch Statistiken: Summe

GameLiftAmazon-Metriken für FleetIQ

Der `Amazon GameLift` Namespace enthält Metriken für die FleetIQ-Spieleservergruppe und die Gameserveraktivität als Teil einer FleetIQ-Standalone-Lösung für das Game-Hosting. Der GameLift Amazon-Service sendet Metriken zu CloudWatch jeder Minute. Weitere Informationen finden Sie CloudWatch im Amazon EC2 Auto Scaling-Benutzerhandbuch unter Überwachen Ihrer Auto Scaling-Gruppen und -Instances mithilfe von Amazon.

Metrik	Beschreibung
`AvailableGameServers`	Spiel-Server, die für Spielausführungen verfügbar sind und derzeit nicht mit dem Gameplay beschäfti gt sind. Diese Zahl enthält Spiel-Server, die beansprucht wurden, aber immer noch den Status VERFÜGBAR haben. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Maße: GameServerGroup
`UtilizedGameServers`	Spielserver, die derzeit mit Gameplay beschäftigt sind. Diese Zahl schließt Gameserver ein, die sich im Status UTILIZED befinden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Maße: GameServerGroup
`DrainingAvailableGameServers`	Spielserver auf Instances, die für die Beendigung geplant sind und die derzeit kein Gameplay unterstüt zen. Diese Spielserver haben die niedrigste Priorität

Metrik	Beschreibung
	, die als Reaktion auf eine neue Anspruchsanforderung beansprucht wird. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Maße: GameServerGroup
`DrainingUtilizedGameServers`	Spiel-Server auf Instances, die zur Beendigung geplant sind und die derzeit das Gameplay unterstützen. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Maße: GameServerGroup
`PercentUtilizedGameServers`	Teil der Spiel-Server, die derzeit Spielausführungen unterstützen. Diese Metrik gibt die Größe der Spiel-Server-Kapazität an, die derzeit verwendet wird. Sie ist nützlich, um eine Auto Scaling-Richtlinie zu steuern, die Instances dynamisch hinzufügen und entfernen kann, um den Anforderungen des Spielers gerecht zu werden. Einheiten: Prozent Relevante CloudWatch Statistiken: Durchschnitt, Minimum, Maximum Maße: GameServerGroup

Metrik	Beschreibung
`GameServerInterruptions`	Spielserver auf Spot-Instances, die aufgrund der begrenzten Spot-Verfügbarkeit unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Abmessungen:GameServerGroup, InstanceType
`InstanceInterruptions`	Spot-Instances, die aufgrund begrenzter Verfügbar keit unterbrochen wurden. Einheiten: Anzahl Relevante CloudWatch Statistiken: Summe Abmessungen:GameServerGroup, InstanceType

Protokollieren von GameLift Amazon-API-Aufrufen mit AWS CloudTrail

Amazon GameLift ist in einen Dienst integriertAWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in Amazon ausgeführt wurdenGameLift. CloudTrailerfasst alle API-Aufrufe für Amazon GameLift als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der GameLift Amazon-Konsole und Codeaufrufe an die GameLift Amazon-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3-Bucket aktivieren, einschließlich Ereignissen für AmazonGameLift. Auch wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole in Event history (Ereignisverlauf) anzeigen. Anhand der von gesammelten Informationen können Sie ermittelnCloudTrail, welche Anfrage an Amazon gestellt wurdeGameLift, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Informationen.

Weitere Informationen zu CloudTrail finden Sie im AWS CloudTrail-Benutzerhandbuch.

GameLiftAmazon-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres Kontos auf AWS-Konto aktiviert. Wenn in Amazon eine Aktivität stattfindetGameLift, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Eventhistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr(em) AWS-Konto anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-Ereignisverlauf.

Erstellen Sie einen Trail, um eine fortlaufende Aufzeichnung der Ereignisse in Ihrem UnternehmenAWS-Konto, einschließlich Veranstaltungen für AmazonGameLift, zu erhalten. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien an einen Amazon S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- In CloudTrail unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien aus mehreren Konten

Alle GameLift Amazon-Aktionen werden von der GameLiftAmazon-API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe `CreatePlayerSession` und `UpdateGameSession` Aktionen Einträge in den CloudTrail Protokolldateien. `CreateGameSession`

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Anmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter CloudTrail-Element userIdentity.

Grundlegendes zu GameLift Amazon-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail-Protokolleintrag, der die Aktionen `CreateFleet` und `DescribeFleetAttributes` demonstriert.

```
{
    "Records": [
        {
            "eventVersion": "1.04",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:user/myUserName",
                "accountId": "111122223333",
                "accessKeyId": AKIAIOSFODNN7EXAMPLE",
                "userName": "myUserName"
            },
            "eventTime": "2015-12-29T23:40:15Z",
            "eventSource": "gamelift.amazonaws.com",
            "eventName": "CreateFleet",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "[]",
            "requestParameters": {
                "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d",
                "eC2InboundPermissions": [
                    {
                        "ipRange": "10.24.34.0/23",
                        "fromPort": 1935,
                        "protocol": "TCP",
                        "toPort": 1935
                    }
                ],
```

```
                "logPaths": [
                    "C:\\game\\serverErr.log",
                    "C:\\game\\serverOut.log"
                ],
                "eC2InstanceType": "c5.large",
                "serverLaunchPath": "C:\\game\\MyServer.exe",
                "description": "Test fleet",
                "serverLaunchParameters": "-paramX=baz",
                "name": "My_Test_Server_Fleet"
            },
            "responseElements": {
                "fleetAttributes": {
                    "fleetId": "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e",
                    "serverLaunchPath": "C:\\game\\MyServer.exe",
                    "status": "NEW",
                    "logPaths": [
                        "C:\\game\\serverErr.log",
                        "C:\\game\\serverOut.log"
                    ],
                    "description": "Test fleet",
                    "serverLaunchParameters": "-paramX=baz",
                    "creationTime": "Dec 29, 2015 11:40:14 PM",
                    "name": "My_Test_Server_Fleet",
                    "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d"
                }
            },
            "requestID": "824a2a4b-ae85-11e5-a8d6-61d5cafb25f2",
            "eventID": "c8fbea01-fbf9-4c4e-a0fe-ad7dc205ce11",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        },
        {
            "eventVersion": "1.04",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:user/myUserName",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "myUserName"
            },
            "eventTime": "2015-12-29T23:40:15Z",
            "eventSource": "gamelift.amazonaws.com",
            "eventName": "DescribeFleetAttributes",
```

```
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "[]",
            "requestParameters": {
                "fleetIds": [
                    "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e"
                ]
            },
            "responseElements": null,
            "requestID": "82e7f0ec-ae85-11e5-a8d6-61d5cafb25f2",
            "eventID": "11daabcb-0094-49f2-8b3d-3a63c8bad86f",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        },
    ]
}
```

Protokollieren von Servernachrichten in Amazon GameLift

Sie können benutzerdefinierte Servermeldungen von Ihren GameLift Amazon-Servern in Protokolldateien erfassen. Die Art und Weise, wie Sie die Protokollierung konfigurieren, hängt davon ab, ob Sie benutzerdefinierte Server oder Echtzeitserver verwenden (siehe die entsprechenden Unterabschnitte in diesem Kapitel).

Themen

- Protokollieren von Servernachrichten (benutzerdefinierte Server)
- Protokollieren von Servernachrichten (Echtzeitserver)

Protokollieren von Servernachrichten (benutzerdefinierte Server)

Sie können benutzerdefinierte Servernachrichten von Ihren GameLift benutzerdefinierten Amazon-Servern in Protokolldateien erfassen. Weitere Informationen zur Protokollierung für Echtzeitserver finden Sie unter Protokollieren von Servernachrichten (Echtzeitserver).

> **Important**
>
> Die Größe einer Protokolldatei pro Spielsitzung ist begrenzt (siehe Amazon- GameLift Endpunkte und -Kontingente im Allgemeine AWS-Referenz). Wenn eine Spielsitzung endet, lädt Amazon die Serverprotokolle in Amazon Simple Storage Service (Amazon S3) GameLift

hoch. Amazon GameLift lädt keine Protokolle hoch, die das Limit überschreiten. Protokolle können sehr schnell wachsen und die Größenbeschränkung überschreiten. Sie sollten Ihre Protokolle überwachen und die Protokollausgabe auf die erforderlichen Nachrichten beschränken.

Konfigurieren der Protokollierung für benutzerdefinierte Server

Bei GameLift benutzerdefinierten Amazon-Servern schreiben Sie Ihren eigenen Code, um die Protokollierung durchzuführen, die Sie als Teil Ihrer Serverprozesskonfiguration konfigurieren. Amazon GameLift verwendet Ihre Protokollierungskonfiguration, um die Dateien zu identifizieren, die am Ende jeder Spielsitzung in Amazon S3 hochgeladen werden müssen.

Die folgenden Anweisungen zeigen, wie Sie die Protokollierung anhand vereinfachter Codebeispiele konfigurieren:

C++

So konfigurieren Sie die Protokollierung (C++)

1. Erstellen Sie einen Vektor von Zeichenfolgen, die Verzeichnispfade zu Spielserver-Protokolldateien sind.

```
std::string serverLog("serverOut.log");        // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. Geben Sie Ihren Vektor als die Ihres ProcessParameters LogParametersObjekts an.

```
Aws::GameLift::Server::ProcessParameters processReadyParameter =
 Aws::GameLift::Server::ProcessParameters(
    std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
    std::bind(&Server::onProcessTerminate, this),
    std::bind(&Server::OnHealthCheck, this),
    std::bind(&Server::OnUpdateGameSession, this),
    listenPort,
    Aws::GameLift::Server::LogParameters(logPaths));
```

3. Geben Sie das ProcessParameters Objekt an, wenn Sie ProcessReady() aufrufen.

```
Aws::GameLift::GenericOutcome outcome =
```

```
   Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

Ein vollständigeres Beispiel finden Sie unter ProcessReady().

C#

So konfigurieren Sie die Protokollierung (C#)

1. Erstellen Sie eine Liste von Zeichenfolgen, die Verzeichnispfade zu Spielserver-Protokolldateien sind.

```
List<string> logPaths = new List<string>();
logPaths.Add("C:\\game\\serverOut.txt");     // Example of a log file that the
 game server writes
```

2. Geben Sie Ihre Liste als die Ihres ProcessParameters LogParametersObjekts an.

```
var processReadyParameter = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(logPaths));
```

3. Geben Sie das ProcessParameters Objekt an, wenn Sie ProcessReady() aufrufen.

```
var processReadyOutcome =
   GameLiftServerAPI.ProcessReady(processReadyParameter);
```

Ein vollständigeres Beispiel finden Sie unter ProcessReady().

Schreiben in Protokolle

Ihre Protokolldateien existieren, nachdem Ihr Serverprozess gestartet wurde. Sie können mit einer beliebigen Methode in die Protokolle schreiben, um in Dateien zu schreiben. Um die gesamte Standardausgabe und Fehlerausgabe Ihres Servers zu erfassen, ordnen Sie die Ausgabestreams den Protokolldateien zu, wie in den folgenden Beispielen:

C++

```
std::freopen("serverOut.log", "w+", stdout);
std::freopen("serverErr.log", "w+", stderr);
```

C#

```
Console.SetOut(new StreamWriter("serverOut.txt"));
Console.SetError(new StreamWriter("serverErr.txt"));
```

Zugreifen auf Serverprotokolle

Wenn eine Spielsitzung endet, speichert Amazon die Protokolle GameLift automatisch in einem Amazon S3-Bucket und speichert sie 14 Tage lang. Um den Speicherort der Protokolle für eine Spielsitzung abzurufen, können Sie den GetGameSessionLogUrl-API-Vorgang verwenden. Um die Protokolle herunterzuladen, verwenden Sie die URL, die der Vorgang zurückgibt.

Protokollieren von Servernachrichten (Echtzeitserver)

Sie können benutzerdefinierte Servernachrichten von Ihren Echtzeitservern in Protokolldateien erfassen. Weitere Informationen zur Protokollierung für benutzerdefinierte Server finden Sie unter Protokollieren von Servernachrichten (benutzerdefinierte Server).

Es gibt verschiedene Arten von Nachrichten, die Sie in Ihren Protokolldateien speichern können (siehe Protokollieren von Nachrichten in Ihrem Serverskript). Zusätzlich zu Ihren benutzerdefinierten Nachrichten geben Ihre Echtzeitserver Systemnachrichten mit denselben Nachrichtentypen aus und schreiben in dieselben Protokolldateien. Sie können die Protokollierungsstufe für Ihre Flotte anpassen, um die Menge der von Ihren Servern generierten Protokollierungsnachrichten zu reduzieren (siehe Anpassen der Protokollierungsebene).

> **Important**
>
> Die Größe einer Protokolldatei pro Spielsitzung ist begrenzt (siehe Amazon- GameLift Endpunkte und -Kontingente im Allgemeine AWS-Referenz). Wenn eine Spielsitzung endet, lädt Amazon die Serverprotokolle in Amazon Simple Storage Service (Amazon S3) GameLift hoch. Amazon GameLift lädt keine Protokolle hoch, die das Limit überschreiten. Protokolle können sehr schnell wachsen und die Größenbeschränkung überschreiten. Sie sollten

Ihre Protokolle überwachen und die Protokollausgabe auf die erforderlichen Nachrichten beschränken.

Protokollieren von Nachrichten in Ihrem Serverskript

Sie können benutzerdefinierte Nachrichten im Skript für Ihre Echtzeit-Server ausgeben. Gehen Sie wie folgt vor, um Servernachrichten an eine Protokolldatei zu senden:

1. Erstellen Sie einen Wert, der den Verweis auf das Logger-Objekt enthält.

```
var logger;
```

2. Rufen Sie in der `init()` Funktion den Logger aus dem Sitzungsobjekt ab und weisen Sie ihn Ihrer Logger-Variablen zu.

```
function init(rtSession) {
    session = rtSession;
    logger = session.getLogger();
}
```

3. Rufen Sie die entsprechende Funktion auf dem Logger auf, um eine Nachricht auszugeben.

   Debuggen von Nachrichten

```
logger.debug("This is my debug message...");
```

   Informative Nachrichten

```
logger.info("This is my info message...");
```

   Warnmeldungen

```
logger.warn("This is my warn message...");
```

   Fehlermeldungen

```
logger.error("This is my error message...");
```

Schwerwiegende Fehlermeldungen

```
logger.fatal("This is my fatal error message...");
```

Schwerwiegende Fehlermeldungen für Kunden

```
logger.cxfatal("This is my customer experience fatal error message...");
```

Ein Beispiel für die Protokollierungsanweisungen in einem Skript finden Sie unter Beispiel für ein Realtime Server-Skript.

Die Ausgabe in den Protokolldateien gibt den Nachrichtentyp (`DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, `CXFATAL`), wie in den folgenden Zeilen aus einem Beispielprotokoll dargestellt:

```
09 Sep 2021 11:46:32,970 [INFO] (gamelift.js) 215: Calling GameLiftServerAPI.InitSDK...
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 220: GameLiftServerAPI.InitSDK succeeded
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 223: Waiting for Realtime server to
 start...
09 Sep 2021 11:46:33,15 [WARN] (index.js) 204: Connection is INSECURE. Messages will be
 sent/received as plaintext.
```

Zugreifen auf Serverprotokolle

Wenn eine Spielsitzung endet, speichert Amazon die Protokolle GameLift automatisch in Amazon S3 und speichert sie 14 Tage lang. Sie können den GetGameSessionLogUrl API-Aufruf verwenden, um den Speicherort der Protokolle für eine Spielsitzung abzurufen. Verwenden Sie die vom API-Aufruf zurückgegebene URL, um die Protokolle herunterzuladen.

Anpassen der Protokollierungsebene

Protokolle können sehr schnell wachsen und die Größenbeschränkung überschreiten.
Sie sollten Ihre Protokolle überwachen und die Protokollausgabe auf die erforderlichen Nachrichten beschränken. Für Echtzeitserver können Sie die Protokollierungsebene anpassen, indem Sie in der Laufzeitkonfiguration Ihrer Flotte einen Parameter in der Form angeben`loggingLevel:`*`LOGGING_LEVEL`*, wobei einer der folgenden Werte *`LOGGING_LEVEL`* ist:

1. `debug`

2. `info` (Standard)
3. `warn`
4. `error`
5. `fatal`
6. `cxfatal`

Diese Liste ist von am wenigsten schwerwiegend (`debug`) bis am schwerwiegendsten (`cxfatal`). Sie legen einen einzelnen fest `loggingLevel` und der Server protokolliert Nachrichten nur mit diesem oder einem höheren Schweregrad. Wenn Sie beispielsweise festlegen, schreiben `loggingLevel:error` alle Server in Ihrer Flotte nur `fatal`-, `error`- und -`cxfatal`Nachrichten in das Protokoll.

Sie können die Protokollierungsstufe für Ihre Flotte festlegen, wenn Sie sie erstellen oder nachdem sie ausgeführt wird. Das Ändern der Protokollierungsebene Ihrer Flotte nach der Ausführung wirkt sich nur auf Protokolle für Spielsitzungen aus, die nach dem Update erstellt wurden. Protokolle für bestehende Spielsitzungen sind davon nicht betroffen. Wenn Sie beim Erstellen Ihrer Flotte keine Protokollierungsebene festlegen, legen Ihre Server die Protokollierungsebene `info` standardmäßig auf fest. Anweisungen zum Festlegen der Protokollierungsebene finden Sie in den folgenden Abschnitten.

Festlegen der Protokollierungsebene beim Erstellen einer Realtime Servers-Flotte (Konsole)

Folgen Sie den Anweisungen unter , Erstellen Sie eine von Amazon GameLift verwaltete Flotte um Ihre Flotte zu erstellen, mit der folgenden Ergänzung:

- Geben Sie im Unterschritt Serverprozesszuweisung des Schritts Prozessverwaltung das Schlüssel-Wert-Paar auf Protokollierungsebene (z. B. `loggingLevel:error`) als Wert für Startparameter an. Verwenden Sie ein nicht alphanumerisches Zeichen (außer Komma), um die Protokollierungsebene von allen zusätzlichen Parametern zu trennen (z. B. `loggingLevel:error +map Winter444`).

Festlegen der Protokollierungsebene beim Erstellen einer Realtime Servers-Flotte (AWS CLI)

Folgen Sie den Anweisungen unter , Erstellen Sie eine von Amazon GameLift verwaltete Flotte um Ihre Flotte zu erstellen, mit der folgenden Ergänzung:

- Geben Sie im Argument für den `--runtime-configuration` Parameter für `create-fleet`das Schlüssel-Wert-Paar der Protokollierungsebene (z. B. `loggingLevel:error`) als Wert für an`Parameters`. Verwenden Sie ein nicht alphanumerisches Zeichen (außer Komma), um die Protokollierungsstufe von allen zusätzlichen Parametern zu trennen. Sehen Sie sich das folgende Beispiel an:

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,
                    MaxConcurrentGameSessionActivations=2,
                    ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,
                                    Parameters=loggingLevel:error +map Winter444,
                                    ConcurrentExecutions=10}]"
```

Festlegen der Protokollierungsstufe für eine laufende Realtime Servers-Flotte (Konsole)

Folgen Sie den Anweisungen unter , Aktualisieren Sie eine Flottenkonfiguration um Ihre Flotte mithilfe der Amazon- GameLift Konsole zu aktualisieren, mit der folgenden Ergänzung:

- Geben Sie auf der Seite Flotte bearbeiten unter Serverprozesszuweisung das Schlüssel-Wert-Paar der Protokollierungsebene (z. B. `loggingLevel:error`) als Wert für Startparameter an. Verwenden Sie ein nicht alphanumerisches Zeichen (außer Komma), um die Protokollierungsebene von allen zusätzlichen Parametern zu trennen (z. B. `loggingLevel:error +map Winter444`).

Festlegen der Protokollierungsebene für eine laufende Realtime Servers-Flotte (AWS CLI)

Folgen Sie den Anweisungen unter , Aktualisieren Sie eine Flottenkonfiguration um Ihre Flotte mithilfe der zu aktualisierenAWS CLI, mit der folgenden Ergänzung:

- Geben Sie im Argument für den `--runtime-configuration` Parameter für `update-runtime-configuration`das Schlüssel-Wert-Paar der Protokollierungsebene (z. B. `loggingLevel:error`) als Wert für an`Parameters`. Verwenden Sie ein nicht alphanumerisches Zeichen (außer Komma), um die Protokollierungsstufe von allen zusätzlichen Parametern zu trennen. Sehen Sie sich das folgende Beispiel an:

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,
                    MaxConcurrentGameSessionActivations=2,
                    ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,
                                    Parameters=loggingLevel:error +map Winter444,
```

```
                                        ConcurrentExecutions=10}]"
```

Sicherheit bei Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Cloud-Sicherheit hat höchste Priorität AWS . Als AWS -Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die eingerichtet wurden, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung zwischen Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS und . Weitere Informationen zu den Compliance-Programmen, die für Amazon gelten GameLift, finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, darunter die Sensibilität Ihrer Daten, die Anforderungen Ihres Unternehmens AWS und die geltenden Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung von Amazon anwenden können GameLift. In den folgenden Themen erfahren Sie, wie Sie Amazon konfigurieren GameLift , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS Dienste nutzen können, die Ihnen helfen, Ihre GameLift Amazon-Ressourcen zu überwachen und zu sichern.

Themen

- Bewährte Sicherheitsmethoden für Amazon GameLift

Datenschutz bei Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Das AWS Modell der gilt für den Datenschutz bei Amazon GameLift. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit

Amazon GameLift oder anderen zusammenarbeiten und die Konsole AWS CLI, API oder AWS SDKs AWS-Services verwenden. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

GameLiftAmazon-spezifische Daten werden wie folgt behandelt:

- Spielserver-Builds und Skripte, die Sie auf Amazon hochladen, GameLift werden in Amazon S3 gespeichert. Nach dem Hochladen ist kein direkter Zugriff des Kunden auf diese Daten mehr möglich. Ein autorisierter Benutzer kann temporären Zugriff erhalten, um Dateien hochzuladen, kann die Dateien jedoch nicht direkt in Amazon S3 anzeigen oder aktualisieren. Verwenden Sie die GameLift Amazon-Konsole oder die Service-API, um Skripts und Builds zu löschen.
- Die Protokolldaten der Spielsitzungen werden nach Abschluss der Spielsitzung für einen begrenzten Zeitraum in Amazon S3 gespeichert. Autorisierte Benutzer können auf die Protokolldaten zugreifen, indem sie sie über einen Link in der GameLift Amazon-Konsole oder durch Aufrufe der Service-API herunterladen.
- Metrik- und Ereignisdaten werden in Amazon gespeichert GameLift und können über die GameLift Amazon-Konsole oder durch Aufrufe der Service-API abgerufen werden. Es können Daten über Flotten, Instances, Platzierungen in Spielsitzungen, Matchmaking-Tickets, Spielsitzungen und Spielersitzungen abgerufen werden. Auf Daten kann auch über Amazon CloudWatch und CloudWatch Events zugegriffen werden.
- Vom Kunden bereitgestellte Daten werden bei Amazon GameLift gespeichert. Autorisierte Benutzer können über Aufrufe der Service-API darauf zugreifen. Potenziell sensible Daten sind etwa Spielerdaten, Spielersitzungs- und Spielsitzungsdaten (einschließlich Verbindungsinformationen), Matchmaker-Daten usw.

> **Note**
>
> Wenn Sie benutzerdefinierte Spieler-IDs in Ihren Anforderungen angeben, wird erwartet, dass es sich bei diesen Werten um anonymisierte UUIDs handelt, die keine identifizierenden Spielerinformationen enthalten.

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag AWS Shared Responsibility Model and GDPR im AWS -Sicherheitsblog.

Verschlüsselung im Ruhezustand

Die Verschlüsselung von GameLift Amazon-spezifischen Daten im Ruhezustand wird wie folgt gehandhabt:

- Spielserver-Builds und -Skripts werden in Amazon S3 S3-Buckets mit serverseitiger Verschlüsselung gespeichert.
- Vom Kunden bereitgestellte Daten werden in einem verschlüsselten Format GameLift bei Amazon gespeichert.

Verschlüsselung während der Übertragung

Verbindungen zu den GameLift Amazon-APIs werden über eine sichere (SSL) Verbindung hergestellt und mit AWS Signature Version 4 authentifiziert (bei der Verbindung über die AWS CLI oder das AWS SDK erfolgt die Signierung automatisch). Die Authentifizierung wird mithilfe der IAM-definierten Zugriffsrichtlinien für die Sicherheitsanmeldeinformationen verwaltet, die zum Herstellen der Verbindung verwendet werden.

Die direkte Kommunikation zwischen Spiele-Clients und Spieleservern geschieht wie folgt:

- Bei benutzerdefinierten Spieleservern, die auf GameLift Amazon-Ressourcen gehostet werden, erfolgt die Kommunikation nicht über den GameLift Amazon-Service. Die Verschlüsselung dieser Kommunikation liegt in der Verantwortung des Kunden. Sie können TLS-fähige Flotten verwenden, damit Ihre Spiele-Clients den Spieleserver bei der Verbindung authentifizieren und die gesamte Kommunikation zwischen Ihrem Spiele-Client und dem Spieleserver verschlüsseln.
- Bei Echtzeitservern mit aktivierter TLS-Zertifikatsgenerierung wird der Datenverkehr zwischen dem Spielclient und den Realtime-Servern, die das Realtime Client SDK verwenden, während der Übertragung verschlüsselt. TCP-Datenverkehr wird mit TLS 1.2 verschlüsselt und UDP-Datenverkehr wird mit DTLS 1.2 verschlüsselt.

Richtlinie für den Datenverkehr zwischen Netzwerken

Sie können sicher remote auf Ihre GameLift Amazon-Instances zugreifen. Für Instances, die Linux verwenden, stellt SSH einen sicheren Kommunikationskanal für den Fernzugriff bereit. Verwenden Sie für Instances, auf denen Windows ausgeführt wird, einen RDP (Remote Desktop Protocol)-Client. Mit Amazon GameLift FleetIQ wird der Fernzugriff auf Ihre Instances mithilfe von AWS Systems Manager Session Manager und Run Command mit TLS 1.2 verschlüsselt, und Anfragen

zum Herstellen einer Verbindung werden mit Sigv4 signiert. Hilfe zum Herstellen einer Verbindung zu einer verwalteten GameLift Amazon-Instance finden Sie unterRemote-Verbindung zu Amazon-GameLift Flotten-Instances.

Identity and Access Management für Amazon GameLift

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem Administratoren den Zugriff auf AWS-Ressourcen sicher steuern können. IAM-Administratoren steuern, wer für die Nutzung von Amazon- GameLift Ressourcen authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) werden kann. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in Amazon GameLift.

Service-Benutzer – Wenn Sie den Amazon- GameLift Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere Amazon- GameLift Funktionen ausführen, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Featuresweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie nicht auf ein Feature in Amazon zugreifen können GameLift, finden Sie weitere Informationen unter Fehlerbehebung für Amazon GameLift -Identität und -Zugriff.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für die Amazon- GameLift Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf Amazon GameLift. Ihre

Aufgabe besteht darin, zu bestimmen, auf welche Amazon- GameLift Funktionen und -Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon verwenden kann GameLift, finden Sie unter Funktionsweise GameLift von Amazon mit IAM.

IAM-Administrator – Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon verfassen können GameLift. Beispiele für GameLift identitätsbasierte Amazon-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon GameLift.

Authentifizierung mit Identitäten

Authentifizierung ist die Art, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuertem Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anfragen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS empfiehlt beispielsweise die Verwendung von Multi-Faktor Authentifizierung (MFA), um die Sicherheit Ihres Kontos zu verbessern. Weitere

Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center-Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto-Root-Benutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode empfiehlt es sich, menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, aufzufordern, den Verbund mit einem Identitätsanbieter zu verwenden, um auf AWS-Services mit temporären Anmeldeinformationen zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus dem Benutzerverzeichnis Ihres Unternehmens, ein Web Identity Provider, AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt werden, auf AWS-Services zugreift. Wenn Verbundidentitäten auf AWS-Konten zugreifen, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen im IAM Identity Center erstellen oder Sie können eine Verbindung mit einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie in allen AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir,

temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff: Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center-Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen: Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff: Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward access sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Serviceverknüpfte Rolle: Eine serviceverknüpfte Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen in Amazon EC2 – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt

werden und – AWS CLIoder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem AWS-Konto anfügen können. Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können verwaltete AWS-Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3, AWS WAF und Amazon VPC sind Beispiele für Dienste, die ACLs unterstützen. Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen:Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. SCPs schränken Berechtigungen für Entitäten in Mitgliedskonten einschließlich des jeweiligen Root-Benutzer des AWS-Kontos ein. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.
- Sitzungsrichtlinien:Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Funktionsweise GameLift von Amazon mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Amazon zu verwalten GameLift, erfahren Sie, welche IAM-Funktionen Sie mit Amazon verwenden können GameLift.

IAM-Funktionen, die Sie mit Amazon verwenden können GameLift

IAM-Feature	Amazon- GameLift Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen Überblick über das Zusammenwirken von Amazon GameLift und anderen -AWSServices mit den meisten IAM-Funktionen finden Sie unter -AWSServices, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien für Amazon GameLift

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon GameLift

Beispiele für GameLift identitätsbasierte Amazon-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon GameLift.

Ressourcenbasierte Richtlinien in Amazon GameLift

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource in unterschiedlichen AWS-Konten befinden, muss ein IAM-Administrator im vertrauenswürdigen Konto

auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Richtlinienaktionen für Amazon GameLift

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie die zugehörige AWS-API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keine passende API-Operation gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon- GameLift Aktionen finden Sie unter Von Amazon definierte Aktionen GameLift in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in Amazon GameLift verwenden das folgende Präfix vor der Aktion:

```
gamelift
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "gamelift:action1",
      "gamelift:action2"
```

```
        ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "gamelift:Describe*"
```

Beispiele für GameLift identitätsbasierte Amazon-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon GameLift.

Richtlinienressourcen für Amazon GameLift

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Amazon- GameLift Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon definierte Ressourcen GameLift in der Service-Autorisierungs-Referenz. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von Amazon definierte Aktionen GameLift.

Einige Amazon- GameLift Ressourcen haben ARN-Werte, wodurch der Zugriff der Ressourcen mithilfe von IAM-Richtlinien verwaltet werden kann. Die Amazon- GameLift Flottenressource hat einen ARN mit der folgenden Syntax:

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon-Ressourcennamen (ARNs) im Allgemeine AWS-Referenz.

Um beispielsweise die `fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa`-Flotte in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN.

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

Um alle Flotten anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie einen Platzhalter (*):

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

Beispiele für GameLift identitätsbasierte Amazon-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon GameLift.

Richtlinienbedingungsschlüssel für Amazon GameLift

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die Bedingung mittels einer logischen OR-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Eine Liste aller globalen AWS-Bedingungsschlüssel finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Eine Liste der Amazon- GameLift Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon GameLift in der Service-Autorisierungs-Referenz. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon definierte Aktionen GameLift.

Beispiele für GameLift identitätsbasierte Amazon-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon GameLift.

ACLs in Amazon GameLift

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon GameLift

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie

können Tags an IAM-Entitäten (Benutzer oder Rollen) und mehrere AWS-Ressourcen anfügen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Ein Beispiel für eine identitätsbasierte Richtlinie, die den Zugriff auf eine Ressource basierend auf den Tags auf dieser Ressource einschränkt, finden Sie unter Anzeigen von Amazon- GameLift Flotten basierend auf Tags.

Verwenden temporärer Anmeldeinformationen mit Amazon GameLift

Unterstützt temporäre Anmeldeinformationen	Ja

Einige AWS-Services Featureieren nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, unter anderem darüber, welche AWS-Services mit temporären Anmeldeinformationen arbeiten, finden Sie unter AWS-Services, die mit IAM arbeiten im IAM-Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen Methode als einem Benutzernamen und einem Passwort bei der AWS Management Console anmelden. Wenn Sie beispielsweise über den Single Sign-On (SSO)-Link Ihres Unternehmens auf AWS zugreifen, erstellt dieser Prozess automatisch temporäre Anmeldeinformationen. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann

die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Sie können mithilfe der AWS CLI- oder AWS-API manuell temporäre Anmeldeinformationen erstellen. Sie können dann diese temporären Anmeldeinformationen verwenden, um auf AWS zuzugreifen. AWS empfiehlt, dass Sie temporäre Anmeldeinformationen dynamisch generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipal-Berechtigungen für Amazon GameLift

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon GameLift

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die Amazon GameLift-Funktionalität beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon dazu Anleitungen GameLift gibt.

Erlauben Sie Ihren von Amazon gehosteten Spieleservern den Zugriff auf andere GameLift-AWSRessourcen, z. B. eine -AWS LambdaFunktion oder eine Amazon-DynamoDB-Datenbank. Da Spieleserver auf Flotten gehostet werden, die Amazon GameLift verwaltet, benötigen Sie eine Servicerolle, die Amazon GameLift eingeschränkten Zugriff auf Ihre anderen AWS Ressourcen gewährt. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Serviceverknüpfte Rollen für Amazon GameLift

Unterstützt serviceverknüpfte Rollen	Nein

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Weitere Informationen zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter -AWSServices, die mit IAM funktionieren im IAM-Benutzerhandbuch. Suchen Sie in der Tabelle nach einem Service, der einen in `Yes` der Spalte Serviceverknüpfte Rollen enthält. Wählen Sie Ja, um die Dokumentation der serviceverknüpften Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon GameLift

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Amazon GameLift-Ressourcen. Sie können auch keine Aufgaben über die AWS Management Console, die AWS Command Line Interface (AWS CLI) oder die AWS-API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von Amazon definiert werden GameLift, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon GameLift in der Service-Autorisierungs-Referenz.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon- GameLift Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS-verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen:Um Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie die AWS-verwaltete Richtlinien die Berechtigungen für viele allgemeine Anwendungsfälle gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete AWS-Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien oder AWS-verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten:Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs:Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch

Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn diese durch ein bestimmtes AWS-Service, wie beispielsweise AWS CloudFormation, verwendet werden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten:IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Bedarf einer Multi-Faktor-Authentifizierung (MFA):Wenn Sie ein Szenario haben, das IAM-Benutzer oder Root-Benutzer in Ihrem AWS-Konto erfordert, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon- GameLift Konsole

Um auf die Amazon- GameLift Konsole zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon GameLift-Ressourcen in Ihrem aufzulisten und anzuzeigenAWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten weiterhin die Amazon- GameLift Konsole verwenden können, fügen Sie Benutzern und Gruppen Berechtigungen mit der Syntax in den folgenden Beispielen und in hinzuBeispiele für Administratorberechtigungen. Weitere Informationen finden Sie unter Benutzerberechtigungen für Amazon verwalten GameLift.

Benutzer, die GameLift über - AWS CLI oder AWS-API-Operationen mit Amazon arbeiten, benötigen keine Mindestberechtigungen für die Konsole. Stattdessen können Sie den Zugriff auf die Operationen beschränken, die der Benutzer ausführen muss. Beispielsweise benötigt ein

Spielerbenutzer, der im Namen von Spielclients handelt, Zugriff, um Spielsitzungen anzufordern, Spieler in Spiele zu platzieren und andere Aufgaben auszuführen.

Informationen zu den erforderlichen Berechtigungen für die Verwendung aller Amazon- GameLift Konsolenfunktionen finden Sie unter Berechtigungssyntax für Administratoren in Beispiele für Administratorberechtigungen.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die AWS-API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
```

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

Erlauben des Spielerzugriffs für Spielsitzungen

Um Spieler in Spielsitzungen zu platzieren, benötigen Spielclients und Backend-Services - Berechtigungen. Beispiele für Richtlinien für diese Szenarien finden Sie unter Beispiele für Benutzerberechtigungen für Spieler.

Zugriff auf eine Amazon- GameLift Warteschlange zulassen

Das folgende Beispiel bietet einem Benutzer Zugriff auf eine bestimmte Amazon GameLift-Warteschlange.

Diese Richtlinie gewährt dem Benutzer Berechtigungen zum Hinzufügen, Aktualisieren und Löschen von Warteschlangenzielen mit den folgenden Aktionen: `gamelift:UpdateGameSessionQueuegamelift:DeleteGameSessionQueue`, und `gamelift:DescribeGameSessionQueues`. Wie gezeigt, verwendet diese Richtlinie das -`Resource`Element, um den Zugriff auf eine einzelne Warteschlange zu beschränken: `gamesessionqueue/examplequeue123`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ViewSpecificQueueInfo",
         "Effect":"Allow",
         "Action":[
            "gamelift:DescribeGameSessionQueues"
         ],
         "Resource":"arn:aws:gamelift:::gamesessionqueue/examplequeue123"
      },
      {
         "Sid":"ManageSpecificQueue",
         "Effect":"Allow",
         "Action":[
            "gamelift:UpdateGameSessionQueue",
            "gamelift:DeleteGameSessionQueue"
         ],
```

```
         "Resource":"arn:aws:gamelift:::gamesessionqueue/examplequeue123"
      }
   ]
}
```

Anzeigen von Amazon- GameLift Flotten basierend auf Tags

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf Amazon- GameLift Ressourcen basierend auf Tags zu steuern. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen können, die das Anzeigen einer Flotte ermöglicht, wenn das `Owner` Tag mit dem Benutzernamen des Benutzers übereinstimmt. Diese Richtlinie gewährt auch die erforderlichen Berechtigungen, um diesen Vorgang in der Konsole abzuschließen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListFleetsInConsole",
            "Effect": "Allow",
            "Action": "gamelift:ListFleets",
            "Resource": "*"
        },
        {
            "Sid": "ViewFleetIfOwner",
            "Effect": "Allow",
            "Action": "gamelift:DescribeFleetAttributes",
            "Resource": "arn:aws:gamelift:*:*:fleet/*",
            "Condition": {
                "StringEquals": {"gamelift:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Zugriff auf eine Spiele-Build-Datei in Amazon S3

Nachdem Sie Ihren Spieleserver in Amazon integriert haben GameLift, laden Sie die Build-Dateien in Amazon S3 hoch. Verwenden Sie die folgende Richtlinie, GameLift damit Amazon auf die Build-Dateien zugreifen kann.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::bucket-name/object-name"
        }
    ]
}
```

Weitere Informationen zum Hochladen von Amazon- GameLift Spieldateien finden Sie unter Laden Sie einen benutzerdefinierten Server-Build auf Amazon hoch GameLift.

Fehlerbehebung für Amazon GameLift -Identität und -Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon GameLift und AWS Identity and Access Management (IAM) auftreten können.

Themen

- Ich bin nicht autorisiert, eine Aktion in Amazon auszuführen GameLift
- Ich bin nicht autorisiert, iam durchzuführen:PassRole
- Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine Amazon- GameLift Ressourcen gewähren

Ich bin nicht autorisiert, eine Aktion in Amazon auszuführen GameLift

Wenn die Ihnen AWS Management Console mitteilt, dass Sie nicht zur Ausführung einer Aktion autorisiert sind, wenden Sie sich an Ihren AWS Kontoadministrator, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zu verwenden, um Details zu einer Warteschlange anzuzeigen, aber keine `gamelift:DescribeGameSessionQueues` Berechtigungen hat:

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

In diesem Fall bittet Mateo seinen Administrator, seine Richtlinien zu aktualisieren, damit er mithilfe der `gamelift:DescribeGameSessionQueues` Aktion Lesezugriff für die `examplequeue123` Ressource erhält.

Ich bin nicht autorisiert, iam durchzuführen:PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Ausführen der `iam:PassRole` Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon übergeben zu können GameLift.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Dienst, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon auszuführen GameLift. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine Amazon-GameLift Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon diese Funktionen GameLift unterstützt, finden Sie unter Funktionsweise GameLift von Amazon mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Protokollierung und Überwachung mit Amazon GameLift

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon GameLift und Ihren AWS Lösungen. Sie sollten von allen Teilen Ihrer AWS-Lösung Überwachungsdaten sammeln, damit Sie Ausfälle, die sich über mehrere Punkte erstrecken, leichter debuggen können.

AWSund Amazon GameLift bieten verschiedene Tools an, mit denen Sie Ihre Game-Hosting-Ressourcen überwachen und auf mögliche Vorfälle reagieren können.

CloudWatchAmazon-Alarme

Mithilfe von CloudWatch Amazon-Alarmen beobachten Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum. Wenn die Metrik einen bestimmten Schwellenwert überschreitet, wird eine Benachrichtigung an ein Amazon SNS-Thema oder eine AWS-Auto-Scaling-Richtlinie gesendet. CloudWatchAlarme werden ausgelöst, wenn sich ihr Zustand ändert, und sie werden für eine bestimmte Anzahl von Zeiträumen aufrechterhalten, nicht weil sie sich in einem bestimmten Zustand befinden. Weitere Informationen finden Sie unter Überwachen Sie Amazon GameLift mit Amazon CloudWatch.

AWSCloudTrailLogs

CloudTrailstellt eine Aufzeichnung der Aktionen bereit, die von einem Benutzer, einer Rolle oder einem AWS Dienst in Amazon ergriffen wurdenGameLift. Anhand der von gesammelten Informationen können Sie ermittelnCloudTrail, welche Anfrage an Amazon gestellt wurdeGameLift, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Informationen. Weitere Informationen finden Sie unter Protokollieren von GameLift Amazon-API-Aufrufen mit AWS CloudTrail.

Konformitätsvalidierung für Amazon GameLift

Amazon GameLift ist nicht im Rahmen von AWS Compliance-Programmen tätig.

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen erstellen AWS können.

> **Note**
>
> AWS-Services Nicht alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz bei Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Die AWS globale Infrastruktur basiert AWS auf Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones

sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur GameLift bietet Amazon die folgenden Funktionen, um Ihre Anforderungen an die Datenstabilität zu erfüllen:

- Warteschlangen für mehrere Regionen — Die Warteschlangen von GameLift Amazon-Spielsitzungen werden verwendet, um neue Spielsitzungen mit verfügbaren Hosting-Ressourcen zu platzieren. Warteschlangen, die sich über mehrere Regionen erstrecken, können Spielsitzungen im Falle eines regionalen Ausfalls umleiten. Weitere Informationen und bewährte Methoden zum Erstellen von Spielsitzungswarteschlangen finden Sie unter Entwerfen Sie eine Warteschlange für Spielsitzungen.
- Automatische Kapazitätsskalierung — Sorgen Sie mithilfe der Skalierungstools von Amazon GameLift für den Zustand und die Verfügbarkeit Ihrer Hosting-Ressourcen. Diese Tools bieten eine Reihe von Optionen, mit denen Sie die Flottenkapazität an die Bedürfnisse Ihres Spiels und Ihrer Spieler anpassen können. Weitere Informationen zur Skalierung finden Sie unter Skalierung der GameLift Amazon-Hosting-Kapazität.
- Verteilung auf Instances — Amazon GameLift verteilt den eingehenden Traffic je nach Flottengröße auf mehrere Instances. Als bewährte Methode sollten Spiele in der Produktion über mehrere Instances verfügen, um die Verfügbarkeit für den Fall zu wahren, dass eine Instance fehlerhaft wird oder nicht mehr reagiert.
- Amazon S3-Speicher — Spieleserver-Builds und Skripte, die auf Amazon hochgeladen werden, GameLift werden in Amazon S3 mithilfe der Standard-Speicherklasse gespeichert, die mehrere Rechenzentrumsreplikationen verwendet, um die Ausfallsicherheit zu erhöhen. Spielsitzungsprotokolle werden auch in Amazon S3 unter Verwendung der Standard-Speicherklasse gespeichert.

Infrastruktursicherheit bei Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Als verwalteter Service GameLift ist Amazon durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe, um GameLift über das Netzwerk auf Amazon zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Wir empfehlen TLS 1.3 oder höher. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Der GameLift Amazon-Service platziert alle Flotten in Amazon Virtual Private Clouds (VPCs), sodass sich jede Flotte in einem logisch isolierten Bereich in der Cloud befindet. AWS Sie können GameLift Amazon-Richtlinien verwenden, um den Zugriff von bestimmten VPC-Endpunkten oder bestimmten VPCs aus zu kontrollieren. Dadurch wird der Netzwerkzugriff auf eine bestimmte GameLift Amazon-Ressource effektiv nur von der spezifischen VPC innerhalb des AWS Netzwerks isoliert. Wenn Sie eine Flotte erstellen, geben Sie einen Bereich von Portnummern und IP-Adressen an. Diese Bereiche begrenzen, wie eingehender Datenverkehr auf gehostete Spieleserver in einer Flotten-VPC zugreifen kann. Verwenden Sie bewährte Standardmethoden für die Sicherheit bei der Auswahl von Flottenzugriffseinstellungen.

Konfiguration und Schwachstellenanalyse in Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Konfiguration und IT-Steuerelemente unterliegen der übergreifenden Verantwortlichkeit von AWS und Ihnen, unserem Kunden. Weitere Informationen finden Sie im Modell der gemeinsamen Verantwortung. AWS AWS kümmert sich um grundlegende Sicherheitsaufgaben wie das Patchen von Gastbetriebssystemen (OS) und Datenbanken, die Firewallkonfiguration und die Notfallwiederherstellung. Diese Verfahren wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in der folgenden Ressource: Amazon Web Services: Überblick über Sicherheitsprozesse (Whitepaper).

Die folgenden bewährten Sicherheitsmethoden beziehen sich auch auf die Konfiguration und Schwachstellenanalyse in Amazon GameLift:

- Kunden sind für die Verwaltung der Software verantwortlich, die auf GameLift Amazon-Instances zum Hosten von Spielen bereitgestellt wird. Das heißt:
  - Von Kunden bereitgestellte Spieleserver-Anwendungssoftware muss regelmäßig gewartet werden, einschließlich Updates und Sicherheitspatches. Um die Spielserversoftware zu aktualisieren, laden Sie einen neuen Build auf Amazon hoch GameLift, erstellen Sie eine neue Flotte dafür und leiten Sie den Traffic auf die neue Flotte um.
  - Das Basis-AMI (Amazon Machine Image), das das Betriebssystem enthält, wird nur aktualisiert, wenn eine neue Flotte erstellt wird. Um das Betriebssystem und andere Anwendungen, die Teil des AMI sind, zu patchen, zu aktualisieren und zu sichern, müssen Flotten regelmäßig recycelt werden, unabhängig von den Updates des Spieleservers.
- Kunden sollten erwägen, ihre Spiele regelmäßig mit den neuesten SDK-Versionen zu aktualisieren, einschließlich des AWS SDK, des Amazon GameLift Server SDK und des Amazon GameLift Client SDK for Realtime Servers.

Bewährte Sicherheitsmethoden für Amazon GameLift

Wenn Sie Amazon GameLift FleetIQ als eigenständige Funktion mit Amazon EC2 verwenden, finden Sie weitere Informationen unter Sicherheit in Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Amazon GameLift bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden stellen allgemeine Richtlinien und keine vollständige Sicherheitslösung dar. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Öffnen Sie keine Ports zum Internet

Es wird dringend davon abgeraten, Ports zum Internet zu öffnen, da dies ein Sicherheitsrisiko darstellt. Wenn Sie UpdateFleetPortSettingszum Beispiel einen Remote-Desktop-Port wie folgt öffnen:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
```

```
      {
        "FromPort": 3389,
        "IpRange": "0.0.0.0/0",
        "Protocol": "RDP",
        "ToPort": 3389
      }
  ]
}
```

dann gestatten Sie jedem im Internet, auf die Instanz zuzugreifen.

Öffnen Sie stattdessen den Port mit einer bestimmten IP-Adresse oder einem bestimmten Adressbereich. Zum Beispiel so:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
      {
        "FromPort": 3389,
        "IpRange": "54.186.139.221/32",
        "Protocol": "TCP",
        "ToPort": 3389
      }
  ]
}
```

Weitere Informationen

Weitere Informationen darüber, wie Sie Ihre Nutzung von Amazon GameLift sicherer machen können, finden Sie in der Säule AWS Well-Architected Tool Sicherheit. .

GameLiftAmazon-Referenzhandbücher

Dieser Abschnitt enthält Referenzdokumentation zur Verwendung von AmazonGameLift.

Themen

- Amazon GameLift Service API-Referenz (AWSSDK)
- Referenz zu Amazon GameLift Realtime Servers
- Referenz zum Amazon GameLift Server-SDK
- Platzierungsveranstaltungen für Spielsitzungen

Amazon GameLift Service API-Referenz (AWSSDK)

Dieses Thema enthält eine aufgabenbasierte Liste von API-Vorgängen zur Verwendung mit von Amazon GameLift verwalteten Hosting-Lösungen, einschließlich des Hostings für benutzerdefinierte Spieleserver und Echtzeitserver. Diese Operationen sind in das AWS SDK im `aws.gamelift` Namespace gepackt. Laden Sie das AWS SDK herunter oder sehen Sie sich die GameLift Amazon-API-Referenzdokumentation an.

Die API umfasst zwei Operationssätze für das Hosting verwalteter Spiele:

- GameLiftAmazon-Hosting-Ressourcen einrichten und verwalten
- Starte Spielsitzungen und schließe dich Spielern an

Die Amazon GameLift Service API enthält auch Operationen zur Verwendung mit anderen GameLift Amazon-Tools und -Lösungen. Eine Liste der FleetIQ APIs finden Sie unter FleetIQ API-Aktionen. Eine Liste der FlexMatch APIs für Matchmaking finden Sie unter FlexMatchAPI-Aktionen.

GameLiftAmazon-Hosting-Ressourcen einrichten und verwalten

Rufen Sie diese Operationen auf, um Hosting-Ressourcen für Ihre Spielserver zu konfigurieren, die Kapazität zu skalieren, um den Anforderungen der Spieler gerecht zu werden, auf Leistungs- und Auslastungskennzahlen zuzugreifen und vieles mehr. Diese API-Operationen werden mit Spielservern verwendet, die auf Amazon gehostet werdenGameLift, einschließlich Echtzeitservern. Sie können die GameLiftAmazon-Konsole für die meisten Ressourcenverwaltungsaufgaben verwenden, oder Sie können den Service mithilfe des Tools AWS Command Line Interface (AWS CLI) oder des AWS SDK aufrufen.

Spielserver für den Einsatz vorbereiten

Laden Sie den Gameservercode Ihres Spiels hoch und konfigurieren Sie ihn, um die Bereitstellung und den Start auf Hosting-Ressourcen vorzubereiten.

Benutzerdefinierte Gameserver-Builds verwalten

- upload-build — Laden Sie Build-Dateien von einem lokalen Pfad hoch und erstellen Sie eine neue GameLift Amazon-Build-Ressource. Dieser Vorgang, der nur als AWS CLI Befehl verfügbar ist, ist die gebräuchlichste Methode zum Hochladen von Gameserver-Builds.
- CreateBuild— Erstellen Sie einen neuen Build mit Dateien, die in einem Amazon S3-Bucket gespeichert sind.
- ListBuilds— Ruft eine Liste aller Builds ab, die in eine GameLift Amazon-Region hochgeladen wurden.
- DescribeBuild— Ruft Informationen ab, die einem Build zugeordnet sind.
- UpdateBuild— Ändern Sie die Build-Metadaten, einschließlich des Build-Namens und der Version.
- DeleteBuild— Entferne einen Build von AmazonGameLift.

Konfigurationsskripts für Echtzeitserver verwalten

- CreateScript— Laden Sie JavaScript Dateien hoch und erstellen Sie eine neue GameLift Amazon-Skriptressource.
- ListScripts— Holen Sie sich eine Liste aller Echtzeit-Skripts, die in eine GameLift Amazon-Region hochgeladen wurden.
- DescribeScript— Ruft Informationen ab, die einem Echtzeit-Skript zugeordnet sind.
- UpdateScript— Ändern Sie die Skriptmetadaten und laden Sie überarbeitete Skriptinhalte hoch.
- DeleteScript— Entfernen Sie ein Realtime-Skript von AmazonGameLift.

Rechenressourcen für das Hosting einrichten

Konfiguriere Hosting-Ressourcen und stelle sie mit deinem Gameserver-Build oder deinem Echtzeitkonfigurationsskript bereit.

Flotten erstellen und verwalten

- CreateFleet— Konfigurieren und implementieren Sie eine neue GameLift Amazon-Flotte von Computerressourcen für den Betrieb Ihrer Spieleserver. Nach der Bereitstellung werden die Spieleserver automatisch wie konfiguriert gestartet und sind bereit, Spielesitzungen zu hosten.
- ListFleets— Holen Sie sich eine Liste aller Flotten in einer GameLift Amazonasregion.
- DeleteFleet— Schließe eine Flotte ab, die keine Spielserver mehr betreibt oder Spieler hostet.
- Flottenstandorte anzeigen/aktualisieren.
  - CreateFleetLocations— Fügen Sie einer bestehenden Flotte, die mehrere Standorte unterstützt, abgelegene Standorte hinzu
  - DescribeFleetLocationAttributes— Holen Sie sich eine Liste aller entfernten Standorte für eine Flotte und sehen Sie sich den aktuellen Status jedes Standorts an.
  - DeleteFleetLocations— Entferne Standorte aus einer Flotte, die mehrere Standorte unterstützt.
- Anzeigen/Aktualisieren von Flottenkonfigurationen.
  - DescribeFleetAttributes/UpdateFleetAttributes— Du kannst die Metadaten und Einstellungen einer Flotte für den Schutz von Spielsitzungen und die Limits zur Ressourcenerstellung einsehen oder ändern.
  - DescribeFleetPortSettings/UpdateFleetPortSettings— Zeigt die für eine Flotte zulässigen Eingangsberechtigungen (IP-Adressen und Porteinstellungsbereiche) an oder ändert sie.
  - DescribeRuntimeConfiguration/UpdateRuntimeConfiguration— Sehen oder ändern Sie, welche Serverprozesse (und wie viele) auf jeder Instance in einer Flotte ausgeführt werden sollen.

Flottenkapazität verwalten

- DescribeEC2 InstanceLimits — Ruft die maximale Anzahl von Instanzen ab, die für das aktuelle AWS Konto und die aktuelle Nutzungsstufe zulässig sind.
- DescribeFleetCapacity— Ruft die aktuellen Kapazitätseinstellungen für die Heimatregion einer Flotte ab.
- DescribeFleetLocationCapacity— Rufen Sie die aktuellen Kapazitätseinstellungen für jeden Standort einer Flotte mit mehreren Standorten ab.
- UpdateFleetCapacity— Passen Sie die Kapazitätseinstellungen für eine Flotte manuell an.
- Richten Sie die automatische Skalierung ein:
  - PutScalingPolicy— Schalten Sie die zielbasierte automatische Skalierung ein oder erstellen Sie eine benutzerdefinierte Autoskalierungsrichtlinie oder aktualisieren Sie eine bestehende Richtlinie.

- DescribeScalingPolicies— Rufen Sie eine bestehende Auto-Scaling-Richtlinie ab.
- DeleteScalingPolicy— Löschen Sie eine Richtlinie zur automatischen Skalierung und verhindern Sie, dass sie sich auf die Kapazität einer Flotte auswirkt.
- StartFleetActions— Starten Sie die Richtlinien für die automatische Skalierung einer Flotte neu.
- StopFleetActions— Setzen Sie die Richtlinien für die automatische Skalierung einer Flotte aus.

Überwachen Sie die Flottenaktivität.

- DescribeFleetUtilization— Ruft Statistiken über die Anzahl der Serverprozesse, Spielsitzungen und Spieler ab, die derzeit in einer Flotte aktiv sind.
- DescribeFleetLocationUtilization— Rufen Sie Nutzungsstatistiken für jeden Standort in einer Flotte mit mehreren Standorten ab.
- DescribeFleetEvents— Zeigt protokollierte Ereignisse für eine Flotte während eines bestimmten Zeitraums an.
- DescribeGameSessions— Ruft Metadaten der Spielsitzung ab, einschließlich der Laufzeit eines Spiels und der aktuellen Spieleranzahl.

Einrichten von Warteschlangen für optimale Platzierung der Spielsitzung

Richten Sie Warteschlangen für mehrere Flotten und Regionen ein, um Spielsitzungen mit den besten verfügbaren Hosting-Ressourcen für Kosten, Latenz und Ausfallsicherheit zu platzieren.

- CreateGameSessionQueue— Erstelle eine Warteschlange, die bei der Bearbeitung von Anfragen für Platzierungen in Spielsitzungen verwendet werden kann.
- DescribeGameSessionQueues— Ruft Warteschlangen für Spielsitzungen ab, die in einer GameLift Amazon-Region definiert wurden.
- UpdateGameSessionQueue— Ändert die Konfiguration einer Warteschlange für Spielsitzungen.
- DeleteGameSessionQueue— Entferne eine Warteschlange für Spielsitzungen aus der Region.

Verwalten von Aliase

Verwenden Sie Aliase, um Ihre Flotten darzustellen oder ein zum Terminal alternatives Ziel zu erstellen. Aliase sind nützlich beim Übergang von Spielaktivitäten von einer Flotte zu einer anderen, wie z. B. bei Aktualisierungen des Spielserver-Builds.

- CreateAlias— Definieren Sie einen neuen Alias und weisen Sie ihn optional einer Flotte zu.
- ListAliases— Ruft alle Flottenaliase ab, die in einer GameLift Amazon-Region definiert sind.
- DescribeAlias— Ruft Informationen zu einem vorhandenen Alias ab.
- UpdateAlias— Ändern Sie die Einstellungen für einen Alias, indem Sie ihn beispielsweise von einer Flotte zu einer anderen umleiten.
- DeleteAlias— Entferne einen Alias aus der Region.
- ResolveAlias— Ruft die Flotten-ID ab, auf die ein bestimmter Alias verweist.

Zugreifen auf Hosting-Instances

Zeigen Sie Informationen zu einzelnen Instances in einer Flotte an oder fordern Sie zur Fehlerbehebung Fernzugriff auf eine bestimmte Flotteninstance an.

- DescribeInstances— Rufen Sie Informationen zu jeder Instance in einer Flotte ab, einschließlich Instance-ID, IP-Adresse, Standort und Status.
- GetInstanceAccess— Fordern Sie die Zugangsdaten an, die für die Remoteverbindung zu einer bestimmten Instance in einer Flotte erforderlich sind.

VPC-Peering einrichten

Erstellen und verwalten Sie VPC-Peering-Verbindungen zwischen Ihren GameLift Amazon-Hosting-Ressourcen und anderen AWS Ressourcen.

- CreateVpcPeeringAuthorization— Autorisieren Sie eine Peering-Verbindung zu einer Ihrer VPCs.
- DescribeVpcPeeringAuthorizations— Rufen Sie gültige Autorisierungen für Peering-Verbindungen ab.
- DeleteVpcPeeringAuthorization— Löscht die Autorisierung einer Peering-Verbindung.
- CreateVpcPeeringConnection— Stellen Sie eine Peering-Verbindung zwischen der VPC für eine GameLift Amazon-Flotte und einer Ihrer VPCs her.
- DescribeVpcPeeringConnections— Rufen Sie Informationen über aktive oder ausstehende VPC-Peering-Verbindungen mit einer GameLift Amazon-Flotte ab.
- DeleteVpcPeeringConnection— Löschen Sie eine VPC-Peering-Verbindung mit einer GameLift Amazon-Flotte.

Starte Spielsitzungen und schließe dich Spielern an

Rufe diese Operationen von deinem Spielclient aus auf, um neue Spielsitzungen zu starten, Informationen über bestehende Spielsitzungen abzurufen und Spieler zu Spielsitzungen zu begleiten. Diese Operationen sind für benutzerdefinierte Spieleserver vorgesehen, die auf Amazon gehostet werdenGameLift. Wenn du Echtzeitserver verwendest, verwalte Spielsitzungen mit demReferenz zur Realtime Server-Client-API (C#).

- Starten Sie neue Spielsitzungen für einen oder mehrere Spieler.
  - StartGameSessionPlacement— Bitten GameLift Sie Amazon, die besten verfügbaren Hosting-Ressourcen zu finden und eine neue Spielsitzung zu starten. Dies ist die bevorzugte Methode, um neue Spielsitzungen zu erstellen. Es stützt sich auf Warteschlangen für Spielsitzungen, um die Verfügbarkeit von Hostings in mehreren Regionen zu verfolgen, und verwendet FleetIQ-Algorithmen, um Platzierungen auf der Grundlage der Spielerlatenz, der Hosting-Kosten, des Standorts usw. zu priorisieren.
  - DescribeGameSessionPlacement— Informieren Sie sich über Details und Status einer Platzierungsanfrage.
  - StopGameSessionPlacement— Stornieren Sie eine Platzierungsanfrage.
  - CreateGameSession— Starte eine neue, leere Spielsitzung an einem bestimmten Flottenstandort. Durch diesen Vorgang haben Sie mehr Kontrolle darüber, wo Sie die Spielsitzung starten, anstatt FleetIQ zur Bewertung der Platzierungsoptionen zu verwenden. Sie müssen der neuen Spielsitzung in einem separaten Schritt Spieler hinzufügen.
- Binden Sie Spieler in bestehende Spielsitzungen ein. Finden Sie laufende Spielsitzungen mit verfügbaren Spielerplätzen und reservieren Sie sie für neue Spieler.
  - CreatePlayerSession— Reserviere einen freien Slot, damit ein Spieler an einer Spielsitzung teilnehmen kann.
  - CreatePlayerSessions— Reservieren Sie offene Plätze für mehrere Spieler, um an einer Spielsitzung teilzunehmen.
- Arbeiten Sie mit Spielsitzungs- und Spielersitzungsdaten. Verwalte Informationen zu Spielsitzungen und Spielersitzungen.
  - SearchGameSessions— Fordere anhand einer Reihe von Suchkriterien eine Liste der aktiven Spielsitzungen an.
  - DescribeGameSessions— Ruft Metadaten für bestimmte Spielsitzungen ab, einschließlich der aktiven Zeit und der aktuellen Spieleranzahl.

- DescribeGameSessionDetails— Ruft Metadaten, einschließlich der Schutzeinstellungen für Spielsitzungen, für eine oder mehrere Spielsitzungen ab.
- DescribePlayerSessions— Informieren Sie sich über die Spieleraktivitäten, einschließlich Status, Spielzeit und Spielerdaten.
- UpdateGameSession— Ändere die Einstellungen der Spielsitzung, z. B. die maximale Spielerzahl und die Teilnahmerichtlinien.
- GetGameSessionLogUrl— Ruft den Speicherort der gespeicherten Logs für eine Spielsitzung ab.

Referenz zu Amazon GameLift Realtime Servers

Dieser Abschnitt enthält Referenzdokumentation für das Amazon GameLift Realtime Servers SDK. Es enthält die Realtime Client API sowie Anleitungen zur Konfiguration Ihres Realtime Server-Skripts.

Themen

- Referenz zur Realtime Server-Client-API (C#)
- Amazon GameLift Realtime Server-Skriptreferenz

Referenz zur Realtime Server-Client-API (C#)

Verwenden Sie die Realtime Client API, um Ihre Multiplayer-Game-Clients für die Verwendung mit Amazon GameLift Realtime Servern vorzubereiten. Weitere Informationen zu dem Integrationsprozess finden Sie unter Bereite deinen Realtime-Server vor. Die Client-API enthält eine Reihe synchroner API-Aufrufe und asynchroner Callbacks, die es einem Spielclient ermöglichen, sich mit einem Echtzeitserver zu verbinden und über den Server Nachrichten und Daten mit anderen Spielclients auszutauschen.

Diese API ist in den folgenden Bibliotheken definiert:

Client.cs

- Synchrone Aktionen
- Asynchrone Callbacks
- Datentypen

So richten Sie die Realtime-Client-API ein

1. Laden Sie das Amazon GameLift Realtime Client SDK herunter.
2. Entwickeln der C # SDK-Bibliotheken. Suchen der Lösungsdatei `GameLiftRealtimeClientSdkNet45.sln`. In der `README.md`-Datei für das C# Server SDK finden Sie Informationen zu den Mindestanforderungen und zusätzliche Build-Optionen. Laden Sie die Lösungsdatei in einer integrierten Entwicklungsumgebung (IDE). Um die SDK-Bibliotheken zu generieren, stellen Sie die NuGet Pakete wieder her und erstellen Sie die Lösung.
3. Füge die Realtime Client-Bibliotheken zu deinem Game-Client-Projekt hinzu.

Referenz zur Realtime Server-Client-API (C#): Aktionen

Diese C#-Realtime-Client-API-Referenz kann Ihnen helfen, Ihr Multiplayer-Spiel für die Verwendung mit Echtzeitservern vorzubereiten, die auf Amazon-Flotten eingesetzt werden. GameLift Weitere Informationen zu dem Integrationsprozess finden Sie unter Bereite deinen Realtime-Server vor.

- Synchrone Aktionen
- Asynchrone Callbacks
- Datentypen

Client()

Initialisiert einen neuen Client für die Kommunikation mit dem Echtzeit-Server und ermittelt den zu verwendenden Verbindungstyp.

Syntax

```
public Client(ClientConfiguration configuration)
```

Parameter

clientConfiguration

Konfigurationsdetails, die den Client/Server-Verbindungstyp angeben. Sie können "Client()" ohne diesen Parameter aufrufen. Dieser Ansatz führt jedoch standardmäßig zu einer ungesicherten Verbindung.

Typ: ClientConfiguration

Required: No

Rückgabewert

Gibt eine Instance des Echtzeit-Clients zur Verwendung bei der Kommunikation mit dem Echtzeit-Server zurück.

Connect()

Fordert eine Verbindung zu einem Serverprozess an, der eine Spielsitzung hostet.

Syntax

```
public ConnectionStatus Connect(string endpoint, int remoteTcpPort, int listenPort,
 ConnectionToken token)
```

Parameter

Endpunkt

DNS-Name oder IP-Adresse der Spielsitzung, mit der eine Verbindung hergestellt werden soll. Der Endpunkt ist in einem `GameSession` Objekt angegeben, das als Antwort auf einen Client-Aufruf an die GameLiftAmazon-API-Aktionen des AWS SDK zurückgegeben wird StartGameSessionPlacementCreateGameSession, oder DescribeGameSessions.

> **Note**
>
> Wenn der Echtzeitserver auf einer Flotte mit einem TLS-Zertifikat ausgeführt wird, müssen Sie den DNS-Namen verwenden.

Typ: Zeichenfolge

Erforderlich: Ja

remoteTcpPort

Die Portnummer für die TCP-Verbindung, die der Spielsitzung zugewiesen ist. Diese Information wird in einem `GameSession` Objekt spezifiziert, das als Antwort auf eine

StartGameSessionPlacementCreateGameSessionoder DescribeGameSession-Anfrage zurückgegeben wird.

Typ: Ganzzahl

Gültige Werte: 1900 bis 2000.

Erforderlich: Ja

listenPort

Die Portnummer, die der Spiele-Client auf über den UDP-Kanal gesendete Nachrichten überwacht.

Typ: Ganzzahl

Gültige Werte: 33400 bis 33500.

Erforderlich: Ja

Token

Optionale Informationen, die den anfragenden Spiele-Client beim Serverprozess identifizieren.

Typ: ConnectionToken

Erforderlich: Ja

Rückgabewert

Gibt einen ConnectionStatusEnum-Wert zurück, der den Verbindungsstatus des Clients angibt.

Disconnect()

Trennt den Spiele-Client von der gegebenenfalls bestehenden Spielsitzung.

Syntax

```
public void Disconnect()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Diese Methode gibt nichts zurück.

NewMessage()

Erstellt ein neues Nachrichtenobjekt mit einem angegebenen Operationscode. Wenn ein Nachrichtenobjekt zurückgegeben wird, vervollständigen Sie den Inhalt der Nachricht, indem Sie ein Ziel angeben, die Zustellungsmethode aktualisieren und eine Datennutzlast wie erforderlich hinzufügen. Wenn Sie diesen Vorgang abgeschlossen haben, senden Sie die Nachricht mithilfe von `SendMessage()`.

Syntax

```
public RTMessage NewMessage(int opCode)
```

Parameter

opCode

Vom Entwickler definierter Operationscode, der ein Spielereignis oder eine Aktion identifiziert, wie beispielsweise eine Spielerbewegung oder eine Serverbenachrichtigung.

Typ: Ganzzahl

Erforderlich: Ja

Rückgabewert

Gibt ein RTMessage-Objekt mit dem angegebene Operationscode und der Standard-Zustellungsmethode zurück. Der Zustellungsziel-Parameter ist standardmäßig auf `FAST` festgelegt.

SendMessage()

Sendet mit der angegebenen Zustellungsmethode eine Nachricht an einen Spieler oder eine Gruppe.

Syntax

```
public void SendMessage(RTMessage message)
```

Parameter

Nachricht

Nachrichtenobjekt, das den Zielempfänger, die Zustellungsmethode und den Inhalt der Nachricht angibt.

Typ: RTMessage

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück.

JoinGroup()

Fügt der Spieler als Mitglied zu einer bestimmten Gruppe hinzu. Gruppen können alle Spieler enthalten, die mit dem Spiel verbunden sind. Sobald der Spieler hinzugefügt wurde, erhält er alle zukünftigen an die Gruppe gesendeten Nachrichten und kann Nachrichten an die gesamte Gruppe senden.

Syntax

```
public void JoinGroup(int targetGroup)
```

Parameter

targetGroup

Eindeutige ID der Gruppe, zu der der Spieler hinzugefügt werden soll. Gruppen-IDs werden vom Entwickler definiert.

Typ: Ganzzahl

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück. Da diese Anfrage mit der zuverlässigen (TCP-)Zustellungsmethode gesendet wird, löst eine fehlgeschlagene Anfrage den Callback OnError() aus.

LeaveGroup()

Entfernt den Spieler als Mitglied aus einer bestimmten Gruppe. Wenn der Spieler kein Mitglied der Gruppe mehr ist, empfängt er keine an die Gruppe gesendeten Nachrichten mehr und kann keine Nachrichten mehr an die gesamte Gruppe senden.

Syntax

```
public void LeaveGroup(int targetGroup)
```

Parameter

targetGroup

Eindeutige ID der Gruppe, aus der der Spieler entfernt werden soll. Gruppen-IDs werden vom Entwickler definiert.

Typ: Ganzzahl

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück. Da diese Anfrage mit der zuverlässigen (TCP-)Zustellungsmethode gesendet wird, löst eine fehlgeschlagene Anfrage den Callback OnError() aus.

RequestGroupMembership()

Anfrage, dass eine Liste der Spieler in der angegebenen Gruppe an den Spiele-Client gesendet wird. Alle Spieler können diese Informationen anfordern, unabhängig davon, ob sie ein Mitglied dieser Gruppe sind oder nicht. Als Antwort auf diese Anfrage wird die Mitgliederliste über einen OnGroupMembershipUpdated()-Callback an den Client gesendet.

Syntax

```
public void RequestGroupMembership(int targetGroup)
```

Parameter

targetGroup

Eindeutige ID der Gruppe, für die Informationen zur Gruppenmitgliedschaft abgerufen werden sollen. Gruppen-IDs werden vom Entwickler definiert.

Typ: Ganzzahl

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück.

Referenz zur Realtime Server-Client-API (C#): Asynchrone Callbacks

Verwenden Sie diese C#-Realtime-Client-API-Referenz, um Ihr Multiplayer-Spiel für die Verwendung mit Echtzeitservern vorzubereiten, die auf Amazon-Flotten eingesetzt werden. GameLift Weitere Informationen zu dem Integrationsprozess finden Sie unter Bereite deinen Realtime-Server vor.

- Synchrone Aktionen
- Asynchrone Callbacks
- Datentypen

Ein Spiele-Client muss diese Callback-Methoden implementieren, um auf Ereignisse reagieren zu können. Der Realtime-Server ruft diese Callbacks auf, um spielbezogene Informationen an den Spielclient zu senden. Callbacks für dieselben Ereignisse können auch mit benutzerdefinierter Spiellogik im Realtime-Serverskript implementiert werden. Siehe Script-Callbacks für Echtzeitserver.

Callback-Methoden sind in `ClientEvents.cs` definiert.

OnOpen()

Wird aufgerufen, wenn der Serverprozess die Verbindungsanfrage des Spiele-Clients akzeptiert und eine Verbindung öffnet.

Syntax

```
public void OnOpen()
```

Parameter

Diese Methode verwendet keine Parameter.

Rückgabewert

Diese Methode gibt nichts zurück.

OnClose()

Wird aufgerufen, wenn der Serverprozess die Verbindung mit dem Spiele-Client beendet, z. B. wenn eine Spielsitzung endet.

Syntax

```
public void OnClose()
```

Parameter

Diese Methode verwendet keine Parameter.

Rückgabewert

Diese Methode gibt nichts zurück.

OnError()

Wird aufgerufen, wenn ein Fehler für eine Realtime Client-API-Anfrage auftritt. Dieser Callback kann an eine Vielzahl von Verbindungsfehlern angepasst werden.

Syntax

```
private void OnError(byte[] args)
```

Parameter

Diese Methode verwendet keine Parameter.

Rückgabewert

Diese Methode gibt nichts zurück.

OnDataReceived()

Wird aufgerufen, wenn der Spielclient eine Nachricht vom Realtime-Server empfängt. Dies ist Methode, mit der die meisten Nachrichten und Benachrichtigungen von einem Spiele-Client empfangen werden.

Syntax

```
public void OnDataReceived(DataReceivedEventArgs dataReceivedEventArgs)
```

Parameter

dataReceivedEventArgs

Information im Zusammenhang mit Nachrichten-Aktivitäten.

Typ: DataReceivedEventArgs

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück.

OnGroupMembershipUpdated()

Wird aufgerufen, wenn die Mitgliedschaft für eine Gruppe, der der Spieler angehört, aktualisiert wurde. Dieser Callback wird auch aufgerufen, wenn ein Client `RequestGroupMembership` aufruft.

Syntax

```
public void OnGroupMembershipUpdated(GroupMembershipEventArgs groupMembershipEventArgs)
```

Parameter

groupMembershipEventArgs

Informationen im Zusammenhang mit Gruppenmitgliedschaftsaktivitäten.

Typ: GroupMembershipEventArgs

Erforderlich: Ja

Rückgabewert

Diese Methode gibt nichts zurück.

Referenz zur Realtime Server-Client-API (C#): Datentypen

Diese C#-Realtime-Client-API-Referenz kann Ihnen helfen, Ihr Multiplayer-Spiel für die Verwendung mit Echtzeitservern vorzubereiten, die auf Amazon-Flotten eingesetzt werden. GameLift Weitere Informationen zu dem Integrationsprozess finden Sie unter Bereite deinen Realtime-Server vor.

- Synchrone Aktionen
- Asynchrone Callbacks
- Datentypen

ClientConfiguration

Informationen darüber, wie der Spielclient eine Verbindung zu einem Realtime-Server herstellt.

Inhalt

ConnectionType

Typ der zu verwendenden Client/Server-Verbindung, entweder gesichert oder ungesichert. Wenn Sie keinen Verbindungstyp angeben, ist der Standardwert ungesichert.

> **Note**
>
> Wenn Sie eine Verbindung zu einem Echtzeit-Server in einer gesicherten Flotte mit einem TLS-Zertifikat herstellen, müssen Sie den Wert "RT_OVER_WSS_DTLS_TLS12" verwenden.

Typ: Ein `ConnectionType` enum-Wert.

Required: No

ConnectionToken

Informationen über den Spielclient und/oder den Spieler, der eine Verbindung mit einem Realtime-Server anfordert.

Inhalt

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird, GameLift wenn eine neue Spielersitzung erstellt wird. Eine Player-Sitzungs-ID ist in einem `PlayerSession` Objekt angegeben, das als Antwort auf einen Client-Aufruf der GameLiftAPI-Aktionen StartGameSessionPlacement, CreateGameSessionDescribeGameSessionPlacement, oder zurückgegeben wird DescribePlayerSessions.

Typ: Zeichenfolge

Erforderlich: Ja

Nutzlast

Vom Entwickler definierte Informationen, die bei der Verbindung an den Echtzeitserver übermittelt werden. Hierzu gehören alle beliebigen Daten, die für einen benutzerdefinierten Anmeldemechanismus verwendet werden könnten. Eine Payload kann beispielsweise Authentifizierungsinformationen bereitstellen, die vom Realtime-Serverskript verarbeitet werden, bevor ein Client eine Verbindung herstellen kann.

Typ: Byte-Array

Required: No

RTMessage

Inhalts- und Lieferinformationen zu einer Nachricht. Eine Nachricht muss entweder einen Zielspieler oder eine Zielgruppe angeben.

Inhalt

opCode

Vom Entwickler definierter Operationscode, der ein Spielereignis oder eine Aktion identifiziert, wie beispielsweise eine Spielerbewegung oder eine Serverbenachrichtigung. Der Operationscode einer Nachricht stellt einen Kontext für die Nutzlast her, die bereitgestellt wird. Für Nachrichten, die mit NewMessage () erstellt wurden, ist der Operationscode bereits gesetzt, er kann jedoch jederzeit geändert werden.

Typ: Ganzzahl

Erforderlich: Ja

targetPlayer

Eindeutige ID des Spielers, der der Empfänger der gesendeten Nachricht sein soll. Das Ziel kann der Server selbst (mithilfe der Server-ID) oder ein anderer Spieler (mithilfe einer Spieler-ID) sein.

Typ: Ganzzahl

Required: No

targetGroup

Eindeutige ID der Gruppe, die die Empfängerin der gesendeten Nachricht sein soll. Gruppen-IDs werden vom Entwickler definiert.

Typ: Ganzzahl

Required: No

deliveryIntent

Gibt an, ob die Nachricht über die zuverlässige TCP-Verbindung oder über den schnellen UDP-Kanal gesendet werden soll. Mit NewMessage() erstellte Nachrichten.

Typ: DeliveryIntent enum

Zulässige Werte: FAST | RELIABLE

Erforderlich: Ja

Nutzlast

Nachrichteninhalt. Diese Informationen werden wie erforderlich gegliedert, um durch den Spiele-Client auf Basis des begleitenden Operationscodes verarbeitet werden zu können. Hierzu können Spielstandsdaten oder sonstige Informationen gehören, die zwischen Spiele-Clients oder zwischen einem Spiele-Client und dem Echtzeit-Server kommuniziert werden müssen.

Typ: Byte-Array

Required: No

DataReceivedEventArgs

Mit einem OnDataReceived()-Callback bereitgestellte Daten.

Inhalt

sender

Eindeutige ID der Entität (Spieler-ID oder Server-ID), von der die Nachricht ursprünglich versendet wurde.

Typ: Ganzzahl

Erforderlich: Ja

opCode

Vom Entwickler definierter Operationscode, der ein Spielereignis oder eine Aktion identifiziert, wie beispielsweise eine Spielerbewegung oder eine Serverbenachrichtigung. Der Operationscode einer Nachricht stellt einen Kontext für die Nutzlast her, die bereitgestellt wird.

Typ: Ganzzahl

Erforderlich: Ja

data

Nachrichteninhalt. Diese Informationen werden wie erforderlich gegliedert, um durch den Spiele-Client auf Basis des begleitenden Operationscodes verarbeitet werden zu können. Hierzu können Spielstandsdaten oder sonstige Informationen gehören, die zwischen Spiele-Clients oder zwischen einem Spiele-Client und dem Echtzeit-Server kommuniziert werden müssen.

Typ: Byte-Array

Required: No

GroupMembershipEventArgs

Mit einem OnGroupMembershipUpdated()-Callback bereitgestellte Daten.

Inhalt

sender

Eindeutige ID des Spielers, der eine Aktualisierung einer Gruppenmitgliedschaft angefordert hat.

Typ: Ganzzahl

Erforderlich: Ja

opCode

Vom Entwickler definierter Operationscode, der ein Spielereignis oder eine Aktion identifiziert.

Typ: Ganzzahl

Erforderlich: Ja

groupId

Eindeutige ID der Gruppe, die die Empfängerin der gesendeten Nachricht sein soll. Gruppen-IDs werden vom Entwickler definiert.

Typ: Ganzzahl

Erforderlich: Ja

playerId

Liste der IDs von Spielern, die aktuelle Mitglieder der angegebenen Gruppe sind.

Typ: Ganzzahl-Array

Erforderlich: Ja

Aufzählungen

Enums, die für das Realtime Client SDK definiert sind, sind wie folgt definiert:

ConnectionStatus

- VERBUNDEN — Der Spielclient ist nur über eine TCP-Verbindung mit dem Realtime-Server verbunden. Alle Nachrichten werden über TCP gesendet, unabhängig vom Zustellungsziel.
- CONNECTED_SEND_FAST — Der Spielclient ist über eine TCP- und eine UDP-Verbindung mit dem Realtime-Server verbunden. Die Möglichkeit, Nachrichten über UDP zu empfangen, wurde jedoch noch nicht überprüft. Daher werden alle an den Spiele-Client gerichteten Nachrichten über TCP versendet.
- CONNECTED_SEND_AND_RECEIVE_FAST — Der Spielclient ist über eine TCP- und eine UDP-Verbindung mit dem Realtime-Server verbunden. Der Spiele-Client kann Nachrichten entweder über TCP oder UDP senden und empfangen.

- Der CONNECTING Game-Client hat eine Verbindungsanfrage gesendet und der Realtime-Server verarbeitet sie.
- DISCONNECTED_CLIENT_CALL — Der Spielclient wurde als Reaktion auf eine Anfrage des Spielclients vom Realtime-Server getrennt. Disconnect()
- GETRENNT — Der Spielclient wurde aus einem anderen Grund als einer Verbindungsunterbrechung des Clients vom Realtime-Server getrennt.

ConnectionType

- RT_OVER_WSS_DTLS_TLS12 — Sicherer Verbindungstyp.

  Zur Verwendung mit Echtzeitservern, die auf einer GameLift Flotte laufen und für die ein TLS-Zertifikat generiert wurde. Bei Verwendung einer sicheren Verbindung wird TCP-Datenverkehr mit TLS 1.2 und UDP-Datenverkehr mit DTLS 1.2 verschlüsselt.
- RT_OVER_WS_UDP_UNSECURED — Unsicherer Verbindungstyp.
- RT_OVER_WEBSOCKET — Unsicherer Verbindungstyp. Dieser Wert wird nicht mehr bevorzugt.

DeliveryIntent

- SCHNELL — Wird über einen UDP-Kanal bereitgestellt.
- ZUVERLÄSSIG — Wird über eine TCP-Verbindung bereitgestellt.

Amazon GameLift Realtime Server-Skriptreferenz

Verwenden Sie diese Ressourcen, um benutzerdefinierte Logik in Ihren Echtzeit-Skripten zu erstellen.

Themen

- Script-Callbacks für Echtzeitserver
- Serverschnittstelle in Echtzeit

Script-Callbacks für Echtzeitserver

Sie können benutzerdefinierte Logik bereitstellen, um auf Ereignisse zu reagieren, indem Sie diese Callbacks in Ihrem Realtime-Skript implementieren.

Init

Initialisiert den Realtime-Server und erhält eine Echtzeit-Serverschnittstelle.

Syntax

```
init(rtsession)
```

onMessage

Wird aufgerufen, wenn eine empfangene Nachricht an den Server gesendet wird.

Syntax

```
onMessage(gameMessage)
```

onHealthCheck

Wird aufgerufen, um den Status der Spielsitzung festzulegen. Standardmäßig ist der Zustandsstatus "Gesund" (oder `true`). Dieser Callback kann implementiert werden, um benutzerdefinierte Zustandsprüfungen durchzuführen und einen Status zurückzugeben.

Syntax

```
onHealthCheck()
```

onStartGameSitzung

Wird aufgerufen, wenn eine neue Spielsitzung beginnt, wobei ein Spielsitzungsobjekt übergeben wird.

Syntax

```
onStartGameSession(session)
```

onProcessTerminate

Wird aufgerufen, wenn der Serverprozess vom GameLift Amazon-Dienst beendet wird. Dies kann als Auslöser dienen, um die Spielsitzung ordnungsgemäß zu verlassen. Es besteht keine Notwendigkeit für den Aufruf von `processEnding()`..

Syntax

```
onProcessTerminate()
```

onPlayerConnect

Wird aufgerufen, wenn ein Spieler eine Verbindung anfordert und die Erstvalidierung bestanden hat.

Syntax

```
onPlayerConnect(connectMessage)
```

onPlayerAccepted

Wird aufgerufen, wenn eine Spielerverbindung akzeptiert wird.

Syntax

```
onPlayerAccepted(player)
```

onPlayerDisconnect

Wird aufgerufen, wenn ein Spieler die Spielsitzung verlässt, entweder durch Senden einer Trennanfrage oder auf andere Weise.

Syntax

```
onPlayerDisconnect(peerId)
```

onProcessStarted

Wird aufgerufen, wenn ein Serverprozess gestartet wird. Dieser Callback ermöglicht es dem Skript, benutzerdefinierte Aufgaben durchzuführen, die zur Vorbereitung des Hostens einer Spielesitzung erforderlich sind.

Syntax

```
onProcessStarted(args)
```

onSendToSpieler

Wird aufgerufen, wenn eine Nachricht auf dem Server von einem Spieler empfangen wird, die an einen anderen Spieler übermittelt werden soll. Dieser Prozess wird ausgeführt, bevor die Nachricht zugestellt wird.

Syntax

```
onSendToPlayer(gameMessage)
```

onSendToGruppe

Wird aufgerufen, wenn eine Nachricht auf dem Server von einem Spieler empfangen wird, die an eine Gruppe übermittelt werden soll. Dieser Prozess wird ausgeführt, bevor die Nachricht zugestellt wird.

Syntax

```
onSendToGroup(gameMessage))
```

onPlayerJoinGruppe

Wird aufgerufen, wenn ein Spieler eine Anfrage zum Beitritt zu einer Gruppe sendet.

Syntax

```
onPlayerJoinGroup(groupId, peerId)
```

onPlayerLeaveGruppe

Wird aufgerufen, wenn ein Spieler eine Anfrage zum Verlassen zu einer Gruppe sendet.

Syntax

```
onPlayerLeaveGroup(groupId, peerId)
```

Serverschnittstelle in Echtzeit

Wenn ein Realtime-Skript initialisiert wird, wird eine Schnittstelle zum Realtime-Server zurückgegeben. Dieses Thema beschreibt die Eigenschaften und Methoden, die über die Schnittstelle verfügbar sind. Erfahren Sie mehr über das Schreiben von Echtzeit-Skripten und sehen Sie sich ein detailliertes Skriptbeispiel unter anErstellen eines Echtzeit-Skripts.

Die Echtzeitschnittstelle bietet Zugriff auf die folgenden Objekte:

- Sitzung

- player
- gameMessage
- Konfiguration

Realtime Session-Objekt

Verwenden Sie diese Methoden, um auf serverbezogene Informationen zuzugreifen und serverbezogene Aktionen durchzuführen.

getPlayers()

Ruft eine Liste von Peer-IDs für Spieler ab, die aktuell mit der Spielsitzung verbunden sind. Liefert ein Array von Spielerobjekten.

Syntax

```
rtSession.getPlayers()
```

broadcastGroupMembershipAktualisieren ()

Löst die Bereitstellung einer aktualisierten Liste der Gruppenmitglieder an die Spielergruppe aus. Geben Sie an, welche Mitgliedschaft übertragen werden soll (groupIdToBroadcast) und welche Gruppe das Update erhalten soll (targetGroupId). Gruppen-IDs müssen eine positive Ganzzahl oder „-1“ sein, um alle Gruppen anzugeben. Ein Beispiel Beispiel für ein Realtime Server-Skript für benutzerdefinierte Gruppen-IDs finden Sie unter.

Syntax

```
rtSession.broadcastGroupMembershipUpdate(groupIdToBroadcast, targetGroupId)
```

getServerId()

Ruft die eindeutige Peer-ID-Kennung des Servers ab, mit der Nachrichten an den Server weitergeleitet werden.

Syntax

```
rtSession.getServerId()
```

getAllPlayersGroupId()

Ermittelt die Gruppen-ID für die Standardgruppe, die alle Spieler enthält, die aktuell mit der Spielsitzung verbunden sind.

Syntax

```
rtSession.getAllPlayersGroupId()
```

processEnding()

Löst den Realtime-Server aus, den Spielserver zu beenden. Diese Funktion muss vom Realtime-Skript aus aufgerufen werden, um eine Spielsitzung ordnungsgemäß zu beenden.

Syntax

```
rtSession.processEnding()
```

getGameSessionAusweis ()

Ermittelt die eindeutige ID der gerade laufenden Spielsitzung.

Syntax

```
rtSession.getGameSessionId()
```

getLogger()

Ruft die Schnittstelle für die Protokollierung ab. Verwenden Sie dies, um Anweisungen zu protokollieren, die in Ihren Spielsitzungsprotokollen erfasst werden. Der Protokollierer unterstützt die Verwendung der Anweisungen "info", "warn" und "error". Zum Beispiel: `logger.info("<string>")`.

Syntax

```
rtSession.getLogger()
```

sendMessage()

Sendet eine mit `newTextGameMessage` oder `newBinaryGameMessage` erstellte Nachricht vom Realtime-Server über den UDP-Kanal an einen Player-Empfänger. Identifizieren Sie die Empfänger anhand der Gruppen-ID.

Syntax

```
rtSession.sendMessage(gameMessage, targetPlayer)
```

sendGroupMessage()

Sendet eine mit `newTextGameMessage` oder `newBinaryGameMessage` vom Realtime-Server erstellte Nachricht an alle Spieler in einer Spielergruppe, die den UDP-Kanal verwenden. Gruppen-IDs müssen eine positive Ganzzahl oder „-1“ sein, um alle Gruppen anzugeben. Ein Beispiel Beispiel für ein Realtime Server-Skript für benutzerdefinierte Gruppen-IDs finden Sie unter.

Syntax

```
rtSession.sendGroupMessage(gameMessage, targetGroup)
```

sendReliableMessage()

Sendet eine mit `newTextGameMessage` oder `newBinaryGameMessage` erstellte Nachricht vom Realtime-Server über den TCP-Kanal an einen Player-Empfänger. Identifizieren Sie die Empfänger anhand der Gruppen-ID.

Syntax

```
rtSession.sendReliableMessage(gameMessage, targetPlayer)
```

sendReliableGroupNachricht ()

Sendet eine mit `newTextGameMessage` oder `newBinaryGameMessage` vom Realtime-Server erstellte Nachricht an alle Spieler in einer Spielergruppe, die den TCP-Kanal verwenden. Gruppen-IDs, die eine positive Ganzzahl oder „-1“ sein müssen, um alle Gruppen anzugeben. Ein Beispiel Beispiel für ein Realtime Server-Skript für benutzerdefinierte Gruppen-IDs finden Sie unter.

Syntax

```
rtSession.sendReliableGroupMessage(gameMessage, targetGroup)
```

newTextGameNachricht ()

Erstellt eine neue Nachricht mit Text, die mithilfe der SendMessage Funktionen vom Server an die Spieleremp fänger gesendet wird. Das Nachrichtenformat ist ähnlich dem Format, das im Realtime Client SDK verwendet wird (siehe RTMessage). Gibt ein `gameMessage`-Objekt zurück.

Syntax

```
rtSession.newTextGameMessage(opcode, sender, payload)
```

newBinaryGameNachricht ()

Erstellt eine neue Nachricht mit Binärdaten, die mithilfe der SendMessage Funktionen vom Server an die Spielerempfänger gesendet werden soll. Das Nachrichtenformat ist ähnlich dem Format, das im Realtime Client SDK verwendet wird (siehe RTMessage). Gibt ein `gameMessage`-Objekt zurück.

Syntax

```
rtSession.newBinaryGameMessage(opcode, sender, binaryPayload)
```

Spielerobjekt

Zugriff auf spielerbezogene Informationen.

player.peerId

Eindeutige ID, die einem Spielclient zugewiesen wird, wenn er sich mit dem Realtime-Server verbindet und der Spielsitzung beitritt.

Spieler. playerSessionId

Spieler-ID, auf die der Spielclient verwiesen hat, als dieser eine Verbindung zum Realtime-Server herstellte und der Spielsitzung beitrat.

Objekt für Spielmitteilungen

Verwenden Sie diese Methoden, um auf Nachrichten zuzugreifen, die vom Realtime-Server empfangen werden. Nachrichten, die von Spiel-Clients empfangen werden, haben die Struktur RTMessage.

getPayloadAsText ()

Ruft die Payload der Spielnachricht als Text ab.

Syntax

```
gameMessage.getPayloadAsText()
```

gameMessage.opcode

Operationscode, der in einer Nachricht enthalten ist.

gameMessage.payload

Payload in einer Nachricht. Kann Text oder binär sein.

gameMessage.sender

Peer-ID des Spiel-Clients, der eine Nachricht gesendet hat.

gameMessage.reliable

Boolean-Wert, der angibt, ob die Nachricht über TCP (true) oder UDP (false) gesendet wurde.

Konfigurationsobjekt

Das Konfigurationsobjekt kann verwendet werden, um Standardkonfigurationen zu überschreiben.

Configuration.Max Players

Die maximale Anzahl von Client-/Server-Verbindungen, die von akzeptiert werden könnenRealTimeServers.

Die Standardeinstellung ist 32.

Konfiguration. pingIntervalTime

Zeitintervall in Millisekunden, in dem der Server versucht, einen Ping an alle verbundenen Clients zu senden, um zu überprüfen, ob die Verbindungen fehlerfrei sind.

Die Standardeinstellung ist 3000 ms.

Referenz zum Amazon GameLift Server-SDK

Dieser Abschnitt enthält Referenzdokumentation für das Amazon GameLift Server SDK. Verwenden Sie das Server-SDK, um Ihre benutzerdefinierten Spieleserver für die Kommunikation mit dem GameLift Amazon-Dienst zu integrieren.

Themen

- Amazon GameLift Server-SDK-Referenz für C++

- Amazon GameLift Server-SDK-Referenz für C#
- Amazon GameLift Server-SDK-Referenz für Go
- Amazon GameLift Server-SDK-Referenz für Unreal Engine

Amazon GameLift Server-SDK-Referenz für C++

Sie können diese Amazon GameLift C++-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereitenGameLift. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Amazon GameLift Server SDK 5.x-Referenz für C++
- Referenz zum Amazon GameLift C++-Server-SDK 3.x

Amazon GameLift Server SDK 5.x-Referenz für C++

Diese Referenz zum Amazon GameLift C++ Server SDK 5.x kann Ihnen helfen, Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

> **Note**
>
> In diesem Thema wird die Amazon GameLift C++-API beschrieben, die Sie beim Erstellen mit der C++-Standardbibliothek (`std`) verwenden können. Insbesondere gilt diese Dokumentation für Code, den Sie mit der `-DGAMELIFT_USE_STD=1` Option kompilieren.

Themen

- Amazon GameLift Server SDK (C++) 5.x-Referenz: Aktionen
- Amazon GameLift Server SDK (C++)-Referenz: Datentypen

Amazon GameLift Server SDK (C++) 5.x-Referenz: Aktionen

Sie können diese Amazon GameLift -C++-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

> **Note**
>
> In diesem Thema wird die Amazon GameLift -C++-API beschrieben, die Sie verwenden können, wenn Sie mit der C++ Standard Library () erstellen`std`. Insbesondere gilt diese Dokumentation für Code, den Sie mit der `-DGAMELIFT_USE_STD=1` Option kompilieren.

Aktionen

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessReadyAsync()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- Zerstören ()

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als the section called "AwsStringOutcome"-Objekt zurück. Das zurückgegebene Objekt enthält die Versionsnummer (Beispiel `5.0.0`). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
 Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine verwaltete EC2-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Server und dem Amazon GameLift-Service einzurichten.

Syntax

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

Rückgabewert

Gibt ein the section called "InitSDKOutcome" Objekt zurück, das angibt, ob der Serverprozess bereit ist, aufzurufenProcessReady().

Beispiel

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
  Aws::GameLift::Server::InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine -AnywhereFlotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode erfordert explizite Serverparameter, um die Kommunikation zwischen dem Server und dem Amazon- GameLift Service einzurichten.

Syntax

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

Parameter

ServerParameters

Um einen Spieleserver auf einer Amazon GameLift Anywhere-Flotte zu initialisieren, erstellen Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL des , der für die Verbindung mit Ihrem Spieleserver WebSocket verwendet wird.
- Die ID des Prozesses, der zum Hosten Ihres Spieleservers verwendet wird.
- Die ID der Datenverarbeitung, die Ihre Spielserverprozesse hostet.
- Die ID der Amazon- GameLift Flotte, die Ihre Amazon GameLift Anywhere-Datenverarbeitung enthält.
- Das von der Amazon- GameLift Operation generierte Autorisierungstoken.

Rückgabewert

Gibt ein the section called “InitSDKOutcome” Objekt zurück, das angibt, ob der Serverprozess bereit ist, aufzurufenProcessReady().

> **Note**
>
> Wenn Aufrufe von für Spiele-Builds fehlschlagen, die in Anywhere-Flotten bereitgestellt `InitSDK()` werden, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wird. Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn für eine neue Flotte bereit.

Beispiel

Amazon GameLift Anywhere-Beispiel

```
//Define the server parameters
std::string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
std::string processId = "PID1234";
std::string fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
std::string hostId = "HardwareAnywhere";
std::string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
Aws::GameLift::Server::Model::ServerParameters serverParameters =
  Aws::GameLift::Server::Model::ServerParameters(webSocketUrl, authToken, fleetId,
 hostId, processId);

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
 Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift , dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufruf von aufInitSDK(). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Parameter

processParameters

Ein ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen von Rückrufmethoden, die im Spielservercode implementiert sind, die der Amazon-GameLift Service aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen für Spielsitzungen spezifischen Dateien, die Amazon GameLift erfassen und speichern soll.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des ProcessReady()-Aufrufs und der Delegate-Funktion.

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");        // Example of a log file written by the
 game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
  Aws::GameLift::Server::ProcessParameters(
    std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
    std::bind(&Server::onProcessTerminate, this),
    std::bind(&Server::OnHealthCheck, this),
    std::bind(&Server::OnUpdateGameSession, this),
    listenPort,
    Aws::GameLift::Server::LogParameters(logPaths)
  );

Aws::GameLift::GenericOutcome outcome =
  Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
  // game-specific tasks when starting a new game session, such as loading map
  GenericOutcome outcome =
    Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
  // game-specific tasks required to gracefully shut down a game session,
  // such as notifying players, preserving game state data, and other cleanup
  GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
```

```
}

bool Server::onHealthCheck()
{
  bool health;
  // complete health evaluation within 60 seconds and set health
  return health;
}
```

ProcessReadyAsync()

Benachrichtigt den Amazon- GameLift Service, dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Diese Methode sollte aufgerufen werden, nachdem der Serverprozess bereit ist, eine Spielsitzung zu hosten. Die Parameter geben die Callback-Funktionsnamen an GameLift , die Amazon unter bestimmten Umständen aufrufen kann. Diese Funktionen müssen im Spiel-Servercode implementiert sein.

Dieser Aufruf ist asynchroner. Wenn Sie einen synchronen Aufruf durchführen möchten, verwenden Sie ProcessReady(). Weitere Details finden Sie unter Initialisieren Sie den Serverprozess.

Syntax

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameter

processParameters

Ein ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen von Rückrufmethoden, die im Spielservercode implementiert sind, die der Amazon-GameLift Service aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen für Spielsitzungen spezifischen Dateien, die Amazon GameLift erfassen und speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");        // This is an example of a log file
 written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
  Aws::GameLift::Server::ProcessParameters(std::bind(&Server::onStartGameSession, this,
 std::placeholders::_1),
  std::bind(&Server::onProcessTerminate, this), std::bind(&Server::OnHealthCheck,
 this),
  std::bind(&Server::OnUpdateGameSession, this), listenPort,
 Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
  Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
  // game-specific tasks when starting a new game session, such as loading map
  GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
  // game-specific tasks required to gracefully shut down a game session,
  // such as notifying players, preserving game state data, and other cleanup
  GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
  // perform health evaluation and complete within 60 seconds
  return health;
```

```
}
```

ProcessEnding()

Benachrichtigt Amazon GameLift , dass der Serverprozess beendet wird. Rufen Sie diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich Herunterfahren der aktiven Spielsitzung) und vor dem Beenden des Prozesses auf. Abhängig vom Ergebnis von wird `ProcessEnding()`der Prozess erfolgreich (0) oder mit einem Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntax

```
Aws::GameLift::GenericOutcome processEndingOutcome =
 Aws::GameLift::Server::ProcessEnding();
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden `ProcessEnding()` und aufgerufen, `Destroy()` bevor der Serverprozess mit einem Erfolgs- oder Fehlerbeendigungscode beendet wird.

```
Aws::GameLift::GenericOutcome processEndingOutcome =
 Aws::GameLift::Server::ProcessEnding();
Aws::GameLift::Server::Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess()) {
  exit(0);
}
else {
  cout << "ProcessEnding() failed. Error: " <<
 processEndingOutcome.GetError().GetErrorMessage();
  exit(-1);
}
```

ActivateGameSession()

Benachrichtigt Amazon GameLift , dass der Serverprozess eine Spielsitzung aktiviert hat und jetzt bereit ist, Spielerverbindungen zu empfangen. Diese Aktion sollte nach der Initialisierung der gesamten Spielsitzung als Teil der `onStartGameSession()` Rückruffunktion aufgerufen werden.

Syntax

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =
 Aws::GameLift::Server::ActivateGameSession();
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel zeigt, dass als Teil der `onStartGameSession()` Delegierungsfunktion `ActivateGameSession()` aufgerufen wird.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
  // game-specific tasks when starting a new game session, such as loading map
  GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
GenericOutcome
 UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy
 newPlayerSessionPolicy);
```

Parameter

playerCreationSessionRichtlinie

Typ: EnumPlayerSessionCreationPolicy-Wert.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
Aws::GameLift::GenericOutcome outcome =

 Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei inaktiven Prozessen, die nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf eine zurückthe section called “GameLiftError”.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als the section called “AwsStringOutcome”-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Bei inaktiven Prozessen, die nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf `Success=True` und `GameSessionId=` zurück"".

Beispiel

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =
 Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess ergreift Maßnahmen, nachdem er einen `onProcessTerminate()` Rückruf von Amazon erhalten hat GameLift. Amazon GameLift ruft `onProcessTerminate()` aus folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht auf Amazon geantwortet hat GameLift.
- Beim Beenden der Instance während eines Herunterskalierungsereignisses.
- Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg gibt die Beendigungszeit als `AwsDateTimeOutcome` Objekt zurück. Der Wert ist die Beendigungszeit, ausgedrückt in verstrichenen Ticks seit `0001 00:00:00`. Der Datum-/Uhrzeitwert `2020-09-13 12:26:40 -000Z` entspricht beispielsweise `637355968000000000` Ticks. Wenn keine Beendigungszeit verfügbar ist, gibt eine Fehlermeldung zurück.

Wenn der Prozess keinen ProcessParameters.OnProcessTerminate()-Callback erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen zum Herunterfahren eines Serverprozesses finden Sie unter Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses.

Beispiel

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
 Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player mit der angegebenen Player-Sitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und validiert werden muss. Amazon GameLift überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Player-Sitzung validiert wurde, GameLift ändert Amazon den Status des Player-Slots von RESERVED in ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die die Validierung und Ablehnung ungültiger Player-Sitzungs-IDs beinhaltet.

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
 playerSessionId)
{
  Aws::GameLift::GenericOutcome connectOutcome =
 Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
  if(connectOutcome.IsSuccess())
  {
    connectionToSessionMap.emplace(connection, playerSessionId);
```

```
    connection.Accept();
  }
  else
  {
    connection.Reject(connectOutcome.GetError().GetMessage();
  }
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player die Verbindung zum Serverprozess getrennt hat. Als Reaktion GameLift ändert Amazon den Player-Slot in „Verfügbar“.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameter

`playerSessionId`

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =
 Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, die Einstellungen, Sitzungsmetadaten und Spielerdaten enthalten. Verwenden Sie diese Methode, um Informationen über Folgendes zu erhalten:

- Eine Einzelplayer-Sitzung
- Alle Spielersitzungen in einer Spielsitzung

- Alle Player-Sitzungen, die einer einzelnen Player-ID zugeordnet sind

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

Parameter

DescribePlayerSessionsRequest

Ein the section called “DescribePlayerSessionsRequest” Objekt, das beschreibt, welche Player-Sitzungen abgerufen werden sollen.

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein the section called “DescribePlayerSessionsOutcome”-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spielersitzungen angefordert, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und auf 10 setzen, GameLift gibt Amazon die ersten 10 Spielersitzungsdatensätze zurück, die der Anforderung entsprechen.

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G
request.SetLimit(10);
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
 Aws::GameLift::Server::DescribePlayerSessions(request);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Diese Aktion ist asynchron. Wenn neue Spieler abgeglichen werden, GameLift liefert Amazon aktualisierte Matchmaker-Daten mithilfe der Callback-Funktion `OnUpdateGameSession()`.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

Parameter

StartMatchBackfillRequest

Ein StartMatchBackfillRequest Objekt, das die folgenden Informationen kommuniziert:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Wenn keine ID angegeben wird, GameLift generiert Amazon eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert befindet sich in den Matchmaker-Daten der Spielsitzung.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein the section called “StartMatchBackfillOutcome” Objekt mit der Match-Backfill-Ticket-ID oder einen Fehler mit einer Fehlermeldung zurück.

Beispiel

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff");  // optional,
 autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"); //from the game
 session matchmaker data
```

```
startBackfillRequest.SetGameSessionArn("the game session ARN");        // can use
 GetGameSessionId()
startBackfillRequest.SetPlayers(players);                                  // from the
 game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
  Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
 Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
  // handle status messages
  // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anforderung ab. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameter

StopMatchBackfillRequest

Ein StopMatchBackfillRequest Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die Ticket-ID, die der Backfill-Anforderung zugewiesen ist.
- Der Matchmaker, an den die Backfill-Anforderung gesendet wurde.
- Die Spielsitzung, die der Backfill-Anforderung zugeordnet ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff");
stopBackfillRequest.SetGameSessionArn("the game session ARN"); // can use
 GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig");
// from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
  Aws::GameLift::Server::StopMatchBackfill(stopBackfillRequest);
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zum Verschlüsseln der Netzwerkverbindung zwischen Ihrer Amazon GameLift Anywhere-Rechenressource und Amazon verwendet wird GameLift. Sie können den Zertifikatpfad verwenden, wenn Sie Ihr Rechengerät bei einer Amazon GameLift Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter RegisterCompute.

Syntax

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

Rückgabewert

Gibt eine the section called “GetComputeCertificateOutcome” zurück.

Beispiel

```
Aws::GameLift::GetComputeCertificateOutcome certificate =
 Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

Ruft IAM-Rollenanmeldeinformationen ab, die Amazon GameLift zur Interaktion mit anderen autorisierenAWS-Services. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Syntax

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
 request);
```

Parameter

GetFleetRoleCredentialsRequest

Rückgabewert

Gibt ein the section called “GetFleetRoleCredentialsOutcome”-Objekt zurück.

Beispiel

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
 getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
 Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Dieses Beispiel zeigt die Verwendung des optionalen `RoleSessionName` Werts, um der Anmeldeinformationssitzung zu Prüfungszwecken einen Namen zuzuweisen. Wenn Sie keinen Rollensitzungsnamen angeben, wird der Standardwert „*`[fleet-id]`*-*`[host-id]`*“ verwendet.

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
 getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");
getFleetRoleCredentialsRequest.SetRoleSessionName("MyFleetRoleSession");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
 Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Zerstören ()

Entlastet das Amazon GameLift -Game-Server-SDK aus dem Speicher. Als bewährte Methode rufen Sie diese Methode nach `ProcessEnding()` und auf, bevor Sie den Prozess beenden. Wenn Sie

eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie `Destroy()` und auf, um neu `InitSDK()` zu initialisieren, bevor Sie Amazon benachrichtigen GameLift , dass der Prozess bereit ist, eine Spielsitzung mit zu hosten`ProcessReady()`.

Syntax

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameter

Es gibt keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
Aws::GameLift::GenericOutcome processEndingOutcome =
 Aws::GameLift::Server::ProcessEnding();
Aws::GameLift::Server::Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess()) {
  exit(0);
}
else {
  cout << "ProcessEnding() failed. Error: " <<
 processEndingOutcome.GetError().GetErrorMessage();
  exit(-1);
}
```

Amazon GameLift Server SDK (C++)-Referenz: Datentypen

Sie können diese Amazon GameLift -C++-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

> **Note**
>
> In diesem Thema wird die Amazon GameLift -C++-API beschrieben, die Sie verwenden können, wenn Sie mit der C++ Standard Library () erstellen`std`. Insbesondere gilt diese Dokumentation für Code, den Sie mit der `-DGAMELIFT_USE_STD=1` Option kompilieren.

Datentypen

- LogParameters
- ProcessParameters
- UpdateGameSession
- GameSession
- ServerParameters
- StartMatchBackfillRequest
- Player
- DescribePlayerSessionsRequest
- StopMatchBackfillRequest
- AttributeValue
- GetFleetRoleCredentialsRequest
- AwsLongOutcome
- AwsStringOutcome
- DescribePlayerSessionsOutcome
- DescribePlayerSessionsResult
- GenericOutcome
- GenericOutcomeCallable
- PlayerSession
- StartMatchBackfillOutcome
- StartMatchBackfillResult
- GetComputeCertificateOutcome
- GetComputeCertificateResult
- GetFleetRoleCredentialsOutcome
- GetFleetRoleCredentialsResult

- InitSDKOutcome
- GameLiftError
- Aufzählungen

LogParameters

Ein Objekt, das Dateien identifiziert, die während einer Spielsitzung generiert wurden und die Amazon nach dem Ende der Spielsitzung GameLift hochladen und speichern soll. Der Spieleserver stellt Amazon GameLift als Teil eines `ProcessParameters` Objekts in einem ProcessReady() Aufruf `LogParameters` zur Verfügung.

Eigenschaften	Beschreibung
LogPaths	Die Liste der Verzeichnispfade zu Spielserver-Protokolldateien GameLift , die Amazon für den zukünftigen Zugriff speichern soll. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Sie definieren Dateipfade und -namen auf Ihrem Spieleserver und speichern sie im Build-Verzeichnis des Root-Spiels. Die Protokollpfade müssen absolute Werte sein. Wenn Ihr Spiele-Build beispielsweise Spielsitzungsprotokolle in einem Pfad wie speichert`MyGame\sessionLogs\` , befindet sich der Pfad `c:\game\MyGame\sessionLogs` auf einer Windows-Instance. Typ: `std:vector<std::string>` Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die an Amazon GameLift in einem gesendet werdenProcessReady().

Eigenschaften	Beschreibung
LogParameters	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert werden. Amazon GameLift kopiert und speichert die Dateien für den zukünftigen Zugriff. Typ: `Aws::GameLift::Server:: LogParameters` Required: No
OnHealthCheck	Die Rückruffunktion, die Amazon GameLift aufruft, um einen Zustandsstatusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess gibt zurück`TRUE`, wenn er fehlerfrei ist, `FALSE` wenn er nicht fehlerfrei ist. Wenn keine Antwort zurückgegeben wird, GameLift zeichnet Amazon den Serverprozess als nicht fehlerfrei auf. Typ: `std::function<bool()> onHealthCheck` Required: No
OnProcessTerminate	Die Callback-Funktion, die Amazon GameLift aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 5 Minuten, bis der Serverprozess heruntergefahren und mit einem -ProcessEnding()Aufruf geantwortet hat, bevor der Serverprozess heruntergefahren wird.

	Typ: `std::function<void()> onProcessTerminate` Erforderlich: Ja
OnRefreshConnection	Der Name der Callback-Funktion, die Amazon GameLift aufruft, um die Verbindung mit dem Spieleserver zu aktualisieren. Typ: `void OnRefreshConnectionDelegate()` Erforderlich: Ja
OnStartGameSession	Die Callback-Funktion, die Amazon GameLift aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift ruft diese Funktion als Antwort auf eine Client-Anforderung aufCreateGameSession. Die Callback-Funktion übergibt ein GameSession Objekt, wie in der Amazon GameLift -API-Referenz definiert. Typ: `const std::function<void(Aws::GameLift::Model::GameSession)> onStartGameSession` Erforderlich: Ja

OnUpdateGameSession	Die Callback-Funktion, die Amazon GameLift aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anforderung verarbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Es übergibt ein GameSession Objekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID. Typ: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession` Required: No
Port	Die Portnummer, die der Serverprozess auf neue Player-Verbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden. Typ: `Integer` Erforderlich: Ja

UpdateGameSession

Dieser Datentyp wird auf ein Spielsitzungsobjekt aktualisiert, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, wenn Backfill verwendet wird, um Spielersitzungen in der Spielsitzung zu füllen.

Eigenschaften	Beschreibung
GameSession	Ein von der Amazon GameLift API definiertes GameSession Objekt. Das -GameSession Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: `Aws::GameLift::Server::GameSession` Erforderlich: Ja
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: `Aws::GameLift::Server::UpdateReason` Erforderlich: Ja
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: `std::string` Required: No

GameSession

Dieser Datentyp enthält Details zu einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Ein Spielsitzungs-ARN hat das folgende Format: `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` .

Eigenschaften	Beschreibung
	Typ: `std::string` Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: `std::string` Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung ausgeführt wird. Typ: `std::string` Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: `int` Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon- GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: `in` Required: No

Eigenschaften	Beschreibung
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon- GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnumme r. Typ: `std::string` Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitz ungseigenschaften, die als einzelner Zeichenfo lgenwert formatiert sind. Typ: `std::string` Required: No
MatchmakerData	Informationen über den Matchmaking-Prozess, der zum Erstellen der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält sie Daten zu allen Spielern, die dem Match zugewiese n sind, einschließlich Spielerattributen und Teamzuweisungen. Typ: `std::string` Required: No

Eigenschaften	Beschreibung
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel-Wert-Paare. Diese Eigenschaften werden mit einer Anfrage zum Starten einer neuen Spielsitzung übergeben. Typ: `std :: vector < GameProperty >` Required: No
DnsName	Die DNS-ID, die der Instance zugewiesen ist, die die Spielsitzung ausführt. Werte haben das folgende Format: • TLS-fähige Flotten: `<unique identifier>.<region identifier>.amazongamelift.com` . • Nicht TLS-fähige Flotten: `ec2-<unique identifier>.compute.amazonaws.com` . Wenn Sie eine Verbindung zu einer Spielsitzung herstellen, die auf einer TLS-fähigen Flotte ausgeführt wird, müssen Sie den DNS-Namen verwenden, nicht die IP-Adresse. Typ: `std::string` Required: No

ServerParameters

Informationen, die verwendet werden, um die Verbindung zwischen dem Spieleserver auf einer Amazon GameLift Anywhere-Flotte und dem Amazon- GameLift Service aufrechtzuerhalten. Diese Informationen werden verwendet, wenn neue Serverprozesse mit gestartet werdenInitSDK().

Verwenden Sie für Server, die auf von Amazon GameLift verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
webSocketUrl	Amazon `GameLiftServerSdkEndpoint` GameLift gibt zurück, wenn Sie `RegisterCompute` für eine Amazon GameLift Anywhere-Rechenressource verwenden. Typ: `std::string` Erforderlich: Ja
processId	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der Ihr Spiel hostet. Typ: `std::string` Erforderlich: Ja
hostId	Die `HostID` wird `ComputeName` verwendet, als Sie Ihre Datenverarbeitung registriert haben. Weitere Informationen finden Sie unter RegisterCompute. Typ: `std::string` Erforderlich: Ja
fleetId	Die eindeutige Kennung der Flotte, bei der die Datenverarbeitung registriert ist. Weitere Informationen finden Sie unter `RegisterCompute`. Typ: `std::string` Erforderlich: Ja

Eigenschaften	Beschreibung
authToken	Das von Amazon generierte Authentifizierungs token GameLift , das Ihren Server bei Amazon authentifiziert GameLift. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: `std::string` Erforderlich: Ja

StartMatchBackfillRequest

Informationen, die zum Erstellen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt diese Informationen Amazon GameLift in einem StartMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Kennung für Spielsitzungen. Die API-Operation gibt die Kennung im ARN-Format `GetGameSessionId` zurück. Typ: `std::string` Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung in Form eines ARN, die der Matchmaker für diese Anforderung verwenden soll. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: `std::string` Erforderlich: Ja

Eigenschaften	Beschreibung
Players	Eine Reihe von Daten, die alle Spieler darstelle n, die sich in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: `std::vector<Player>` Erforderlich: Ja
TicketId	Eine eindeutige Kennung für ein Matchmaki ng- oder Match-Backfill-Anforderungsticket. Wenn Sie keinen Wert angeben, GameLift generiert Amazon einen. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen. Typ: `std::string` Required: No

Player

Dieser Datentyp stellt einen Spieler im Matchmaking dar. Beim Starten einer Matchmaking-Anfrage hat ein Spieler eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift fügt Teaminformationen hinzu, nachdem ein Match durchgeführt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, die in Millisekunden ausgedrückt werden und die Latenz angeben, die ein Spieler erfährt, wenn er mit einem Standort verbunden ist.

Eigenschaften	Beschreibung
	Wenn diese Eigenschaft verwendet wird, wird der Spieler nur mit den aufgelisteten Standorten abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet, müssen die Spieler die zu vergleichende Latenz melden. Typ: `Dictionary<string,int>` Required: No
PlayerAttributes	Eine Sammlung von Schlüssel-Wert-Paaren, die Spielerinformationen für die Verwendung im Matchmaking enthalten. Spielerattributschlüssel müssen mit dem in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: `std::map<std::string,AttributeValue>` Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: `std::string` Required: No

Eigenschaften	Beschreibung
Team	Der Name des Teams, dem der Spieler in einem Spiel zugewiesen ist. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: `std::string` Required: No

DescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Player-Sitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem DescribePlayerSessions() Aufruf an Amazon bereit GameLift.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für Spielsitzungen. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitz ung anzufragen. Das ID-Format für Spielsitzungen ist `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. ist `GameSessionID` eine benutzerdefinierte ID-Zeichenfolge oder ein Typ: `std::string` Required: No
PlayerSessionId	Die eindeutige Kennung für eine Player-Si tzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Player-Sitzung anzufordern. Typ: `std::string`

Eigenschaften	Beschreibung
	Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter, um alle Player-Sitzungen für einen bestimmten Player anzufordern. Siehe Generieren Sie Spieler-IDs. Typ: `std::string` Required: No
PlayerSessionStatusFilter	Der Status der Player-Sitzung, nach dem Ergebnisse gefiltert werden sollen. Zu den möglichen Player-Sitzungsstatus gehören: • RESERVED – Die Player-Sitzungsanf orderung wurde empfangen, aber der Player hat keine Verbindung zum Serverprozess hergestellt oder wurde validiert. • ACTIVE – Der Player wurde vom Serverpro zess validiert und ist verbunden. • COMPLETED – Die Player-Verbindung wurde unterbrochen. • TIMEDOUT – Eine Player-Sitzungsanf orderung wurde empfangen, aber der Player hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeout-Limits (60 Sekunden) validiert. Typ: `std::string` Required: No

Eigenschaften	Beschreibung
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `std::string` Required: No
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `int` Required: No

StopMatchBackfillRequest

Informationen, die zum Abbrechen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt diese Informationen dem Amazon- GameLift Service in einem StopMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Spielsitzungs-ID der abgebroch enen Anfrage. Typ: `char[]` Required: No
MatchmakingConfigurationArn	Eine eindeutige Kennung des Matchmakers, an den diese Anforderung gesendet wurde. Typ: `char[]`

Eigenschaften	Beschreibung
	Required: No
TicketId	Eine eindeutige Kennung des zu stornierenden Backfill-Anforderungstickets. Typ: `char[]` Required: No

AttributeValue

Verwenden Sie diese Werte in Player Attribut-Schlüssel-Wert-Paaren. Mit diesem Objekt können Sie einen Attributwert mit einem der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenfolgen-Array oder Datenzuordnung. Jedes `AttributeValue` Objekt muss genau eine der verfügbaren Eigenschaften verwenden: `S`, `NSL`, oder `SDM`.

Eigenschaften	Beschreibung
AttrType	Gibt den Typ des Attributwerts an. Mögliche Attributwerttypen sind: • NONE • STRING • DOUBLE • STRING_LIST • STRING_DOUBLE_MAP Required: No
S	Stellt einen Zeichenfolgenattributwert dar. Typ: `std::string` Required: No
N	Stellt einen numerischen Attributwert dar.

Eigenschaften	Beschreibung
	Typ: `double` Required: No
SL	Stellt ein Array von Zeichenfolgenattributwerten dar. Typ: `std::vector<std::string>` Required: No
SDM	Stellt ein Wörterbuch mit Zeichenfolgenschlüsseln und doppelten Werten dar. Typ: `std::map<std::string, double>` Required: No

GetFleetRoleCredentialsRequest

Dieser Datentyp gibt dem Spieleserver eingeschränkten Zugriff auf Ihre anderen AWS Ressourcen. Weitere Informationen finden Sie unter Einrichten einer IAM-Servicerolle für Amazon GameLift.

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert. Typ: `std::string` Required: No
RoleSessionName	Der Rollensitzungsname, den Sie verwenden können, um eine AWS Security Token Service AssumeRole Sitzung eindeutig zu identifizieren. Dieser Name wird in Prüfungsprotokollen wie z. B. in offengelegt CloudTrail.

Eigenschaften	Beschreibung
	Typ: `std::string` Required: No

AwsLongOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `long` Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `long&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

AwsStringOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `std::string` Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `long&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

DescribePlayerSessionsOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion.

Eigenschaften	Beschreibung
	Typ: the section called “DescribePlayerSessionsResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

DescribePlayerSessionsResult

Eine Sammlung von Objekten, die Eigenschaften für jede Spielersitzung enthalten, die der Anforderung entspricht.

Eigenschaften	Beschreibung
NextToken	Ein Token, das den Beginn der nächsten sequentiellen Seite mit Ergebnissen angibt. Verwenden Sie das Token, das mit einem

Eigenschaften	Beschreibung
	vorherigen Aufruf dieses Vorgangs zurückgeg eben wird. Um am Anfang der Ergebnismenge zu beginnen, geben Sie keinen Wert an. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert. Typ: `std::string` Erforderlich: Ja
PlayerSessions	Typ: `IList<`the section called “PlayerSession” `>` Erforderlich:
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `std::string&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GenericOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GenericOutcomeCallable

Dieser Datentyp ist ein asynchrones generisches Ergebnis. Er besitzt die folgenden Eigenschaften:

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

PlayerSession

Dieser Datentyp stellt eine Spielersitzung dar, die Amazon an den Spieleserver GameLift übergibt. Weitere Informationen finden Sie unter PlayerSession.

Eigenschaften	Beschreibung
CreationTime	Typ: `long` Required: No
FleetId	Typ: `std::string` Required: No
GameSessionId	Typ: `std::string` Required: No
IpAddress	Typ: `std::string` Required: No
PlayerData	Typ: `std::string` Required: No
PlayerId	Typ: `std::string` Required: No
PlayerSessionId	Typ: `std::string` Required: No
Port	Typ: `int` Required: No
Status	Status der Spielersitzung für die Filterung der Ergebnisse. Wenn ein PlayerSessionId oder bereitgestellt PlayerId wird, PlayerSessionStatusFilter hat das keine Auswirkungen auf die Antwort. Typ: Eine `PlayerSessionStatus` Aufzählung. Die folgenden Werte sind möglich:

Eigenschaften	Beschreibung
	• ACTIVE • COMPLETED • NOT_SET • RESERVIERT • TIMEDOUT Required: No
TerminationTime	Typ: `long` Required: No
DnsName	Typ: `std::string` Required: No

StartMatchBackfillOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “StartMatchBackfillResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `StartMatchBackfillResult&&` Required: No

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

StartMatchBackfillResult

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
TicketId	Eine eindeutige Kennung für ein Matchmaking-Ticket. Wenn hier keine Ticket-ID angegeben ist, GameLift generiert Amazon eine in Form einer UUID. Verwenden Sie diese Kennung, um den Status des Match-Backfill-Tickets nachzuverfolgen und die Match-Ergebnisse abzurufen. Typ: `std::string` Required: No

GetComputeCertificateOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetComputeCertificateResult” Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `Aws::GameLift::Server::Model::GetComputeCertificateResult&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrer Datenverarbeitung und der Hostname der Datenverarbeitung.

Eigenschaften	Beschreibung
CertificatePath	Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine von Amazon

Eigenschaften	Beschreibung
	GameLift verwaltete Flotte verwenden, enthält dieser Pfad: • `certificate.pem` : Das Endbenutz erzertifikat. Die vollständige Zertifikatkette ist die Kombination von , die an dieses Zertifikat `certificateChain.pem` angehängt ist. • `certificateChain.pem` : Die Zertifika tkette, die das Stammzertifikat und Zwischenzertifikate enthält. • `rootCertificate.pem` : Das Stammzert ifikat. • `privateKey.pem` : Der private Schlüssel für das Endbenutzerzertifikat. Typ: `std::string` Required: No
ComputeName	Der Name Ihrer Rechenressource. Typ: `std::string` Required: No

GetFleetRoleCredentialsOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetFleetRoleCredentialsResult”

Eigenschaften	Beschreibung
	Required: No
ResultWithOwnership	Das Ergebnis der Aktion, umgewandelt in einen R-Wert, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GetFleetRoleCredentialsResult

Eigenschaften	Beschreibung
AccessKeyId	Die Zugriffsschlüssel-ID für die Authentifizierung und den Zugriff auf Ihre -AWSRessourcen. Typ: `string` Required: No

Eigenschaften	Beschreibung
AssumedRoleId	Die ID des Benutzers, zu dem die Servicerolle gehört. Typ: `string` Required: No
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört. Typ: `string` Required: No
Ablauf	Die Zeit bis zum Ablauf Ihrer Sitzungsa nmeldeinformationen. Typ: `DateTime` Required: No
SecretAccessKey	Die ID des geheimen Zugriffsschlüssels für die Authentifizierung. Typ: `string` Required: No
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren -AWSRessourcen interagiert. Typ: `string` Required: No

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

InitSDKOutcome

> **Note**
>
> `InitSDKOutcome` wird nur zurückgegeben, wenn Sie das SDK mit dem `-std`Flag erstellen. Wenn Sie mit dem `-nostd`Flag erstellen, the section called “GenericOutcome” wird stattdessen zurückgegeben.

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GameLiftError

Eigenschaften	Beschreibung
ErrorType	Der Fehlertyp. Typ: Eine `GameLiftErrorType` Aufzählung von . ??? Required: No
ErrorName	Der Name des Fehlers. Typ: `std::string` Required: No
ErrorMessage	Die Fehlermeldung. Typ: `std::string` Required: No

Aufzählungen

Für das Amazon GameLift Server SDK (C++) definierte Enums sind wie folgt definiert:

GameLiftErrorType

Zeichenfolgenwert, der den Fehlertyp angibt. Gültige Werte sind:

- BAD_REQUEST_EXCEPTION
- GAMESESSION_ID_NOT_SET – Die Spielsitzungs-ID wurde nicht festgelegt.
- INTERNAL_SERVICE_AUSNAHME
- LOCAL_CONNECTION_FAILED – Die lokale Verbindung zu Amazon GameLift ist fehlgeschlagen.
- NETWORK_NOT_INITIALIZED – Das Netzwerk wurde nicht initialisiert.
- SERVICE_CALL_FAILED – Ein Aufruf an einen -AWSService ist fehlgeschlagen.
- WEBSOCKET_CONNECT_FAILURE

- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- ALREADY_INITIALIZED – Der Amazon GameLift Server oder Client wurde bereits mit Initialize() initialisiert.
- FLEET_MISMATCH – Die Zielflotte stimmt nicht mit der Flotte einer gameSession oder playerSession überein.
- GAMELIFT_CLIENT_NOT_INITIALIZED – Der Amazon- GameLift Client wurde nicht initialisiert.
- GAMELIFT_SERVER_NOT_INITIALIZED – Der Amazon- GameLift Server wurde nicht initialisiert.
- GAME_SESSION_ENDED_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung beendet wurde.
- GAME_SESSION_NOT_READY – Die Amazon GameLift Server-Spielsitzung wurde nicht aktiviert.
- GAME_SESSION_READY_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist.
- INITIALIZATION_MISMATCH – Eine Client-Methode wurde nach Server::Initialize() oder umgekehrt aufgerufen.
- NOT_INITIALIZED – Der Amazon GameLift Server oder Client wurde nicht mit Initialize() initialisiert.
- NO_TARGET_ALIASID_SET – Eine Ziel-aliasId wurde nicht festgelegt.
- NO_TARGET_FLEET_SET – Eine Zielflotte wurde nicht festgelegt.
- PROCESS_ENDING_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess endet.
- PROCESS_NOT_ACTIVE – Der Serverprozess ist noch nicht aktiv, nicht an ein gebunden GameSessionund kann weder akzeptieren noch verarbeiten PlayerSessions.
- PROCESS_NOT_READY – Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- PROCESS_READY_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess bereit ist.
- SDK_VERSION_DETECTION_FAILED – SDK-Versionserkennung fehlgeschlagen.
- STX_CALL_FAILED – Ein Aufruf der XStx-Server-Backend-Komponente ist fehlgeschlagen.

- STX_INITIALIZATION_FAILED – Die XStx-Server-Backend-Komponente konnte nicht initialisiert werden.
- UNEXPECTED_SpeedER_SESSION – Der Server ist auf eine nicht registrierte Player-Sitzung gestoßen.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE – Wiederholbarer Fehler beim Senden einer Nachricht an den GameLift Service WebSocket.
- WEBSOCKET_SEND_MESSAGE_FAILURE – Es konnte keine Nachricht an den GameLift Service gesendet werden WebSocket.
- MATCH_BACKFILL_REQUEST_VALIDATION – Die Validierung der Anforderung ist fehlgeschlagen.
- SpeedER_SESSION_REQUEST_VALIDATION – Die Validierung der Anforderung ist fehlgeschlagen.

PlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.
- NOT_SET – Die Spielsitzung ist nicht so eingestellt, dass neue Spielersitzungen akzeptiert oder abgelehnt werden.

Referenz zum Amazon GameLift C++-Server-SDK 3.x

Sie können diese Referenz zum Amazon GameLift C++-Server-SDK 3.x verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Referenz zum Amazon GameLift Server SDK (C++): Aktionen
- Referenz zum Amazon GameLift Server SDK (C++): Datentypen

Referenz zum Amazon GameLift Server SDK (C++): Aktionen

Sie können diese Amazon GameLift C++-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereitenGameLift. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Aktionen

- AcceptPlayerSession()
- ActivateGameSession()
- DescribePlayerSessions()
- GetGameSessionId()
- GetInstanceCertificate()
- GetSdkVersion()
- GetTerminationTime()
- InitSDK()
- ProcessEnding()
- ProcessReady()
- ProcessReadyAsync()
- RemovePlayerSession()
- StartMatchBackfill()
- StopMatchBackfill()
- TerminateGameSession()
- UpdatePlayerSessionCreationPolicy()
- Zerstören ()

AcceptPlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spielersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Überprüfung benötigt. Amazon GameLift überprüft, ob die Spielersitzungs-ID gültig ist, d. h., ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung GameLift ändert Amazon den Status des Spielerslots von RESERVED auf ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

Parameter

playerSessionId

Eindeutige ID, die vom GameLift Amazon-Dienst als Antwort auf einen Aufruf der GameLift Amazon-API-Aktion des AWS SDK ausgegeben wurde CreatePlayerSession. Der Spielclient verweist auf diese ID, wenn er sich mit dem Serverprozess verbindet.

Typ: std: :string

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht eine Funktion für die Verarbeitung einer Verbindungsanforderung, einschließlich dem Validieren und Abweisen ungültiger Spielersitzungs-IDs.

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
 playerSessionId){
    Aws::GameLift::GenericOutcome connectOutcome =
        Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage();
    }
}
```

ActivateGameSession()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess eine Spielsitzung gestartet hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion muss im Rahmen der `onStartGameSession()`-Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
GenericOutcome ActivateGameSession();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird `ActivateGameSession()` als Teil der `onStartGameSession()`-Callback-Funktion aufgerufen.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
   // game-specific tasks when starting a new game session, such as loading map
   GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();
}
```

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions (
```

```
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest
&describePlayerSessionsRequest);
```

Parameter

describePlayerSessionsAnfrage

Ein DescribePlayerSessionsRequest-Objekt, mit dem beschrieben wird, welche Spielersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein `DescribePlayerSessionsOutcome`-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem GameLift PlayerSessionAmazon-API-Datentyp des AWS SDK identisch ist.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spielersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn Sie den `Limit` Wert weglassen `NextToken` und auf 10 setzen, GameLift gibt Amazon die Datensätze der ersten 10 Spielersitzungen zurück, die der Anfrage entsprechen.

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G
request.SetLimit(10);
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

GetGameSessionId()

Ruft den eindeutigen Bezeichner der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist. Der Bezeichner wird im ARN-Format zurückgegeben: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`.

Bei inaktiven Prozessen, die noch nicht mit einer Spielsitzung aktiviert wurden, gibt der Aufruf `Success = True` und `GameSessionId = ""` (eine leere Zeichenfolge) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `AwsStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

Ruft den Speicherort eines PEM-kodierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Managergeneriert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, wobei die Zertifikatskonfiguration auf GENERATED gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz

gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen über das Zertifikat und die Zertifikatskettendaten finden Sie unter GetCertificateAntwortelemente in der AWS Certificate Manager API-Referenz.

Beispiel

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

Gibt die aktuelle Versionsnummer des verwendeten SDKs zurück.

Syntax

```
AwsStringOutcome GetSdkVersion();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `AwsStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5"). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
    Aws::GameLift::Server::GetSdkVersion();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf vom GameLift Amazon-Dienst erhalten hat. Amazon GameLift kann aus den folgenden `onProcessTerminate()` Gründen anrufen: (1) wenn der Serverprozess

einen schlechten Zustand gemeldet hat oder Amazon nicht reagiert hatGameLift, (2) wenn die Instance während eines Scale-Down-Ereignisses beendet wird oder (3) wenn eine Instance aufgrund einer Spot-Unterbrechung beendet wird.

Wenn der Prozess einen onProcessTerminate() Rückruf erhalten hat, ist der zurückgegebene Wert die geschätzte Abbruchzeit. Wenn der Prozess keinen onProcessTerminate() Rückruf erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen über das Herunterfahren eines Serverprozesses.

Syntax

```
AwsLongOutcome GetTerminationTime();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird die Abbruchzeit als AwsLongOutcome Objekt zurückgegeben. Der Wert ist die Abbruchzeit, ausgedrückt in verstrichenen Ticks seit 0001 00:00:00. Beispielsweise entspricht der Datums-/Uhrzeitwert 2020-09-13 12:26:40 -000Z 637355968000000000 Ticks. Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Initialisiert das Amazon GameLift SDK. Diese Methode sollte beim Start aufgerufen werden, bevor eine andere Initialisierung GameLift im Zusammenhang mit Amazon erfolgt.

Syntax

```
InitSDKOutcome InitSDK();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein InitSdkOutcome Objekt zurückgegeben, das angibt, dass der Serverprozess zum Aufrufen bereit istProcessReady().

Beispiel

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess heruntergefahren wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem Beendigungscode 0 beendet werden. Andere Beendigungscodes sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Sobald die Methode mit einem Code von 0 beendet wird, können Sie den Prozess mit einem erfolgreichen Exit-Code beenden. Sie können den Vorgang auch mit einem Fehlercode beenden. Wenn Sie den Vorgang mit einem Fehlercode beenden, zeigt das Flottenereignis an, dass der Prozess ungewöhnlich beendet wurde (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Syntax

```
GenericOutcome ProcessEnding();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
```

```
if (outcome.Success)
    exit(0);  // exit with success
// otherwise, exit with error code
exit(errorCode);
```

ProcessReady()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess bereit ist, Spielesitzungen zu hosten. Rufen Sie diese Methode auf, nachdem Sie die Einrichtungsaufgaben erfolgreich aufgerufen InitSDK() und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Dieser Aufruf ist ein synchron. Wenn Sie einen asynchronen Aufruf durchführen möchten, verwenden Sie ProcessReadyAsync(). Weitere Details finden Sie unter Initialisieren Sie den Serverprozess.

Syntax

```
GenericOutcome ProcessReady(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameter

processParameters

Ein ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spieleservercode implementierten Callback-Methoden, die der GameLift Amazon-Dienst zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen spielsitzungsspezifischen Dateien, die Amazon erfassen und GameLift speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des ProcessReady()-Aufrufs und der Callback-Funktion.

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");        // Example of a log file written by the
 game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);

int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
 Aws::GameLift::Server::ProcessParameters(
    std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
    std::bind(&Server::onProcessTerminate, this),
    std::bind(&Server::OnHealthCheck, this),
    std::bind(&Server::OnUpdateGameSession, this),
    listenPort,
    Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcome outcome =
   Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
   // game-specific tasks when starting a new game session, such as loading map
   GenericOutcome outcome =
       Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
   // game-specific tasks required to gracefully shut down a game session,
   // such as notifying players, preserving game state data, and other cleanup
   GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
```

```
    return health;
}
```

ProcessReadyAsync()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess bereit ist, Spielesitzungen zu hosten. Diese Methode sollte aufgerufen werden, sobald der Serverprozess zum Hosten einer Spielsitzung bereit ist. Die Parameter geben die Namen der Callback-Funktionen an, die Amazon unter bestimmten Umständen aufrufen GameLift soll. Diese Funktionen müssen im Spiel-Servercode implementiert sein.

Dieser Aufruf ist asynchroner. Wenn Sie einen synchronen Aufruf durchführen möchten, verwenden Sie ProcessReady(). Weitere Details finden Sie unter Initialisieren Sie den Serverprozess.

Syntax

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameter

processParameters

Ein ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spieleservercode implementierten Callback-Methoden, die der GameLift Amazon-Dienst zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen spielsitzungsspezifischen Dateien, die Amazon erfassen und GameLift speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
// Set parameters and call ProcessReady
```

```
std::string serverLog("serverOut.log");        // This is an example of a log file
 written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);

int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
 Aws::GameLift::Server::ProcessParameters(
    std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
    std::bind(&Server::onProcessTerminate, this),
    std::bind(&Server::OnHealthCheck, this),
    std::bind(&Server::OnUpdateGameSession, this),
    listenPort,
    Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
   Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
   // game-specific tasks when starting a new game session, such as loading map
   GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
   // game-specific tasks required to gracefully shut down a game session,
   // such as notifying players, preserving game state data, and other cleanup
   GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}
```

RemovePlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spieler-Sitzungs-ID die Verbindung zum Serverprozess getrennt hat. Als Reaktion darauf GameLift ändert Amazon den Spielerplatz auf verfügbar, sodass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);
```

Parameter

playerSessionId

Eindeutige ID, die vom GameLift Amazon-Dienst als Antwort auf einen Aufruf der GameLift Amazon-API-Aktion des AWS SDK ausgegeben wurde CreatePlayerSession. Der Spielclient verweist auf diese ID, wenn er sich mit dem Serverprozess verbindet.

Typ: std: :string

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion StartMatchBackfill(). Mit dieser Aktion können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der GameLift Amazon-Dienst aktualisierte Matchmaker-Daten, indem er die Callback-Funktion aufruft. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest
 &startBackfillRequest);
```

Parameter

StartMatchBackfillRequest

Ein StartMatchBackfillRequest-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben GameLift wird, generiert Amazon automatisch eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.
- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Gibt ein StartMatchBackfillOutcome Objekt mit dem Match-Backfill-Ticket oder einen Fehler mit einer Fehlermeldung zurück. Der Ticketstatus kann mit der AWS SDK-Aktion DescribeMatchmaking() verfolgt werden.

Beispiel

```
// Build a backfill request
```

```
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("a ticket ID");
  //optional, autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration
 ARN"); //from the game session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");
  // can use GetGameSessionId()
startBackfillRequest.SetPlayers(players);
    //from the game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
 Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
   // handle status messages
   // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

Bricht eine aktive mit dem Befehl StartMatchBackfill() erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion StopMatchmaking(). Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Syntax

```
GenericOutcome StopMatchBackfill (
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);
```

Parameter

StopMatchBackfillRequest

Ein StopMatchBackfillRequest-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde

- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("the ticket ID");
stopBackfillRequest.SetGameSessionArn("the game session ARN");
  // can use GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration ARN");
  // from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
    Aws::GameLift::Server::StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess ProcessEnding() nach dem Ende einer Spielsitzung aufgerufen werden.

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Es sollte erst aufgerufen werden, nachdem das Verfahren zur Beendigung der Spielsitzung abgeschlossen ist, da es Amazon signalisiertGameLift, dass der Serverprozess sofort für das Hosten einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach Beendigung der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen an, ProcessEnding() um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
GenericOutcome TerminateGameSession();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. Siehe auch die AWS SDK-Aktion UpdateGameSession().

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

Parameter

newPlayerSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Typ: Aws:GameLift: :Modell:: PlayerSessionCreationPolicy enum. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
Aws::GameLift::GenericOutcome outcome =
 Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Zerstören ()

Räumt den von initSDK () zugewiesenen Speicher während der Gameserver-Initialisierung auf. Verwende diese Methode, nachdem du einen Gameserverprozess beendet hast, um zu vermeiden, dass Serverspeicher verschwendet wird.

Syntax

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameter

Es gibt keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird der von initSDK zugewiesene Speicher bereinigt, nachdem ein Gameserverprozess beendet wurde.

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {
  Aws::GameLift::Server::Destroy();
  exit(0);
}
```

Referenz zum Amazon GameLift Server SDK (C++): Datentypen

Sie können diese Amazon GameLift C++-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereitenGameLift. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Diese API wird in `GameLiftServerAPI.h`, `LogParameters.h` und `ProcessParameters.h` definiert.

- Aktionen
- Datentypen

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielersitzung(en) abgerufen werden sollen. Sie können ihn wie folgt verwenden:

- Geben Sie eine PlayerSessionId an, um eine bestimmte Spielersitzung anzufordern.
- Geben Sie ein GameSessionId an, um alle Spielersitzungen in der angegebenen Spielsitzung anzufordern.
- Geben Sie einPlayerId, um alle Spielersitzungen für den angegebenen Spieler anzufordern.

Verwenden Sie für große Sammlungen von Spielersitzungen die Paginierungsparameter, um Ergebnisse in aufeinander folgenden Blöcken abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von **<**ID-String**>** ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Required: No

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit NextToken, um Ergebnisse als Satz aufeinanderfolgender Seiten zu erhalten. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Required: No

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Required: No

PlayerId

Eindeutiger Bezeichner für den Spieler. Player-IDs sind vom Entwickler definiert. Siehe Generieren Sie Spieler-IDs.

Typ: Zeichenfolge

Required: No

PlayerSessionId

Eindeutiger Bezeichner für eine Spielersitzung.

Typ: Zeichenfolge

Required: No

PlayerSessionStatusFilter

Status der Spielersitzung für die Filterung der Ergebnisse. Mögliche Spielersitzungsstatus sind u. a.:

- RESERVED – Die Anfrage nach einer Spielersitzung ist eingegangen, der Spieler ist jedoch noch nicht mit dem Serverprozess verbunden und/oder noch nicht validiert.
- ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.
- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT – Eine Anfrage nach einer Spielersitzung ist eingegangen, der Spieler hat jedoch keine Verbindung hergestellt und/oder wurde nicht innerhalb des Timeout-Zeitraums (60 Sekunden) validiert.

Typ: Zeichenfolge

Required: No

LogParameters

Dieser Datentyp wird verwendet, um zu ermitteln, welche Dateien während einer Spielsitzung generiert wurden und die Amazon GameLift nach Ende der Spielsitzung hochladen und speichern soll. Diese Informationen werden dem GameLift Amazon-Dienst in einem ProcessReady() Anruf mitgeteilt.

Inhalt

logPaths

Verzeichnispfade zu den Logdateien des Spielservers, die Amazon für den zukünftigen GameLift Zugriff speichern soll. Diese Dateien werden während jeder Spielsitzung generiert. Dateipfade und Namen sind in Ihrem Spieleserver definiert und im Root-Spiel-Build-Verzeichnis gespeichert. Die Log-Pfade müssen absolut sein. Zum Beispiel: Wenn Ihr Spiel-Build Spielsitzungsprotokolle in einem Pfad wie `MyGame\sessionlogs\` speichert, ist der Protokollpfad `c:\game\MyGame\sessionLogs` (auf einer Windows-Instance) oder `/local/game/MyGame/sessionLogs` (auf einer Linux-Instance).

Typ: std:vector<std::string>

Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die bei einem ProcessReady() Anruf an den GameLift Amazon-Dienst gesendet werden.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: Aws::GameLift: :Server:: LogParameters

Required: No

onStartGameSitzung

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift ruft diese Funktion als Antwort auf die Kundenanfrage auf CreateGameSession. Die Callback-Funktion übergibt ein GameSessionObjekt (definiert in der Amazon GameLift Service API-Referenz).

Typ: `const std::function<void(Aws::GameLift::Model::GameSession)> onStartGameSession`

Erforderlich: Ja

onProcessTerminate

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon fünf Minuten, bis der Serverprozess heruntergefahren ist, und reagiert mit einem ProcessEnding() Anruf. Wird keine Antwort empfangen, wird der Server-Prozess beendet.

Typ: `std::function<void()> onProcessTerminate`

Required: No

onHealthCheck

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 60 Sekunden auf eine Antwort. Wenn keine Antwort empfangen wird, zeichnet Amazon den Serverprozess als fehlerhaft auf.

Typ: `std::function<bool()> onHealthCheck`

Required: No

onUpdateGameSitzung

Name der Callback-Funktion, die der GameLift Amazon-Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion

auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Es übergibt ein GameSessionObjekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID.

Typ: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession`

Required: No

StartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden in einem StartMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die API-Aktion GetGameSessionId() gibt den Bezeichner im ARN-Format zurück.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker. Erfahren Sie mehr über Matchmaker-Daten in Word mit Matchmaker-Daten.

Typ: Zeichenfolge

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon

GameLift API-Referenzhandbuch. Um Spielerattribute, IDs und Teamzuweisungen zu finden, suchen Sie im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Typ: std:vectorhttps://docs.aws.amazon.com/gamelift/latest/apireference/API_Player.html <player>

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben GameLift wird, generiert Amazon einen in Form einer UUID. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen.

Typ: Zeichenfolge

Required: No

StopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abzubrechen. Die Informationen werden in einem StopMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: Zeichenfolge

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzubrechenden Tickets für die Backfill-Anforderung.

Typ: Zeichenfolge

Erforderlich: Ja

Amazon GameLift Server-SDK-Referenz für C#

Sie können diese Amazon GameLift C#-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity
- Amazon GameLift Server SDK 4.x-Referenz für C#

Amazon GameLift Server SDK 5.x-Referenz für C# und Unity

Sie können diese Referenz zum Amazon GameLift C#-Server-SDK 5.x verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten. GameLift Einzelheiten zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur Verwendung des C#-Server-SDK-Plug-ins für Unity finden Sie unterIntegrieren Sie Amazon GameLift in ein Unity-Projekt. Das Amazon GameLift Server SDK 5.x für C# unterstützt.NET 4.6 und.NET 6.

Themen

- Amazon GameLift Server SDK-Referenz für C# und Unity: Aktionen
- Amazon GameLift Server-SDK-Referenz für C# und Unity: Datentypen

Amazon GameLift Server SDK-Referenz für C# und Unity: Aktionen

Diese Amazon GameLift -C#-Server-SDK-Referenz hilft Ihnen dabei, Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur

Verwendung des C#-Server-SDK-Plugins für Unity finden Sie unter Integrieren Sie Amazon GameLift in ein Unity-Projekt.

Aktionen

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- Zerstören ()

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
AwsStringOutcome GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als the section called "AwsStringOutcome"-Objekt zurück. Die zurückgegebene Zeichenfolge enthält die Versionsnummer

(Beispiel `5.0.0`). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine verwaltete EC2-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Server und dem Amazon GameLift-Service einzurichten.

Syntax

```
GenericOutcome InitSDK();
```

Rückgabewert

Bei Erfolg gibt ein - InitSdkOutcome Objekt zurück, um anzuzeigen, dass der Serverprozess bereit ist, aufzurufenProcessReady().

Beispiel

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine -AnywhereFlotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode erfordert explizite Serverparameter, um die Kommunikation zwischen dem Server und dem Amazon- GameLift Service einzurichten.

Syntax

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

Parameter

ServerParameters

Um einen Spieleserver auf einer Amazon GameLift Anywhere-Flotte zu initialisieren, erstellen Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL des , der für die Verbindung mit Ihrem Spieleserver WebSocket verwendet wird.
- Die ID des Prozesses, der zum Hosten Ihres Spieleservers verwendet wird.
- Die ID der Datenverarbeitung, die Ihre Spielserverprozesse hostet.
- Die ID der Amazon- GameLift Flotte, die Ihre Amazon GameLift Anywhere-Datenverarbeitung enthält.
- Das von der Amazon- GameLift Operation generierte Autorisierungstoken.

Rückgabewert

Bei Erfolg gibt ein - InitSdkOutcome Objekt zurück, um anzuzeigen, dass der Serverprozess bereit ist, aufzurufenProcessReady().

> **Note**
>
> Wenn Aufrufe von für Spiele-Builds fehlschlagen, die in Anywhere-Flotten bereitgestellt `InitSDK()` werden, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wird. Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn für eine neue Flotte bereit.

Beispiel

```
//Define the server parameters
string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
string processId = "PID1234";
string fleetId = "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
string hostId = "HardwareAnywhere";
string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
ServerParameters serverParameters =
```

```
  new ServerParameters(webSocketUrl, processId, hostId, fleetId, authToken);

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift , dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufruf von aufInitSDK(). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameter

ProcessParameters

Ein `ProcessParameters` Objekt enthält Informationen über den Serverprozess.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht sowohl die Implementierung der -Methode als auch die Implementierung der Delegierungsfunktion.

```
// Set parameters and call ProcessReady
ProcessParameters processParams = new ProcessParameters(
  this.OnStartGameSession,
  this.OnProcessTerminate,
  this.OnHealthCheck,
  this.OnUpdateGameSession,
  port,
  new LogParameters(new List<string>()
  // Examples of log and error files written by the game server
```

```
  {
    "C:\\game\\logs",
    "C:\\game\\error"
  })
);
GenericOutcome processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

ProcessEnding()

Benachrichtigt Amazon GameLift , dass der Serverprozess beendet wird. Rufen Sie diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich Herunterfahren der aktiven Spielsitzung) und vor dem Beenden des Prozesses auf. Abhängig vom Ergebnis von wird `ProcessEnding()`der Prozess erfolgreich (0) oder mit einem Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntax

```
GenericOutcome ProcessEnding()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden `ProcessEnding()` und aufgerufen, `Destroy()` bevor der Serverprozess mit einem Erfolgs- oder Fehlerbeendigungscode beendet wird.

```
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
GameLiftServerAPI.Destroy();

if (processEndingOutcome.Success)
  {
    Environment.Exit(0);
  }
else
  {
    Console.WriteLine("ProcessEnding() failed. Error: " +
 processEndingOutcome.Error.ToString());
```

```
    Environment.Exit(-1);
  }
```

ActivateGameSession()

Benachrichtigt Amazon GameLift , dass der Serverprozess eine Spielsitzung aktiviert hat und jetzt Spielerverbindungen empfangen kann. Diese Aktion sollte nach der Initialisierung der gesamten Spielsitzung als Teil der onStartGameSession() Rückruffunktion aufgerufen werden.

Syntax

```
GenericOutcome ActivateGameSession()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird ActivateGameSession() als Teil der onStartGameSession()-Delegate-Funktion aufgerufen.

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

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
 playerSessionPolicy)
```

Parameter

playerSessionPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
GenericOutcome updatePlayerSessionPolicyOutcome =

 GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei inaktiven Prozessen, die nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf eine zurückthe section called “GameLiftError”.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als the section called “AwsStringOutcome”-Objekt zurück. Wenn nicht erfolgreich, wird eine Fehlermeldung zurückgegeben.“

Beispiel

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf von Amazon erhalten hat GameLift. Amazon GameLift ruft `onProcessTerminate()` aus folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht auf Amazon geantwortet hat GameLift.
- Beim Beenden der Instance während eines Herunterskalierungsereignisses.
- Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg gibt die Beendigungszeit als the section called “AwsDateTimeOutcome” Objekt zurück. Der Wert ist die Beendigungszeit, ausgedrückt in verstrichenen Ticks seit `0001 00:00:00`. Der Datum-/Uhrzeitwert `2020-09-13 12:26:40 -000Z` entspricht beispielsweise `637355968000000000` Ticks. Wenn keine Beendigungszeit verfügbar ist, gibt eine Fehlermeldung zurück.

Beispiel

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player mit der angegebenen Player-Sitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und validiert werden muss. Amazon GameLift überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Player-Sitzung validiert wurde, GameLift ändert Amazon den Status des Player-Slots von RESERVED in ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von ausgegeben wird GameLift , wenn eine neue Spielersitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht eine Funktion für die Verarbeitung einer Verbindungsanforderung, einschließlich dem Validieren und Abweisen ungültiger Spielersitzungs-IDs.

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerSessionId)
{
  GenericOutcome acceptPlayerSessionOutcome =
 GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
  if(acceptPlayerSessionOutcome.Success)
  {
    connectionToSessionMap.emplace(connection, playerSessionId);
    connection.Accept();
  }
  else
  {
    connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
  }
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player die Verbindung zum Serverprozess getrennt hat. Als Reaktion GameLift ändert Amazon den Player-Slot in „Verfügbar“.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
GenericOutcome removePlayerSessionOutcome =
 GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, die Einstellungen, Sitzungsmetadaten und Spielerdaten enthalten. Verwenden Sie diese Aktion, um für eine einzelne Spielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

Parameter

DescribePlayerSessionsRequest

Ein the section called “DescribePlayerSessionsRequest” Objekt, das beschreibt, welche Player-Sitzungen abgerufen werden sollen.

Rückgabewert

Bei Erfolg gibt ein the section called “DescribePlayerSessionsOutcome” Objekt zurück, das eine Reihe von Spielersitzungsobjekten enthält, die den Anforderungsparametern entsprechen.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spielersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und auf 10 setzen, GameLift gibt Amazon die ersten 10 Spielersitzungsdatensätze zurück, die der Anforderung entsprechen.

```
// Set request parameters
DescribePlayerSessionsRequest describePlayerSessionsRequest = new
 DescribePlayerSessionsRequest()
{
  GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for the
 current game session
  Limit = 10,
  PlayerSessionStatusFilter =
    PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
DescribePlayerSessionsOutcome describePlayerSessionsOutcome =
  GameLiftServerAPI.DescribePlayerSessions(describePlayerSessionsRequest);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Diese Aktion ist asynchron. Wenn neue Spieler abgeglichen werden, GameLift liefert Amazon aktualisierte Matchmaker-Daten mithilfe der Callback-Funktion `OnUpdateGameSession()`.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

Parameter

StartMatchBackfillRequest

Ein `StartMatchBackfillRequest` Objekt enthält Informationen über die Backfill-Anforderung.

Rückgabewert

Gibt ein the section called “StartMatchBackfillOutcome” Objekt mit der Match-Backfill-Ticket-ID oder einen Fehler mit einer Fehlermeldung zurück.

Beispiel

```
// Build a backfill request
StartMatchBackfillRequest startBackfillRequest = new StartMatchBackfillRequest()
{
  TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional
  MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
  GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    // gets ID for
 current game session
  MatchmakerData matchmakerData =
    MatchmakerData.FromJson(gameSession.MatchmakerData),  // gets matchmaker data for
 current players
  // get matchmakerData.Players
  // remove data for players who are no longer connected
  Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
StartMatchBackfillOutcome startBackfillOutcome =
 GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
  // game-specific tasks to prepare for the newly matched players and update matchmaker
 data as needed
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anforderung ab. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameter

StopMatchBackfillRequest

Ein `StopMatchBackfillRequest` Objekt, das Details zu dem von Ihnen gestoppten Matchmaking-Ticket enthält.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
// Set backfill stop request parameters
StopMatchBackfillRequest stopBackfillRequest = new StopMatchBackfillRequest(){
  TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional, if not provided one is
 autogenerated
  MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
  GameSessionId = GameLiftServerAPI.GetGameSessionId().Result    //gets the ID for the
 current game session
};
GenericOutcome stopBackfillOutcome =
 GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zum Verschlüsseln der Netzwerkverbindung zwischen dem Spieleserver und Ihrem Spieleclient verwendet wird. Sie können den Zertifikatpfad verwenden, wenn Sie Ihr Rechengerät bei einer Amazon GameLift Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter RegisterCompute.

Syntax

```
GetComputeCertificateOutcome GetComputeCertificate();
```

Rückgabewert

Gibt ein GetComputeCertificateResponse Objekt zurück, das Folgendes enthält:

- CertificatePath: Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine von Amazon GameLift verwaltete Flotte verwenden, enthält dieser Pfad:
  - `certificate.pem`: Das Endbenutzerzertifikat. Die vollständige Zertifikatkette ist die Kombination von , die an dieses Zertifikat `certificateChain.pem` angehängt ist.
  - `certificateChain.pem`: Die Zertifikatkette, die das Stammzertifikat und Zwischenzertifikate enthält.
  - `rootCertificate.pem`: Das Stammzertifikat.
  - `privateKey.pem`: Der private Schlüssel für das Endbenutzerzertifikat.
- ComputeName: Der Name Ihrer Rechenressource.

Beispiel

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
 GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

Ruft IAM-Rollen-Anmeldeinformationen ab, die Amazon GameLift zur Interaktion mit anderen autorisierenAWS-Services. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Syntax

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
 request);
```

Parameter

GetFleetRoleCredentialsRequest

Rollenanmeldeinformationen, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver erweitern.

Rückgabewert

Gibt ein the section called “GetFleetRoleCredentialsOutcome”-Objekt zurück.

Beispiel

```
// form the fleet credentials request
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new
 GetFleetRoleCredentialsRequest(){
  RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"
};
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =
 GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Zerstören ()

Entlastet das Amazon GameLift Game Server SDK aus dem Speicher. Als bewährte Methode rufen Sie diese Methode nach `ProcessEnding()` und auf, bevor Sie den Prozess beenden. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie `Destroy()` und auf, um sie neu `InitSDK()` zu initialisieren, bevor Sie Amazon benachrichtigen GameLift , dass der Prozess bereit ist, eine Spielsitzung mit zu hosten`ProcessReady()`.

Syntax

```
GenericOutcome Destroy()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
// Operations to end game sessions and the server process
```

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

// Shut down and destroy the instance of the GameLift Game Server SDK
GenericOutcome destroyOutcome = GameLiftServerAPI.Destroy();

// Exit the process with success or failure
if (processEndingOutcome.Success)
  {
    Environment.Exit(0);
  }
else
  {
    Console.WriteLine("ProcessEnding() failed. Error: " +
 processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
  }
```

Amazon GameLift Server-SDK-Referenz für C# und Unity: Datentypen

Diese Amazon GameLift C# Server-SDK-Referenz kann Ihnen helfen, Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur Verwendung des C#-Server-SDK-Plug-ins für Unity finden Sie unterIntegrieren Sie Amazon GameLift in ein Unity-Projekt.

Datentypen

- LogParameters
- ProcessParameters
- UpdateGameSession
- GameSession
- ServerParameters
- StartMatchBackfillRequest
- Player
- DescribePlayerSessionsRequest
- StopMatchBackfillRequest
- GetFleetRoleCredentialsRequest
- AttributeValue
- AwsStringOutcome

- GenericOutcome
- DescribePlayerSessionsOutcome
- DescribePlayerSessionsResult
- PlayerSession
- StartMatchBackfillOutcome
- StartMatchBackfillResult
- GetComputeCertificateOutcome
- GetComputeCertificateResult
- GetFleetRoleCredentialsOutcome
- GetFleetRoleCredentialsResult
- AwsDateTimeOutcome
- GameLiftError
- Aufzählungen

LogParameters

Verwenden Sie diesen Datentyp, um zu ermitteln, welche Dateien während einer Spielsitzung generiert wurden und die der Spielserver GameLift nach Ende der Spielsitzung auf Amazon hochladen soll. Der Gameserver kommuniziert `LogParameters to` Amazon GameLift in einem ProcessReady() Anruf.

Eigenschaften	Beschreibung
LogPaths	Die Liste der Verzeichnispfade zu den Logdateien des Spielservers, GameLift die Amazon für den zukünftigen Zugriff speichern soll. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Du definierst Dateipfade und Namen auf deinem Gameserve r und speicherst sie im Root-Game-Build-Ve rzeichnis. Die Log-Pfade müssen absolut sein. Wenn Ihr Game-Build beispielsweise Spielesitzungsprot okolle in einem Pfad wie speichert`MyGame\se`

	`ssionLogs\` , dann würde sich der Pfad `c:\game\MyGame\sessionLogs` auf einer Windows-Instance befinden. Typ: `List<String>` Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die bei einem ProcessReady() Anruf GameLift an Amazon gesendet wurden.

Eigenschaften	Beschreibung
LogParameters	Das Objekt mit einer Liste von Verzeichnispfaden zu Logdateien von Spielesitzungen. Typ: `Aws::GameLift::Server:: LogParameters` Required: Yes
OnHealthCheck	Der Name der Callback-Funktion, die Amazon GameLift aufruft, um einen Integritätsstatusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 60 Sekunden auf eine Antwort. Wenn keine Antwort eingeht, GameLift zeichnet Amazon den Serverprozess als fehlerhaft auf. Typ: `void OnHealthCheckDelegate()` Required: Yes
OnProcessTerminate	Der Name der Callback-Funktion, die Amazon GameLift aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach

	dem Aufruf dieser Funktion GameLift wartet Amazon fünf Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem ProcessEnding() Anruf, bevor der Serverpro zess heruntergefahren wird. Typ: `void OnProcessTerminate Delegate()` Required: Yes
OnStartGameSession	Der Name der Callback-Funktion, die Amazon GameLift aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift ruft diese Funktion als Antwort auf die Kundenanfrage auf CreateGameSession. Die Callback-Funktion verwendet ein in der GameLift Amazon-API-Referenz definiertes GameSessionObjekt. Typ: `void OnStartGameSession Delegate( GameSession )` Required: Yes
OnUpdateGameSession	Der Name der Callback-Funktion, die Amazon GameLift aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage verarbeitet wurde, um aktualisierte Matchmake r-Daten bereitzustellen. Es übergibt ein GameSessionObjekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Tic ket-ID. Typ: void OnUpdateGameSessionDelegate (UpdateGameSession) Required: No

Port	Die Portnummer, an der der Serverprozess auf neue Player-Verbindungen wartet. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnumme r ist in Spielsitzungs- und Spielersitzungsobj ekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden. Typ: `Integer` Required: Yes

UpdateGameSession

Aktualisierte Informationen für ein Spielsitzungsobjekt, einschließlich des Grundes, warum die Spielsitzung aktualisiert wurde. Wenn sich das Update auf eine Backfill-Aktion bezieht, enthält dieser Datentyp die Backfill-Ticket-ID.

Eigenschaften	Beschreibung
GameSession	Ein von der GameLift Amazon-API definiert es GameSessionObjekt. Das `GameSession` Objekt enthält Eigenschaften, die eine Spielsitz ung beschreiben. Typ: `GameSession GameSession()` Required: Yes
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: `UpdateReason UpdateReason()` Required: Yes

Eigenschaften	Beschreibung
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: `String` Required: Yes

GameSession

Einzelheiten einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Eine Spielesitzung (ARN) hat das folgende Format:`arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` . Typ: `String` Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: `String` Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung läuft. Typ: `String` Required: No

Eigenschaften	Beschreibung
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: `Integer` Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem GameLift Amazon-Sp ieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnumme r. Typ: `Integer` Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem GameLift Amazon-Sp ieleserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnumme r. Typ: `String` Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitz ungseigenschaften, die als einzelner Zeichenfo lgenwert formatiert sind. Typ: `String` Required: No

Eigenschaften	Beschreibung
MatchmakerData	Die Informationen über den Matchmaking-Prozess, mit dem die Spielsitzung erstellt wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält sie Daten über alle dem Spiel zugewiesenen Spieler, einschließlich Spielerattribute und Teamzuwei sungen. Typ: `String` Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenscha ften für eine Spielsitzung, formatiert als Schlüssel/Wert-Paare. Diese Eigenschaften werden mit der Anfrage übergeben, eine neue Spielsitzung zu starten. Typ: `Dictionary<string, string>` Required: No

Eigenschaften	Beschreibung
DnsName	Die DNS-ID, die der Instanz zugewiesen wurde, die die Spielsitzung ausführt. Werte haben das folgende Format: • TLS-fähige Flotten:. `<unique identifier>.<region identifier>.amazongamelift.com` • Nicht TLS-fähige Flotten:. `ec2-<unique identifier>.compute.amazonaws.com` Wenn du eine Verbindung zu einer Spielsitzung herstellst, die auf einer TLS-fähigen Flotte läuft, musst du den DNS-Namen verwenden, nicht die IP-Adresse. Typ: `String` Required: No

ServerParameters

Informationen, die zur Aufrechterhaltung der Verbindung zwischen einem GameLift Anywhere Amazon-Server und dem GameLift Amazon-Dienst verwendet werden. Diese Information wird verwendet, wenn neue Serverprozesse mit gestartet InitSDK() werden. Verwenden Sie für Server, die auf von Amazon GameLift verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
WebSocketUrl	Die `GameLiftServerSdkEndpoint` wurden zurückgegeben, `RegisterCompute` als Sie Teil von Amazon waren GameLiftAnywhere. Typ: `String`

Eigenschaften	Beschreibung
	Required: Yes
ProcessId	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der dein Spiel hostet. Typ: `String` Required: Yes
HostId	Eine eindeutige Kennung für den Host mit den Serverprozessen, die dein Spiel hosten. Die HostID wurde ComputeName verwendet, als Sie Ihren Computer registriert haben. Weitere Informationen finden Sie unter RegisterCompute Typ: `String` Required: Yes
FleetId	Die Flotten-ID der Flotte, für die der Computer registriert ist. Weitere Informationen finden Sie unter RegisterCompute. Typ: `String` Required: Yes
AuthToken	Das von Amazon generierte Authentifizierungstoken GameLift, das Ihren Server bei Amazon GameLift authentifiziert. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: `String` Required: Yes

StartMatchBackfillRequest

Informationen, die verwendet wurden, um eine Matchmaking-Backfill-Anfrage zu erstellen. Der Gameserver übermittelt diese Informationen GameLift in einem StartMatchBackfill() Anruf an Amazon.

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige ID der Spielsitzung. Die API-Operation `GetGameSessionId` gibt den Bezeichner im ARN-Format zurück. Typ: `String` Required: Yes
MatchmakingConfigurationArn	Die eindeutige Kennung in Form eines ARN, die der Matchmaker für diese Anfrage verwenden kann. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Mit Matchmaker-Daten arbeiten. Typ: `String` Required: Yes
Players	Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informati onen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: `List<Player>` Required: Yes

Eigenschaften	Beschreibung
TicketId	Die eindeutige Kennung für ein Matchmaki ng- oder Match-Backfill-Anfrageticket. Wenn Sie keinen Wert angeben, GameLift generiert Amazon einen. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen. Typ: `String` Required: No

Player

Repräsentiert einen Spieler im Matchmaking. Wenn eine Matchmaking-Anfrage gestartet wird, hat ein Spieler eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift fügt Teaminformationen hinzu, nachdem ein Spiel ausgetragen wurde.

Eigenschaften	Beschreibung
LatencyInFRAU	Eine Reihe von Werten, ausgedrückt in Millisekunden, die die Latenz angeben, die ein Spieler erlebt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Spieler nur den aufgelisteten Orten zugeordnet. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet , müssen die Spieler die zu vergleichende Latenz melden. Typ: `Dictionary<string, int>` Required: No

Eigenschaften	Beschreibung
PlayerAttributes	Eine Sammlung von Schlüssel:Wert-Paaren, die Spielerinformationen zur Verwendung beim Matchmaking enthalten. Die Schlüssel für Spielerattribute müssen mit denen übereinst immen, die in einem Matchmaking-Regelsatz PlayerAttributes verwendet werden. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: `Dictionary<string, AttributeValue` Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: `String` Required: No
Team	Der Name der Mannschaft, der der Spieler in einem Spiel zugewiesen ist. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: `String` Required: No

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielersitzung(en) abgerufen werden sollen. Es kann auf verschiedene Arten verwendet werden: (1) gib a PlayerSessionId an, um eine bestimmte Spielersitzung anzufordern; (2) gib a GameSessionId an, um alle Spielersitzungen in der angegebenen Spielsitzung anzufordern; oder (3) gib a PlayerId an, um alle Spielersitzungen für den angegebenen Spieler anzufordern. Verwenden Sie für große Sammlungen von Spielersitzungen die Paginierungsparameter, um Ergebnisse als aufeinander folgende Seiten abzurufen.

Eigenschaften	Beschreibung
GameSessionId	Die eindeutige ID der Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitzung anzufrage n. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von <ID string > ist entweder eine benutzerdefinierte ID-Zeiche nfolge (sofern bei der Erstellung der Spielsitz ung angegeben) oder eine generierte Zeichenfo lge. Typ: `String` Required: No
PlayerSessionId	Die eindeutige Kennung für eine Spielersitzung. Typ: `String` Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Siehe Generieren Sie Spieler-IDs. Typ: `String` Required: No
PlayerSessionStatusFilter	Der Sitzungsstatus des Spielers, nach dem die Ergebnisse gefiltert werden sollen. Mögliche Spielersitzungsstatus sind u. a.: • RESERVED – Die Anfrage nach einer Spielersitzung ist eingegangen, der Spieler ist jedoch noch nicht mit dem Serverprozess verbunden und/oder noch nicht validiert.

Eigenschaften	Beschreibung
	• ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden. • COMPLETED – Die Spielerverbindung wurde beendet. • TIMEDOUT – Eine Anfrage nach einer Spielersitzung ist eingegangen, der Spieler hat jedoch keine Verbindung hergestellt und/oder wurde nicht innerhalb des Timeout-Zeitraums (60 Sekunden) validiert. Typ: `String` Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `String` Required: No
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Spielersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `int` Required: No

StopMatchBackfillRequest

Informationen, die verwendet wurden, um eine Matchmaking-Backfill-Anfrage zu stornieren. Der Spieleserver übermittelt diese Informationen in einem StopMatchBackfill() Anruf an den GameLift Amazon-Dienst.

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Spielsitzungs-ID der Anfrage, die storniert wird. Typ: `string` Required: Yes
MatchmakingConfigurationArn	Die eindeutige Kennung des Matchmakers, an den diese Anfrage gesendet wurde. Typ: `string` Required: Yes
TicketId	Die eindeutige Kennung des Tickets für die Backfill-Anfrage, das storniert werden soll. Typ: `string` Required: Yes

GetFleetRoleCredentialsRequest

Dieser Datentyp gibt dem Spielserver eingeschränkten Zugriff auf deine anderen AWS Ressourcen. Weitere Informationen finden Sie unter Einrichten einer IAM-Servicerolle für Amazon GameLift.

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert.

Eigenschaften	Beschreibung
	Typ: `string` Required: Yes
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenanmeldeinformationen beschreibt. Typ: `string` Required: No

AttributeValue

Verwenden Sie diese Werte in Player Attributschlüssel-Wert-Paaren. Mit diesem Objekt können Sie einen Attributwert mithilfe eines der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenfolgenarray oder Datenzuordnung. Jedes `AttributeValue` Objekt kann nur eine der verfügbaren Eigenschaften verwenden.

Eigenschaften	Beschreibung
ATTR-Typ	Gibt den Typ des Attributwerts an. Typ: Ein `AttrType` Aufzählungswert. Required: No
S	Stellt einen Zeichenketten-Attributwert dar. Typ: `string` Required: Yes
N	Stellt einen numerischen Attributwert dar. Typ: `double` Required: Yes

Eigenschaften	Beschreibung
SL	Stellt ein Array von Zeichenketten-Attributwerten dar. Typ: `string[]` Required: Yes
SDM	Stellt ein Wörterbuch mit Zeichenkettenschlüsseln und Doppelwerten dar. Typ: `Dictionary<string, double>` Required: Yes

AwsStringOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `string` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GenericOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

DescribePlayerSessionsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “DescribePlayerSessionsResult” Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war.

Eigenschaften	Beschreibung
	Typ: the section called “GameLiftError” Required: No

DescribePlayerSessionsResult

Eigenschaften	Beschreibung
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Spielersitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `string` Required: Yes
PlayerSessions	Eine Sammlung von Objekten, die Eigenschaften für jede Spielersitzung enthalten, die der Anfrage entspricht. Typ: `IList<`the section called “PlayerSession” `>` Erforderlich:
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError”

Eigenschaften	Beschreibung
	Required: No

PlayerSession

Eigenschaften	Beschreibung
CreationTime	Typ: `long` Required: Yes
FleetId	Typ: `string` Required: Yes
GameSessionId	Typ: `string` Required: Yes
IpAddress	Typ: `string` Required: Yes
PlayerData	Typ: `string` Required: Yes
PlayerId	Typ: `string` Required: Yes
PlayerSessionId	Typ: `string` Required: Yes
Port	Typ: `int` Required: Yes

Eigenschaften	Beschreibung
Status	Typ: Eine `PlayerSessionStatus` Aufzählung. Required: Yes
TerminationTime	Typ: `long` Required: Yes
DnsName	Typ: `string` Required: Yes

StartMatchBackfillOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “StartMatchBackfillResult” Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

StartMatchBackfillResult

Eigenschaften	Beschreibung
TicketId	Typ: `string` Required: Yes

GetComputeCertificateOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetComputeCertificateResult” Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrem Computer und der Hostname des Computers.

Eigenschaften	Beschreibung
CertificatePath	Typ: `string` Required: Yes
ComputeName	Typ: `string` Required: Yes

GetFleetRoleCredentialsOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “GetFleetRoleCredentialsResult” Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GetFleetRoleCredentialsResult

Eigenschaften	Beschreibung
AccessKeyId	Die Zugangsschlüssel-ID zur Authentifizierung und zur Bereitstellung des Zugriffs auf Ihre AWS Ressourcen. Typ: `string` Required: No
AssumedRoleId	Die ID des Benutzers, dem die Servicerolle gehört. Typ: `string` Required: No
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, dem die Servicerolle gehört. Typ: `string` Required: No
Ablauf	Die Zeit, bis Ihre Sitzungsdaten ablaufen. Typ: `DateTime` Required: No
SecretAccessKey	Die geheime Zugangsschlüssel-ID für die Authentifizierung. Typ: `string` Required: No

Eigenschaften	Beschreibung
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren AWS Ressourcen interagiert. Typ: `string` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

AwsDateTimeOutcome

Dieser Datentyp resultiert aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `DateTime` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Required: Yes

Eigenschaften	Beschreibung
Fehler	Der Fehler, der aufgetreten ist, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

GameLiftError

Eigenschaften	Beschreibung
ErrorType	Der Fehlertyp. Typ: Eine `GameLiftErrorType` Aufzählung. Required: No
ErrorName	Der Name des Fehlers. Typ: `string` Required: No
ErrorMessage	Die Fehlermeldung. Typ: `string` Required: No

Aufzählungen

Enums, die für das Amazon GameLift Server SDK (C#) definiert sind, sind wie folgt definiert:

AttrType

- NONE
- SCHNUR
- VERDOPPELN

- ZEICHENFOLGENLISTE
- STRING_DOUBLE_MAP

GameLiftErrorType

Zeichenfolgenwert, der den Fehlertyp angibt. Gültige Werte sind:

- SERVICE_CALL_FAILED — Ein Anruf an einen AWS Dienst ist fehlgeschlagen.
- LOCAL_CONNECTION_FAILED — Die lokale Verbindung zu Amazon ist fehlgeschlagen. GameLift
- NETWORK_NOT_INITIALIZED — Das Netzwerk wurde nicht initialisiert.
- GAMESESSION_ID_NOT_SET — Die Spielsitzungs-ID wurde nicht festgelegt.
- UNGÜLTIGE ANFRAGE_AUSNAHME
- INTERNES_SERVICE_AUSNAHME
- ALREADY_INITIALIZED — Der Amazon GameLift Server oder Client wurde bereits mit Initialize () initialisiert.
- FLEET_MISMATCH — Die Zielflotte entspricht nicht der Flotte einer GameSession oder PlayerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED — Der Amazon-Client wurde nicht initialisiert. GameLift
- GAMELIFT_SERVER_NOT_INITIALIZED — Der Amazon-Server wurde nicht initialisiert. GameLift
- GAME_SESSION_ENDED_FAILED — Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung beendet ist.
- GAME_SESSION_NOT_READY — Die Amazon GameLift Server-Spielesitzung wurde nicht aktiviert.
- GAME_SESSION_READY_FAILED — Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist.
- INITIALIZATION_MISMATCH — Eine Client-Methode wurde nach Server: :Initialize () aufgerufen oder umgekehrt.
- NOT_INITIALIZED — Der Amazon GameLift Server oder Client wurde nicht mit Initialize () initialisiert.
- NO_TARGET_ALIASID_SET — Eine Ziel-AliasID wurde nicht gesetzt.
- NO_TARGET_FLEET_SET — Eine Zielflotte wurde nicht festgelegt.
- PROCESS_ENDING_FAILED — Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess beendet ist.

- PROCESS_NOT_ACTIVE — Der Serverprozess ist noch nicht aktiv, nicht an einen gebunden und kann ihn nicht GameSession annehmen oder verarbeiten. PlayerSessions
- PROCESS_NOT_READY — Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- PROCESS_READY_FAILED — Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess bereit ist.
- SDK_VERSION_DETECTION_FAILED — Die SDK-Versionserkennung ist fehlgeschlagen.
- STX_CALL_FAILED — Ein Aufruf der xSTX-Server-Backend-Komponente ist fehlgeschlagen.
- STX_INITIALIZATION_FAILED — Die xSTx-Server-Backend-Komponente konnte nicht initialisiert werden.
- UNEXPECTED_PLAYER_SESSION — Der Server hat eine unregistrierte Spielersitzung festgestellt.
- WEBSOCKET_CONNECT_FEHLER
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- UNGÜLTIGE URL FÜR WEBSOCKET_CONNECT_FEHLGESCHLAGEN
- TIMEOUT FÜR WEBSOCKET_CONNECT_FAILURE_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE — Wiederherstellbarer Fehler beim Senden einer Nachricht an den Dienst. GameLift WebSocket
- WEBSOCKET_SEND_MESSAGE_FAILURE — Fehler beim Senden einer Nachricht an den Dienst. GameLift WebSocket
- MATCH_BACKFILL_REQUEST_VALIDATION — Die Validierung der Anfrage ist fehlgeschlagen.
- PLAYER_SESSION_REQUEST_VALIDATION — Die Validierung der Anfrage ist fehlgeschlagen.

PlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.
- NOT_SET — Die Spielsitzung ist nicht so eingestellt, dass sie neue Spielersitzungen akzeptiert oder verweigert.

PlayerSessionStatus

- AKTIV

- COMPLETED
- NICHT_GESETZT
- RESERVIERT
- TIMEOUT

Amazon GameLift Server SDK 4.x-Referenz für C#

Diese Referenz zum Amazon GameLift C# Server SDK 4.x kann Ihnen helfen, Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten. GameLift Weitere Informationen zu dem Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Referenz zum GameLift Amazon-Server-SDK (C#): Aktionen
- Referenz zum Amazon GameLift Server SDK (C#): Datentypen

Referenz zum GameLift Amazon-Server-SDK (C#): Aktionen

Sie können diese Amazon GameLift C#-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

- Aktionen
- Datentypen

AcceptPlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spielersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Überprüfung benötigt. Amazon GameLift überprüft, ob die Spielersitzungs-ID gültig ist, d. h., ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung GameLift ändert Amazon den Status des Spielerslots von RESERVED auf ACTIVE.

Syntax

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird, GameLift wenn eine neue Spielersitzung erstellt wird. Eine Player-Sitzungs-ID ist in einem `PlayerSession` Objekt angegeben, das als Antwort auf einen Client-Aufruf der GameLiftAPI-Aktionen StartGameSessionPlacement, CreateGameSessionDescribeGameSessionPlacement, oder zurückgegeben wird DescribePlayerSessions.

Typ: Zeichenfolge

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht eine Funktion für die Verarbeitung einer Verbindungsanforderung, einschließlich dem Validieren und Abweisen ungültiger Spielersitzungs-IDs.

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerSessionId){
    var acceptPlayerSessionOutcome =
 GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
     if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
     else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```

ActivateGameSession()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion muss im Rahmen der

`onStartGameSession()`-Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
GenericOutcome ActivateGameSession()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird `ActivateGameSession()` als Teil der `onStartGameSession()`-Delegate-Funktion aufgerufen.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map

    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

Parameter

describePlayerSessionsAnfrage

Ein DescribePlayerSessionsRequest-Objekt, mit dem beschrieben wird, welche Spielersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein `DescribePlayerSessionsOutcome`-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem GameLift PlayerSessionAmazon-API-Datentyp des AWS SDK identisch ist.

Beispiel

Dieses Beispiel zeigt eine Anforderung für alle Spielersitzungen, die derzeit aktiv mit einer angegebenen Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextTokenund auf 10 setzen, gibt Amazon GameLift die Datensätze der ersten 10 Spielersitzungen zurück, die der Anfrage entsprechen.

```
// Set request parameters
var describePlayerSessionsRequest = new
 Aws.GameLift.Server.Model.DescribePlayerSessionsRequest()
{
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for
 the current game session
    Limit = 10,
    PlayerSessionStatusFilter =
 PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::Model::DescribePlayerSessions(describePlayerSessionRequest);
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist.

Bei inaktiven Prozessen, die noch nicht mit einer Spielsitzung aktiviert wurden, gibt der Aufruf `Success` = `True` und `GameSessionId` = `""` (eine leere Zeichenfolge) zurück.

Syntax

```
AwsStringOutcome GetGameSessionId()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `AwsStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

Ruft den Speicherort eines PEM-kodierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Managergeneriert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, wobei die Zertifikatskonfiguration auf GENERATED gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz

gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen über das Zertifikat und die Zertifikatskettendaten finden Sie unter GetCertificateAntwortelemente in der AWS Certificate Manager API-Referenz.

Beispiel

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
AwsStringOutcome GetSdkVersion()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `AwsStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5"). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf vom GameLift Amazon-Dienst erhalten hat. Amazon GameLift kann aus den folgenden `onProcessTerminate()` Gründen anrufen: (1) wegen eines

schlechten Zustands (der Serverprozess hat den Port-Zustand gemeldet oder hat Amazon nicht reagiert)GameLift, (2) wenn die Instance während eines Scale-Down-Ereignisses beendet wird oder (3) wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Wenn der Prozess einen onProcessTerminate() Rückruf erhalten hat, ist der zurückgegebene Wert die geschätzte Abbruchzeit. Wenn der Prozess keinen onProcessTerminate() Rückruf erhalten hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen über das Herunterfahren eines Serverprozesses.

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird die Abbruchzeit als AwsDateTimeOutcome Objekt zurückgegeben. Der Wert ist die Abbruchzeit, ausgedrückt in verstrichenen Ticks seit 0001 00:00:00. Beispielsweise entspricht der Datums-/Uhrzeitwert 2020-09-13 12:26:40 -000Z 637355968000000000 Ticks. Wenn keine Kündigungszeit verfügbar ist, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

Initialisiert das Amazon GameLift SDK. Diese Methode sollte beim Start aufgerufen werden, bevor eine andere Initialisierung GameLift im Zusammenhang mit Amazon erfolgt.

Syntax

```
InitSDKOutcome InitSDK()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein InitSdkOutcome Objekt zurückgegeben, das angibt, dass der Serverprozess zum Aufrufen bereit istProcessReady().

Beispiel

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

ProcessEnding()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess heruntergefahren wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem Beendigungscode 0 beendet werden. Andere Beendigungscodes sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Sobald die Methode mit einem Code von 0 beendet wird, können Sie den Prozess mit einem erfolgreichen Exit-Code beenden. Sie können den Vorgang auch mit einem Fehlercode beenden. Wenn Sie den Vorgang mit einem Fehlercode beenden, zeigt das Flottenereignis an, dass der Prozess ungewöhnlich beendet wurde (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Syntax

```
GenericOutcome ProcessEnding()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
   Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess bereit ist, Spielesitzungen zu hosten. Rufen Sie diese Methode auf, nachdem Sie die Einrichtungsaufgaben erfolgreich aufgerufen InitSDK() und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameter

processParameters

Ein ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spieleservercode implementierten Callback-Methoden, die der GameLift Amazon-Dienst zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen spielsitzungsspezifischen Dateien, die Amazon erfassen und GameLift speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des ProcessReady()-Aufrufs und der Delegate-Funktion.

```
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
   this.OnGameSession,
   this.OnProcessTerminate,
   this.OnHealthCheck,
```

```
   this.OnGameSessionUpdate,
   port,
   new LogParameters(new List<string>()          // Examples of log and error files
 written by the game server
   {
      "C:\\game\\logs",
      "C:\\game\\error"
   })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

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

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

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

RemovePlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spieler-Sitzungs-ID die Verbindung zum Serverprozess getrennt hat. Als Reaktion darauf GameLift ändert Amazon den Spielerplatz auf verfügbar, sodass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird, GameLift wenn eine neue Spielersitzung erstellt wird. Eine Player-Sitzungs-ID ist in einem `PlayerSession` Objekt angegeben, das als Antwort auf einen Client-Aufruf der GameLiftAPI-Aktionen StartGameSessionPlacement, CreateGameSessionDescribeGameSessionPlacement, oder zurückgegeben wird DescribePlayerSessions.

Typ: Zeichenfolge

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
Aws::GameLift::GenericOutcome disconnectOutcome =
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion StartMatchBackfill(). Mit dieser Aktion können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der GameLift Amazon-Service mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

Parameter

StartMatchBackfillRequest

Ein StartMatchBackfillRequest-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben GameLift wird, generiert Amazon automatisch eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.
- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Gibt ein StartMatchBackfillOutcome Objekt mit der Match-Backfill-Ticket-ID oder einen Fehler mit einer Fehlermeldung zurück.

Beispiel

```
// Build a backfill request
var startBackfillRequest = new AWS.GameLift.Server.Model.StartMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional
    MatchmakingConfigurationArn = "the matchmaker configuration ARN",
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    // gets ID for
 current game session
        //get player data for all currently connected players
            MatchmakerData matchmakerData =
              MatchmakerData.FromJson(gameSession.MatchmakerData);  // gets matchmaker
 data for current players
            // get matchmakerData.Players
```

```
            // remove data for players who are no longer connected
    Players = ListOfPlayersRemainingInTheGame
};

// Send backfill request
var startBackfillOutcome = GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
   // game-specific tasks to prepare for the newly matched players and update
 matchmaker data as needed
}
```

StopMatchBackfill()

Bricht eine aktive mit dem Befehl StartMatchBackfill() erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion StopMatchmaking(). Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Syntax

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameter

StopMatchBackfillRequest

Ein StopMatchBackfillRequest-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde
- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

```
// Set backfill stop request parameters

var stopBackfillRequest = new AWS.GameLift.Server.Model.StopMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional, if not provided one is autogenerated
    MatchmakingConfigurationArn = "the matchmaker configuration ARN", //from the game
 session matchmaker data
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result    //gets the ID for
 the current game session
};

var stopBackfillOutcome =
 GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess ProcessEnding() nach dem Ende einer Spielsitzung aufgerufen werden.

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Es sollte erst aufgerufen werden, nachdem das Verfahren zur Beendigung der Spielsitzung abgeschlossen ist, da es Amazon signalisiertGameLift, dass der Serverprozess sofort für das Hosten einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach Beendigung der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen an, ProcessEnding() um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
GenericOutcome TerminateGameSession()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel zeigt einen Serverprozess am Ende einer Spielsitzung.

```
// game-specific tasks required to gracefully shut down a game session,
// such as notifying players, preserving game state data, and other cleanup

var terminateGameSessionOutcome = GameLiftServerAPI.TerminateGameSession();
var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. (Siehe auch die Aktion UpdateGameSession() in der Amazon GameLift Service API-Referenz).

Syntax

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
 playerSessionPolicy)
```

Parameter

newPlayerSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Eingabe: PlayerSessionCreationPolicy enum. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
var updatePlayerSessionCreationPolicyOutcomex =

 GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

Referenz zum Amazon GameLift Server SDK (C#): Datentypen

Sie können diese Amazon GameLift C#-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon GameLift vorzubereiten. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

- Aktionen
- Datentypen

LogParameters

Dieser Datentyp wird verwendet, um zu ermitteln, welche Dateien während einer Spielsitzung generiert wurden und die Amazon GameLift nach Ende der Spielsitzung hochladen und speichern soll. Diese Informationen werden dem GameLift Amazon-Dienst in einem ProcessReady() Anruf mitgeteilt.

Inhalt

logPaths

Liste der Verzeichnispfade zu den Logdateien des Spielservers, GameLift die Amazon für den zukünftigen Zugriff speichern soll. Diese Dateien werden von einem Serverprozess während jeder Spielsitzung generiert; Dateipfade und -namen werden in Ihrem Spiel-Server definiert und im Spiel-Bild-Stammverzeichnis gespeichert. Die Log-Pfade müssen absolut sein. Zum Beispiel: Wenn Ihr Spiel-Build Spielsitzungsprotokolle in einem Pfad wie `MyGame\sessionlogs`

\ speichert, ist der Protokollpfad `c:\game\MyGame\sessionLogs` (auf einer Windows-Instance) oder `/local/game/MyGame/sessionLogs` (auf einer Linux-Instance).

Typ: Liste<String>

Required: No

DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielersitzung(en) abgerufen werden sollen. Es kann auf verschiedene Arten verwendet werden: (1) gib a PlayerSessionId an, um eine bestimmte Spielersitzung anzufordern; (2) gib a GameSessionId an, um alle Spielersitzungen in der angegebenen Spielsitzung anzufordern; oder (3) gib a PlayerId an, um alle Spielersitzungen für den angegebenen Spieler anzufordern. Verwenden Sie für große Sammlungen von Spielersitzungen die Paginierungsparameter, um Ergebnisse als aufeinander folgende Seiten abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von <ID string > ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Required: No

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit NextToken, um Ergebnisse als Satz aufeinanderfolgender Seiten zu erhalten. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Required: No

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Required: No

PlayerId

Eindeutiger Bezeichner für den Spieler. Player-IDs sind vom Entwickler definiert. Siehe Generieren Sie Spieler-IDs.

Typ: Zeichenfolge

Required: No

PlayerSessionId

Eindeutiger Bezeichner für eine Spielersitzung.

Typ: Zeichenfolge

Required: No

PlayerSessionStatusFilter

Status der Spielersitzung für die Filterung der Ergebnisse. Mögliche Spielersitzungsstatus sind u. a.:

- RESERVED – Die Anfrage nach einer Spielersitzung ist eingegangen, der Spieler ist jedoch noch nicht mit dem Serverprozess verbunden und/oder noch nicht validiert.
- ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.
- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT – Eine Anfrage nach einer Spielersitzung ist eingegangen, der Spieler hat jedoch keine Verbindung hergestellt und/oder wurde nicht innerhalb des Timeout-Zeitraums (60 Sekunden) validiert.

Typ: Zeichenfolge

Required: No

ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die bei einem ProcessReady() Anruf an den GameLift Amazon-Dienst gesendet werden.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: `Aws::GameLift::Server::LogParameters`

Erforderlich: Ja

onStartGameSitzung

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift ruft diese Funktion als Antwort auf die Kundenanfrage auf CreateGameSession. Die Callback-Funktion verwendet ein GameSessionObjekt (definiert in der Amazon GameLift Service API-Referenz).

Typ: `void OnStartGameSessionDelegate(GameSession gameSession)`

Erforderlich: Ja

onProcessTerminate

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon fünf Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem ProcessEnding() Anruf, bevor der Serverprozess heruntergefahren wird.

Typ: `void OnProcessTerminateDelegate()`

Erforderlich: Ja

onHealthCheck

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 60 Sekunden auf eine Antwort. Wenn keine Antwort empfangen wird, zeichnet Amazon den Serverprozess als fehlerhaft auf.

Typ: `bool OnHealthCheckDelegate()`

Erforderlich: Ja

onUpdateGameSitzung

Name der Callback-Funktion, die der GameLift Amazon-Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Es übergibt ein GameSessionObjekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID.

Typ: `void OnUpdateGameSessionDelegate ( UpdateGameSession updateGameSession )`

Required: No

StartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden in einem StartMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die SDK-Methode GetGameSessionId() gibt den Bezeichner im ARN-Format zurück.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker. Weitere Informationen zu Matchmaker-Daten finden Sie unter Mit Matchmaker-Daten arbeiten.

Typ: Zeichenfolge

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon GameLift API-Referenzhandbuch. Um Spielerattribute, IDs und Teamzuweisungen zu finden, suchen Sie im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Typ: Player[ ]

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben GameLift wird, generiert Amazon einen in Form einer UUID. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen.

Typ: Zeichenfolge

Required: No

StopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abzubrechen. Die Informationen werden in einem StopMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: Zeichenfolge

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: Zeichenfolge

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzubrechenden Tickets für die Backfill-Anforderung.

Typ: Zeichenfolge

Erforderlich: Ja

Amazon GameLift Server-SDK-Referenz für Go

Sie können diese Amazon GameLift Go-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereitenGameLift. Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Amazon GameLift Server SDK (Go)-Referenz: Aktionen
- Amazon GameLift Server SDK (Go)-Referenz: Datentypen

Amazon GameLift Server SDK (Go)-Referenz: Aktionen

Sie können diese Amazon- GameLift Go-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

`GameLiftServerAPI.go` definiert die Go-Server-SDK-Aktionen.

Aktionen

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
func GetSdkVersion() (string, error)
```

Rückgabewert

Bei Erfolg gibt die aktuelle SDK-Version als Zeichenfolge zurück. Die zurückgegebene Zeichenfolge enthält die Versionsnummer (Beispiel `5.0.0`). Wenn nicht erfolgreich, gibt eine Fehlermeldung wie zurück`common.SdkVersionDetectionFailed`.

Beispiel

```
version, err := server.GetSdkVersion()
```

InitSDK()

Initialisiert das Amazon GameLift SDK. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode richtet die Kommunikation zwischen dem Server und dem Amazon- GameLift Service ein.

Syntax

```
func InitSDK(params ServerParameters) error
```

Parameter

ServerParameters

Um einen Spieleserver auf einer Amazon GameLift Anywhere-Flotte zu initialisieren, erstellen Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL des , der für die Verbindung mit Ihrem Spieleserver WebSocket verwendet wird.
- Die ID des Prozesses, der zum Hosten Ihres Spieleservers verwendet wird.
- Die ID der Datenverarbeitung, die Ihre Spielserverprozesse hostet.
- Die ID der Amazon- GameLift Flotte, die Ihre Amazon GameLift Anywhere-Datenverarbeitung enthält.
- Das von der Amazon- GameLift Operation generierte Autorisierungstoken.

Um einen Spieleserver auf einer von Amazon GameLift verwalteten EC2-Flotte zu initialisieren, erstellen Sie ein `ServerParameters` Objekt ohne Parameter. Bei diesem Aufruf richtet der Amazon- GameLift Agent die Datenverarbeitungsumgebung ein und stellt automatisch eine Verbindung zum Amazon- GameLift Service für Sie her.

Rückgabewert

Bei Erfolg gibt den `nil` Fehler zurück, um anzuzeigen, dass der Serverprozess bereit ist, aufzurufenProcessReady().

> **Note**
>
> Wenn Aufrufe von für Spiele-Builds fehlschlagen, die in Anywhere-Flotten bereitgestellt `InitSDK()` werden, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wird. Sie müssen diesen Wert explizit auf die verwendete

Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn für eine neue Flotte bereit.

Beispiel

Amazon GameLift Anywhere-Beispiel

```
//Define the server parameters
serverParameters := ServerParameters {
  WebSocketURL: "wss://us-west-1.api.amazongamelift.com",
  ProcessID: "PID1234",
  HostID: "HardwareAnywhere",
  FleetID: "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa",
  AuthToken: "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
}

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
err := server.InitSDK(serverParameters)
```

Beispiel für Amazon GameLift Managed EC2

```
//Define the server parameters
serverParameters := ServerParameters {}

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

Benachrichtigt Amazon GameLift , dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufruf von aufInitSDK(). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
func ProcessReady(param ProcessParameters) error
```

Parameter

ProcessParameters

Ein ProcessParameters Objekt teilt die folgenden Informationen über den Serverprozess mit:

- Die Namen der Callback-Methoden, die im Spielserver-Code implementiert sind, die der Amazon- GameLift Service aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, die der Serverprozess überwacht.
- Der LogParameters Datentyp, der den Pfad zu allen für Spielsitzungen spezifischen Dateien enthält, die Amazon GameLift erfassen und speichern soll.

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt. Gibt zurück`nil`, wenn die Methode erfolgreich ist.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des ProcessReady()-Aufrufs und der Delegate-Funktion.

```
// Define the process parameters
processParams := ProcessParameters {
  OnStartGameSession: gameProcess.OnStartGameSession,
  OnUpdateGameSession: gameProcess.OnGameSessionUpdate,
  OnProcessTerminate: gameProcess.OnProcessTerminate,
  OnHealthCheck: gameProcess.OnHealthCheck,
  Port: port,
  LogParameters: LogParameters {    // logging and error example
    []string {"C:\\game\\logs", "C:\\game\\error"}
  }
}

err := server.ProcessReady(processParams)
```

ProcessEnding()

Benachrichtigt Amazon GameLift , dass der Serverprozess beendet wird. Rufen Sie diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich Herunterfahren der aktiven

Spielsitzung) und vor dem Beenden des Prozesses auf. Abhängig vom Ergebnis von wird `ProcessEnding()`der Prozess erfolgreich (0) oder mit einem Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntax

```
func ProcessEnding() error
```

Rückgabewert

Gibt einen 0-Fehlercode oder einen definierten Fehlercode zurück.

Beispiel

```
// operations to end game sessions and the server process
defer func() {
  err := server.ProcessEnding()
  server.Destroy()
  if err != nil {
    fmt.Println("ProcessEnding() failed. Error: ", err)
    os.Exit(-1)
  } else {
    os.Exit(0)
  }
}
```

ActivateGameSession()

Benachrichtigt Amazon GameLift , dass der Serverprozess eine Spielsitzung aktiviert hat und jetzt bereit ist, Spielerverbindungen zu empfangen. Diese Aktion wird nach der Initialisierung der gesamten Spielsitzung als Teil der `onStartGameSession()` Callback-Funktion aufgerufen.

Syntax

```
func ActivateGameSession() error
```

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt.

Beispiel

Dieses Beispiel zeigt, dass als Teil der `onStartGameSession()` Delegierungsfunktion `ActivateGameSession()` aufgerufen wird.

```
func OnStartGameSession(GameSession gameSession) {
  // game-specific tasks when starting a new game session, such as loading map
  // Activate when ready to receive players
  err := server.ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

Parameter

playerSessionCreationRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- **`model.AcceptAll`** – Akzeptieren Sie alle neuen Player-Sitzungen.
- **`model.DenyAll`** – Verweigern Sie alle neuen Player-Sitzungen.

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn ein Fehler auftritt.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Syntax

```
func GetGameSessionID() (string, error)
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg gibt die Spielsitzungs-ID und den Nil-Fehler zurück. Bei inaktiven Prozessen, die noch nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf eine leere Zeichenfolge und einen leeren `nil` Fehler zurück.

Beispiel

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

Gibt die Zeit zurück, zu der ein Serverprozess heruntergefahren werden soll, wenn eine Beendigungszeit verfügbar ist. Ein Serverprozess führt diese Aktion aus, nachdem er einen `onProcessTerminate()` Rückruf von Amazon erhalten hat GameLift. Amazon GameLift ruft `onProcessTerminate()` aus folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet oder nicht auf Amazon geantwortet hat GameLift.
- Beim Beenden der Instance während eines Herunterskalierungsereignisses.
- Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Syntax

```
func GetTerminationTime() (int64, error)
```

Rückgabewert

Bei Erfolg gibt den Zeitstempel in Epochensekunden zurück, in dem der Serverprozess heruntergefahren werden soll, und eine `nil` Fehlerbeendigung. Der Wert ist die Beendigungszeit, ausgedrückt in verstrichenen Ticks von `0001 00:00:00`. Der Datum-/Uhrzeitwert `2020-09-13 12:26:40 -000Z` entspricht beispielsweise `637355968000000000` Ticks. Wenn keine Beendigungszeit verfügbar ist, gibt eine Fehlermeldung zurück.

Beispiel

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player mit der angegebenen Player-Sitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und validiert werden muss. Amazon GameLift überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Player-Sitzung validiert wurde, GameLift ändert Amazon den Status des Player-Slots von `RESERVED` in `ACTIVE`.

Syntax

```
func AcceptPlayerSession(playerSessionID string) error
```

Parameter

`playerSessionId`

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die die Validierung und Ablehnung ungültiger Player-Sitzungs-IDs beinhaltet.

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {
    err := server.AcceptPlayerSession(playerSessionID)
    if err != nil {
        connection.Accept()
    } else {
        connection.Reject(err.Error())
    }
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player die Verbindung zum Serverprozess getrennt hat. Als Reaktion GameLift ändert Amazon den Player-Slot in „Verfügbar“.

Syntax

```
func RemovePlayerSession(playerSessionID string) error
```

Parameter

`playerSessionId`

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, die Einstellungen, Sitzungsmetadaten und Spielerdaten enthalten. Verwenden Sie diese Methode, um Informationen über Folgendes zu erhalten:

- Eine Einzelplayer-Sitzung

- Alle Spielersitzungen in einer Spielsitzung
- Alle Player-Sitzungen, die einer einzelnen Player-ID zugeordnet sind

Syntax

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)
 (result.DescribePlayerSessionsResult, error) {
 return srv.describePlayerSessions(&req)
}
```

Parameter

DescribePlayerSessionsRequest

Ein -`DescribePlayerSessionsRequest`Objekt beschreibt, welche Player-Sitzungen abgerufen werden sollen.

Rückgabewert

Bei Erfolg gibt ein `DescribePlayerSessionsResult` Objekt zurück, das eine Reihe von Spielersitzungsobjekten enthält, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spielersitzungen angefordert, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und auf 10 setzen, GameLift gibt Amazon die ersten 10 Spielersitzungsdatensätze zurück, die der Anforderung entsprechen.

```
// create request
describePlayerSessionsRequest := request.NewDescribePlayerSessions()
describePlayerSessionsRequest.GameSessionID, _ = server.GetGameSessionID() // get ID
 for the current game session
describePlayerSessionsRequest.Limit = 10                                  // return the
 first 10 player sessions
describePlayerSessionsRequest.PlayerSessionStatusFilter = "ACTIVE"         // Get all
 player sessions actively connected to the game session

describePlayerSessionsResult, err :=
 server.DescribePlayerSessions(describePlayerSessionsRequest)
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Diese Aktion ist asynchron. Wenn neue Spieler abgeglichen werden, GameLift liefert Amazon aktualisierte Matchmaker-Daten mithilfe der Callback-Funktion `OnUpdateGameSession()`.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
 (result.StartMatchBackfillResult, error)
```

Parameter

StartMatchBackfillRequest

Ein StartMatchBackfillRequest Objekt teilt die folgenden Informationen mit:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Wenn keine ID angegeben wird, GameLift generiert Amazon eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert befindet sich in den Matchmaker-Daten der Spielsitzung.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein `StartMatchBackfillResult` Objekt mit der Match-Backfill-Ticket-ID oder einen Fehler mit einer Fehlermeldung zurück.

Beispiel

```
// form the request
startBackfillRequest := request.NewStartMatchBackfill()
```

```
startBackfillRequest.RequestID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff"          //
 optional
startBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
var matchMaker model.MatchmakerData
if err := matchMaker.UnmarshalJSON([]byte(gameSession.MatchmakerData)); err != nil {

    return
}
startBackfillRequest.Players = matchMaker.Players
res, err := server.StartMatchBackfill(startBackfillRequest)

// Implement callback function for backfill
func OnUpdateGameSession(myGameSession model.GameSession) {
    // game-specific tasks to prepare for the newly matched players and update
 matchmaker data as needed
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anforderung ab. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Syntax

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

Parameter

StopMatchBackfillRequest

Ein StopMatchBackfillRequest Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die Ticket-ID, die der Backfill-Anforderung zugewiesen ist.
- Der Matchmaker, an den die Backfill-Anforderung gesendet wurde.
- Die Spielsitzung, die der Backfill-Anforderung zugeordnet ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
stopBackfillRequest := request.NewStopMatchBackfill()  // Use this function to create
 request
stopBackfillRequest.TicketID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
stopBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"

//error
err := server.StopMatchBackfill(stopBackfillRequest)
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zum Verschlüsseln der Netzwerkverbindung zwischen dem Spieleserver und Ihrem Spieleclient verwendet wird. Sie können den Zertifikatpfad verwenden, wenn Sie Ihr Rechengerät bei einer Amazon GameLift Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter RegisterCompute.

Syntax

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

Rückgabewert

Gibt ein `GetComputeCertificateResult` Objekt zurück, das Folgendes enthält:

- CertificatePath: Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource. Wenn Sie eine von Amazon GameLift verwaltete Flotte verwenden, enthält dieser Pfad:
  - `certificate.pem`: Das Endbenutzerzertifikat. Die vollständige Zertifikatkette ist die Kombination von , die an dieses Zertifikat `certificateChain.pem` angehängt ist.
  - `certificateChain.pem`: Die Zertifikatkette, die das Stammzertifikat und Zwischenzertifikate enthält.
  - `rootCertificate.pem`: Das Stammzertifikat.
  - `privateKey.pem`: Der private Schlüssel für das Endbenutzerzertifikat.
- ComputeName: Der Name Ihrer Rechenressource.

Beispiel

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

Ruft die Anmeldeinformationen der Servicerolle ab, die Sie erstellen, um die Berechtigungen auf Ihre anderen auf Amazon AWS-Services zu erweitern GameLift. Diese Anmeldeinformationen ermöglichen es Ihrem Spieleserver, Ihre -AWSRessourcen zu nutzen. Weitere Informationen finden Sie unter Einrichten einer IAM-Servicerolle für Amazon GameLift.

Syntax

```
func GetFleetRoleCredentials(
  req request.GetFleetRoleCredentialsRequest,
) (result.GetFleetRoleCredentialsResult, error) {
  return srv.getFleetRoleCredentials(&req)
}
```

Parameter

GetFleetRoleCredentialsRequest

Rollenanmeldeinformationen, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver erweitern.

Rückgabewert

Gibt ein `GetFleetRoleCredentialsResult` Objekt zurück, das Folgendes enthält:

- AssumedRoleUserArn – Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört.
- AssumedRoleId – Die ID des Benutzers, zu dem die Servicerolle gehört.
- AccessKeyId – Die Zugriffsschlüssel-ID zur Authentifizierung und Bereitstellung des Zugriffs auf Ihre -AWSRessourcen.
- SecretAccessKey – Die ID des geheimen Zugriffsschlüssels für die Authentifizierung.
- SessionToken – Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren -AWSRessourcen interagiert.
- Ablauf – Die Zeit bis zum Ablauf Ihrer Sitzungsanmeldeinformationen.

Beispiel

```
// form the customer credentials request
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction"

credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Zerstören ()

Entlastet das Amazon GameLift Game Server SDK aus dem Speicher. Als bewährte Methode rufen Sie diese Methode nach `ProcessEnding()` und auf, bevor Sie den Prozess beenden. Wenn Sie eine Anywhere-Flotte verwenden und Serverprozesse nicht nach jeder Spielsitzung beenden, rufen Sie `Destroy()` und auf, um sie neu `InitSDK()` zu initialisieren, bevor Sie Amazon benachrichtigen GameLift , dass der Prozess bereit ist, eine Spielsitzung mit zu hosten`ProcessReady()`.

Syntax

```
func Destroy() error {
 return srv.destroy()
}
```

Rückgabewert

Gibt einen Fehler mit einer Fehlermeldung zurück, wenn die Methode fehlschlägt.

Beispiel

```
// operations to end game sessions and the server process
defer func() {
  err := server.ProcessEnding()
  server.Destroy()
  if err != nil {
    fmt.Println("ProcessEnding() failed. Error: ", err)
    os.Exit(-1)
  } else {
    os.Exit(0)
  }
}
```

Amazon GameLift Server SDK (Go)-Referenz: Datentypen

Sie können diese Amazon- GameLift Go-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Datentypen

- LogParameters
- ProcessParameters
- UpdateGameSession
- GameSession
- ServerParameters
- StartMatchBackfillRequest
- Player
- DescribePlayerSessionsRequest
- StopMatchBackfillRequest
- GetFleetRoleCredentialsRequest

LogParameters

Ein Objekt, das Dateien identifiziert, die während einer Spielsitzung generiert wurden und die Amazon nach dem Ende der Spielsitzung GameLift hochladen und speichern soll. Der Spieleserver stellt Amazon GameLift als Teil eines `ProcessParameters` Objekts in einem ProcessReady() Aufruf `LogParameters` zur Verfügung.

Eigenschaften	Beschreibung
`LogPaths`	Die Liste der Verzeichnispfade zu Spielserver-Protokolldateien, die Amazon für GameLift den zukünftigen Zugriff speichern soll. Der Serverprozess generiert diese Dateien während jeder Spielsitzung. Sie definieren Dateipfade und -namen auf Ihrem Spieleserver und speichern sie im Stammspiel-Build-V erzeichnis. Die Protokollpfade müssen absolute Werte sein. Wenn Ihr Spiele-Build beispielsweise Spielsitzungsprotokolle in einem Pfad wie speichert`MyGame`

	`\sessionLogs\` , befindet sich der Pfad `c:\game\MyGame\sessionLogs` auf einer Windows-Instance. Typ: `[]string` Required: No

ProcessParameters

Ein Objekt, das die Kommunikation zwischen einem Serverprozess und Amazon beschreibt GameLift. Der Serverprozess stellt diese Informationen Amazon GameLift mit einem Aufruf von zur VerfügungProcessReady().

Eigenschaften	Beschreibung
`LogParameters`	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert werden. Amazon GameLift kopiert und speichert die Dateien für den zukünftigen Zugriff. Typ: `LogParameters` Required: No
`OnHealthCheck`	Die Rückruffunktion, die Amazon GameLift aufruft, um einen Zustandsstatusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess gibt zurückTRUE, wenn er fehlerfrei ist, FALSE wenn er nicht fehlerfrei ist. Wenn keine Antwort zurückgegeben wird, GameLift zeichnet Amazon den Serverprozess als nicht fehlerfrei auf. Typ: `OnHealthCheck func() bool` Required: No
`OnProcessTerminate`	Die Callback-Funktion, die Amazon GameLift aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 5 Minuten, bis der Serverprozess heruntergefahren und mit einem -ProcessEnding()Aufruf geantwortet hat, bevor der Serverprozess heruntergefahren wird.

	Typ: `OnProcessTerminate func()` Erforderlich: Ja
`OnStartGameSession`	Die Callback-Funktion, die Amazon GameLift aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anforderung verarbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Es übergibt ein GameSession Objekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID. Typ: `OnStartGameSession func (model.GameSession )` Erforderlich: Ja
`OnUpdateGameSession`	Die Callback-Funktion, die Amazon GameLift aufruft, um aktualisierte Spielsitzungsinformationen an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion nach der Verarbeitung einer Match-Backfill-Anforderung auf, um aktualisierte Matchmaker-Daten bereitzustellen. Typ: `OnUpdateGameSession func (model.UpdateGameSession)` Required: No
`Port`	Die Portnummer, die der Serverprozess auf neue Player-Verbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden. Typ: `int` Erforderlich: Ja

UpdateGameSession

Die aktualisiert ein Spielsitzungsobjekt, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, wenn Backfill verwendet wird, um Spielersitzungen in der Spielsitzung zu füllen.

Eigenschaften	Beschreibung
GameSession	Ein von der Amazon GameLift API definiertes GameSession Objekt. Das -`GameSession` Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: `GameSession GameSession()` Erforderlich: Ja
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: `UpdateReason UpdateReason()` Erforderlich: Ja
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: `String` Required: No

GameSession

Die Details einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Ein Amazon-Ressourcenname (ARN) für Spielsitzungen hat das folgende Format: `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` . Typ: `String` Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: `String`

Eigenschaften	Beschreibung
	Required: No
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung ausgeführt wird. Typ: `String` Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: `Integer` Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon-GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: `Integer` Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon-GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: `String` Required: No
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: `String` Required: No

Eigenschaften	Beschreibung
MatchmakerData	Die Informationen über den Matchmaking-Prozess, der zum Erstellen der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält sie Daten zu allen Spielern, die dem Match zugewiesen sind, einschließlich Spielerat tributen und Teamzuweisungen. Typ: `String` Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel-Wert-Paare. Diese Eigenschaften werden mit einer Anfrage zum Starten einer neuen Spielsitzung übergeben. Typ: `map[string] string` Required: No
DnsName	Die DNS-ID, die der Instance zugewiesen ist, die die Spielsitzung ausführt. Werte haben das folgende Format: • TLS-fähige Flotten: `<unique identifier>.<region identifie r>.amazongamelift.com` . • Nicht TLS-fähige Flotten: `ec2-<unique identifier>.comput e.amazonaws.com` . Wenn Sie eine Verbindung zu einer Spielsitzung herstellen, die auf einer TLS-fähigen Flotte ausgeführt wird, müssen Sie den DNS-Namen verwenden, nicht die IP-Adresse. Typ: `String` Required: No

ServerParameters

Informationen, die verwendet werden, um die Verbindung zwischen einem Amazon GameLift Anywhere-Server und dem Amazon- GameLift Service aufrechtzuerhalten. Diese Informationen werden verwendet, wenn neue Serverprozesse mit gestartet werdenInitSDK(). Verwenden Sie für Server, die auf von Amazon GameLift verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
`WebSocket URL`	Amazon `GameLiftServerSdkEndpoint` GameLift gibt zurück, wenn Sie `RegisterCompute` für eine Amazon GameLift Anywhere-Rechenressource verwenden. Typ: `string` Erforderlich: Ja
`ProcessID`	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der Ihr Spiel hostet. Typ: `string` Erforderlich: Ja
`HostID`	Die eindeutige Kennung der Rechenressource, die den neuen Serverprozess hostet. Die `HostID` wird `ComputeName` verwendet, als Sie Ihre Datenverarbeitung registriert haben. Weitere Informationen finden Sie unter RegisterCompute. Typ: `string` Erforderlich: Ja
`FleetID`	Die eindeutige Kennung der Flotte, bei der die Datenverarbeitung registriert ist. Weitere Informationen finden Sie unter RegisterCompute. Typ: `string` Erforderlich: Ja

Eigenschaften	Beschreibung
`AuthToken`	Das von Amazon generierte Authentifizierungstoken GameLift , das Ihren Server bei Amazon authentifiziert GameLift. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: `string` Erforderlich: Ja

StartMatchBackfillRequest

Informationen, die zum Erstellen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt Amazon diese Informationen GameLift in einem StartMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Kennung der Spielsitzung. Die API-Operation gibt die Kennung im ARN-Format `GetGameSessionId` zurück. Typ: `String` Erforderlich: Ja
MatchmakingConfigurationArn	Die eindeutige Kennung (in Form eines ARN), die der Matchmaker für diese Anforderung verwenden soll. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: `String` Erforderlich: Ja
Players	Ein Datensatz, der alle Spieler darstellt, die sich derzeit in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: `[]model.Player`

Eigenschaften	Beschreibung
	Erforderlich: Ja
TicketId	Die eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anf orderungsticket. Wenn Sie keinen Wert angeben, GameLift generiert Amazon einen. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen. Typ: `String` Required: No

Player

Das Objekt, das einen Spieler im Matchmaking darstellt. Wenn eine Matchmaking-Anfrage beginnt, verfügt ein Spieler über eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift fügt Teaminformationen hinzu, nachdem ein Match durchgeführt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, die in Millisekunden ausgedrückt werden und die Latenz angeben, die ein Spieler erfährt, wenn er eine Verbindung zu einem Standort herstellt. Wenn diese Eigenschaft verwendet wird, wird der Spieler nur mit den aufgelist eten Standorten abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet, müssen die Spieler die zu vergleichende Latenz melden. Typ: `map[string] int` Required: No
PlayerAttributes	Eine Sammlung von Schlüssel-Wert-Paaren, die Spielerinformationen für die Verwendung im Matchmaking enthalten. Spielerattributschlüssel müssen mit dem in einem Matchmaking-Regelsatz PlayerAttributes verwendeten übereinstimmen.

Eigenschaften	Beschreibung
	Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: `map[string] AttributeValue` Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: `String` Required: No
Team	Der Name des Teams, dem der Spieler in einem Spiel zugewiesen ist. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: `String` Required: No

DescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Player-Sitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem DescribePlayerSessions() Aufruf von Amazon bereit GameLift.

Eigenschaften	Beschreibung
`GameSessionID`	Eine eindeutige Spielsitzungs-ID. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitzung anzufragen. Das ID-Format für Spielsitzungen ist `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>` . ist `GameSessionID` eine benutzerdefinierte ID-Zeichenfolge oder eine generierte Zeichenfolge. Typ: `String` Required: No

Eigenschaften	Beschreibung
`PlayerSessionID`	Die eindeutige Kennung für eine Player-Sitzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Player-Sitzung anzufordern. Typ: `String` Required: No
`PlayerID`	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter , um alle Player-Sitzungen für einen bestimmten Player anzufordern. Siehe Generieren Sie Spieler-IDs. Typ: `String` Required: No
`PlayerSessionStatusFilter`	Der Status der Player-Sitzung, nach dem die Ergebnisse gefiltert werden sollen. Zu den möglichen Player-Sitzungsstatus gehören: • RESERVED – Die Player-Sitzungsanforderung wurde empfangen, aber der Player hat keine Verbindung zum Serverprozess hergestellt oder validiert. • ACTIVE – Der Player wurde vom Serverprozess validiert und ist verbunden. • COMPLETED – Die Player-Verbindung wurde unterbrochen. • TIMEDOUT – Eine Player-Sitzungsanforderung wurde empfangen, aber der Player hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeout-Limits (60 Sekunden) validiert. Typ: `String` Required: No
`NextToken`	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `String` Required: No

Eigenschaften	Beschreibung
`Limit`	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `int` Required: No

StopMatchBackfillRequest

Informationen, die zum Abbrechen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt diese Informationen dem Amazon- GameLift Service in einem StopMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Die eindeutige Spielsitzungs-ID der abgebrochenen Anfrage. Typ: `string` Required: No
MatchmakingConfigurationArn	Die eindeutige Kennung des Matchmakers, an den diese Anforderung gesendet wurde. Typ: `string` Required: No
TicketId	Die eindeutige Kennung des zu stornierenden Backfill-Anforderungstickets. Typ: `string` Required: No

GetFleetRoleCredentialsRequest

Die Rollenanmeldeinformationen, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver erweitern. Weitere Informationen finden Sie unter Einrichten einer IAM-Servicerolle für Amazon GameLift.

Eigenschaften	Beschreibung
RoleArn	Der ARN der Servicerolle, die den eingeschränkten Zugriff auf Ihre -AWSRessourcen erweitert. Typ: `string` Erforderlich: Ja
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenanmeldeinformationen beschreibt. Typ: `string` Erforderlich: Ja

Amazon GameLift Server-SDK-Referenz für Unreal Engine

Diese Amazon GameLift Server-SDK-Referenz kann Ihnen helfen, Ihre Unreal Engine-Spieleprojekte für die Verwendung mit Amazon GameLift vorzubereiten. Weitere Informationen zu dem Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Diese API ist in `GameLiftServerSDK.h` und `GameLiftServerSDKModels.h` definiert.

So richten Sie das Unreal Engine-Plug-in ein und zeigen Codebeispiele an Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt:

Themen

- Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x
- Referenz zum Amazon GameLift Unreal Engine-Server-SDK 3.x

Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x

Sie können diese Referenz zum Amazon GameLift Unreal Engine Server-SDK 5.x verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten. GameLift Einzelheiten zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur Verwendung des Unreal SDK-Server-Plug-ins finden Sie unterIntegrieren Sie Amazon GameLift in ein Unreal Engine-Projekt.

Themen

- Amazon GameLift Server SDK (Unreal) 5.x-Referenz: Aktionen
- Amazon GameLift Server SDK (Unreal)-Referenz: Datentypen

Amazon GameLift Server SDK (Unreal) 5.x-Referenz: Aktionen

Sie können diese Amazon GameLift Unreal Server SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur Verwendung des Server-Plugins Unreal SDK finden Sie unter Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt.

Aktionen

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()

- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()

> **Note**
>
> In diesem Thema wird die Amazon GameLift -C++-API beschrieben, die Sie beim Erstellen für die Unreal Engine verwenden können. Insbesondere gilt diese Dokumentation für Code, den Sie mit der `-DBUILD_FOR_UNREAL=1` Option kompilieren.

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
FGameLiftStringOutcome GetSdkVersion();
```

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als the section called "FGameLiftStringOutcome"-Objekt zurück. Das zurückgegebene Objekt enthält die Versionsnummer (Beispiel `5.0.0`). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
 Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine verwaltete EC2-Flotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode liest Serverparameter aus der Host-Umgebung, um die Kommunikation zwischen dem Server und dem Amazon GameLift-Service einzurichten.

Syntax

```
FGameLiftGenericOutcome InitSDK()
```

Rückgabewert

Bei Erfolg gibt ein `InitSdkOutcome` Objekt zurück, das angibt, dass der Serverprozess bereit ist, aufzurufenProcessReady().

Beispiel

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
FGameLiftGenericOutcome initSdkOutcome = gameLiftSdkModule->InitSDK();
```

InitSDK()

Initialisiert das Amazon GameLift SDK für eine -AnywhereFlotte. Rufen Sie diese Methode beim Start auf, bevor eine andere Initialisierung im Zusammenhang mit Amazon GameLift erfolgt. Diese Methode erfordert explizite Serverparameter, um die Kommunikation zwischen dem Server und dem Amazon- GameLift Service einzurichten.

Syntax

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

Parameter

FServerParameters

Um einen Spieleserver auf einer Amazon GameLift Anywhere-Flotte zu initialisieren, erstellen Sie ein `ServerParameters` Objekt mit den folgenden Informationen:

- Die URL des , der für die Verbindung mit Ihrem Spieleserver WebSocket verwendet wird.
- Die ID des Prozesses, der zum Hosten Ihres Spieleservers verwendet wird.
- Die ID der Datenverarbeitung, die Ihre Spielserverprozesse hostet.
- Die ID der Amazon- GameLift Flotte, die Ihre Amazon GameLift Anywhere-Datenverarbeitung enthält.
- Das von der Amazon- GameLift Operation generierte Autorisierungstoken.

Rückgabewert

Bei Erfolg gibt ein `InitSdkOutcome` Objekt zurück, das angibt, dass der Serverprozess bereit ist, aufzurufenProcessReady().

> **Note**
>
> Wenn Aufrufe von für Spiele-Builds fehlschlagen, die in Anywhere-Flotten bereitgestellt `InitSDK()` werden, überprüfen Sie den `ServerSdkVersion` Parameter, der beim Erstellen der Build-Ressource verwendet wird. Sie müssen diesen Wert explizit auf die verwendete Server-SDK-Version festlegen. Der Standardwert für diesen Parameter ist 4.x, was nicht kompatibel ist. Um dieses Problem zu beheben, erstellen Sie einen neuen Build und stellen Sie ihn für eine neue Flotte bereit.

Beispiel

```
//Define the server parameters
FServerParameters serverParameters;
parameters.m_authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
parameters.m_fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
parameters.m_hostId = "HardwareAnywhere";
parameters.m_processId = "PID1234";
parameters.m_webSocketUrl = "wss://us-west-1.api.amazongamelift.com";

//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
FGameLiftGenericOutcome initSdkOutcome = gameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

Benachrichtigt Amazon GameLift , dass der Serverprozess bereit ist, Spielsitzungen zu hosten. Rufen Sie diese Methode nach dem Aufruf von aufInitSDK(). Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Parameter

processParameters

Ein FProcessParameters Objekt, das die folgenden Informationen über den Serverprozess kommuniziert:

- Namen von Rückrufmethoden, die im Spielservercode implementiert sind, die der Amazon-GameLift Service aufruft, um mit dem Serverprozess zu kommunizieren.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen für Spielsitzungen spezifischen Dateien, die Amazon GameLift erfassen und speichern soll.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel veranschaulicht die Implementierung des ProcessReady()-Aufrufs und der Delegate-Funktion.

```
//Calling ProcessReady tells GameLift this game server is ready to receive incoming
 game sessions!
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = gameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

Benachrichtigt Amazon GameLift , dass der Serverprozess beendet wird. Rufen Sie diese Methode nach allen anderen Bereinigungsaufgaben (einschließlich Herunterfahren der aktiven Spielsitzung) und vor dem Beenden des Prozesses auf. Abhängig vom Ergebnis von wird `ProcessEnding( )`der Prozess erfolgreich (0) oder mit einem Fehler (-1) beendet und generiert ein Flottenereignis. Wenn der Prozess mit einem Fehler beendet wird, ist das generierte Flottenereignis `SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Syntax

```
FGameLiftGenericOutcome ProcessEnding()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
//OnProcessTerminate callback. GameLift will invoke this callback before shutting down
 an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
 etc., before being shut down.
//In this case, we simply tell GameLift we are indeed going to shutdown.
params->OnTerminate.BindLambda([=]() {
  UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
  gameLiftSdkModule->ProcessEnding();
});
```

ActivateGameSession()

Benachrichtigt Amazon GameLift , dass der Serverprozess eine Spielsitzung aktiviert hat und jetzt Spielerverbindungen empfangen kann. Diese Aktion sollte nach der Initialisierung der gesamten Spielsitzung als Teil der `onStartGameSession()` Rückruffunktion aufgerufen werden.

Syntax

```
FGameLiftGenericOutcome ActivateGameSession()
```

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

Dieses Beispiel zeigt, dass als Teil der `onStartGameSession()` Delegierungsfunktion `ActivateGameSession()` aufgerufen wird.

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

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt.

Syntax

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
 policy)
```

Parameter

playerCreationSessionRichtlinie

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel werden die Richtlinien für die aktuelle Spielsitzung für neue Spieler so festgelegt, dass alle Spieler akzeptiert werden.

```
FGameLiftGenericOutcome outcome = gameLiftSdkModule-
>UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die vom aktiven Serverprozess gehostet wird.

Bei inaktiven Prozessen, die nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf eine zurückthe section called “FGameLiftError”.

Syntax

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als the section called “FGameLiftStringOutcome”-Objekt zurück. Wenn nicht erfolgreich, wird eine Fehlermeldung zurückgegeben.“

Bei inaktiven Prozessen, die nicht mit einer Spielsitzung aktiviert sind, gibt der Aufruf `Success=True` und `GameSessionId=` zurück`""`.

Beispiel

```
//When a game session is created, GameLift sends an activation request to the game
 server and passes along the game session object containing game properties and other
 settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
 invoke GameLiftServerAPI.ActivateGameSession()
```

```
auto onGameSession = [=](Aws::GameLift::Server::Model::GameSession gameSession)
{
  FString gameSessionId = FString(gameSession.GetGameSessionId());
  UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"), *gameSessionId);
  gameLiftSdkModule->ActivateGameSession();
};
```

GetTerminationTime()

Gibt die Zeit zurück, für die das Herunterfahren eines Serverprozesses geplant ist (wenn eine Zeit zum Beenden verfügbar ist). Ein Serverprozess ergreift Maßnahmen, nachdem er einen `onProcessTerminate()` Rückruf von Amazon erhalten hat GameLift. Amazon GameLift ruft `onProcessTerminate()` aus folgenden Gründen auf:

- Wenn der Serverprozess einen schlechten Zustand gemeldet hat oder nicht auf Amazon geantwortet hat GameLift.
- Beim Beenden der Instance während eines Herunterskalierungsereignisses.
- Wenn eine Instance aufgrund einer Spot-Instance-Unterbrechung beendet wird.

Syntax

```
AwsDateTimeOutcome GetTerminationTime()
```

Rückgabewert

Bei Erfolg gibt die Beendigungszeit als `AwsDateTimeOutcome` Objekt zurück. Der Wert ist die Beendigungszeit, ausgedrückt in verstrichenen Ticks seit `0001 00:00:00`. Der Datum-/Uhrzeitwert `2020-09-13 12:26:40 -000Z` entspricht beispielsweise `637355968000000000` Ticks. Wenn keine Beendigungszeit verfügbar ist, gibt eine Fehlermeldung zurück.

Wenn der Prozess keinen Rückruf erhalten `ProcessParameters.OnProcessTerminate()` hat, wird eine Fehlermeldung zurückgegeben. Weitere Informationen zum Herunterfahren eines Serverprozesses finden Sie unter Reagieren Sie auf eine Benachrichtigung zum Herunterfahren des Serverprozesses.

Beispiel

```
AwsDateTimeOutcome TermTimeOutcome = gameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player mit der angegebenen Player-Sitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und validiert werden muss. Amazon GameLift überprüft, ob die Player-Sitzungs-ID gültig ist. Nachdem die Player-Sitzung validiert wurde, GameLift ändert Amazon den Status des Player-Slots von RESERVED in ACTIVE.

Syntax

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

In diesem Beispiel wird eine Verbindungsanforderung behandelt, die die Validierung und Ablehnung ungültiger Player-Sitzungs-IDs beinhaltet.

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerSessionId, const
 FString& playerId)
{
  #if WITH_GAMELIFT
  UE_LOG(GameServerLog, Log, TEXT("Accepting GameLift PlayerSession: %s . PlayerId:
 %s"), *playerSessionId, *playerId);
  FString gsId = GetCurrentGameSessionId();
  if (gsId.IsEmpty()) {
    UE_LOG(GameServerLog, Log, TEXT("No GameLift GameSessionId. Returning early!"));
    return false;
  }

  if (!gameLiftSdkModule->AcceptPlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession not Accepted."));
```

```
    return false;
  }

  // Add PlayerSession from internal data structures keeping track of connected players
  connectedPlayerSessionIds.Add(playerSessionId);
  idToPlayerSessionMap.Add(playerSessionId, PlayerSession{ playerId,
 playerSessionId });
  return true;
  #else
  return false;
  #endif
}
```

RemovePlayerSession()

Benachrichtigt Amazon GameLift , dass ein Player die Verbindung zum Serverprozess getrennt hat. Als Reaktion GameLift ändert Amazon den Player-Slot in „Verfügbar“.

Syntax

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

Parameter

`playerSessionId`

Eindeutige ID, die von Amazon ausgestellt wird GameLift , wenn eine neue Player-Sitzung erstellt wird.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
bool GameLiftManager::RemovePlayerSession(const FString& playerSessionId)
{
  #if WITH_GAMELIFT
  UE_LOG(GameServerLog, Log, TEXT("Removing GameLift PlayerSession: %s"),
 *playerSessionId);
```

```
  if (!gameLiftSdkModule->RemovePlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession Removal Failed"));
    return false;
  }

  // Remove PlayerSession from internal data structures that are keeping track of
 connected players
  connectedPlayerSessionIds.Remove(playerSessionId);
  idToPlayerSessionMap.Remove(playerSessionId);

  // end the session if there are no more players connected
  if (connectedPlayerSessionIds.Num() == 0) {
    EndSession();
  }

  return true;
  #else
  return false;
  #endif
}
```

DescribePlayerSessions()

Ruft Spielersitzungsdaten ab, die Einstellungen, Sitzungsmetadaten und Spielerdaten enthalten. Verwenden Sie diese Methode, um Informationen über Folgendes zu erhalten:

- Eine Einzelplayer-Sitzung
- Alle Spielersitzungen in einer Spielsitzung
- Alle Player-Sitzungen, die einer einzelnen Player-ID zugeordnet sind

Syntax

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
 FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameter

FGameLiftDescribePlayerSessionsRequest

Ein the section called “FGameLiftDescribePlayerSessionsRequest” Objekt, das beschreibt, welche Player-Sitzungen abgerufen werden sollen.

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein the section called "FGameLiftDescribePlayerSessionsOutcome"-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen.

Beispiel

In diesem Beispiel werden alle Spielersitzungen angefordert, die aktiv mit einer bestimmten Spielsitzung verbunden sind. Wenn Sie den Grenzwert weglassen NextToken und auf 10 setzen, GameLift gibt Amazon die ersten 10 Spielersitzungsdatensätze zurück, die der Anforderung entsprechen.

```
void GameLiftManager::DescribePlayerSessions()
{
  #if WITH_GAMELIFT
  FString localPlayerSessions;
  for (auto& psId : connectedPlayerSessionIds)
  {
    PlayerSession ps = idToPlayerSessionMap[psId];
    localPlayerSessions += FString::Printf(TEXT("%s : %s  ; "), *(ps.playerSessionId),
 *(ps.playerId));
  }
  UE_LOG(GameServerLog, Log, TEXT("LocalPlayerSessions: %s"), *localPlayerSessions);

  UE_LOG(GameServerLog, Log, TEXT("Describing PlayerSessions in this GameSession"));
  FGameLiftDescribePlayerSessionsRequest request;
  request.m_gameSessionId = GetCurrentGameSessionId();

  FGameLiftDescribePlayerSessionsOutcome outcome = gameLiftSdkModule-
>DescribePlayerSessions(request);
  LogDescribePlayerSessionsOutcome(outcome);
  #endif
}
```

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Diese Aktion ist asynchron. Wenn neue Spieler abgeglichen werden, GameLift liefert Amazon aktualisierte Matchmaker-Daten mithilfe der Callback-Funktion `OnUpdateGameSession()`.

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
 &startBackfillRequest);
```

Parameter

FStartMatchBackfillRequest

Ein StartMatchBackfillRequest Objekt, das die folgenden Informationen kommuniziert:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Informationen sind optional. Wenn keine ID angegeben wird, GameLift generiert Amazon eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert befindet sich in den Matchmaker-Daten der Spielsitzung.
- Die ID der Spielsitzung, die aufgefüllt werden soll.
- Die verfügbaren Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Rückgabewert

Gibt ein `StartMatchBackfillOutcome` Objekt mit der Match-Backfill-Ticket-ID oder einen Fehler mit einer Fehlermeldung zurück.

Beispiel

```
FGameLiftStringOutcome FGameLiftServerSDKModule::StartMatchBackfill(const
 FStartMatchBackfillRequest& request)
{
  #if WITH_GAMELIFT
  Aws::GameLift::Server::Model::StartMatchBackfillRequest sdkRequest;
  sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
  sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

 sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn
  for (auto player : request.m_players) {
    Aws::GameLift::Server::Model::Player sdkPlayer;
```

```
    sdkPlayer.SetPlayerId(TCHAR_TO_UTF8(*player.m_playerId));
    sdkPlayer.SetTeam(TCHAR_TO_UTF8(*player.m_team));
    for (auto entry : player.m_latencyInMs) {
      sdkPlayer.WithLatencyMs(TCHAR_TO_UTF8(*entry.Key), entry.Value);
    }

    std::map<std::string, Aws::GameLift::Server::Model::AttributeValue>
sdkAttributeMap;
    for (auto attributeEntry : player.m_playerAttributes) {
      FAttributeValue value = attributeEntry.Value;
      Aws::GameLift::Server::Model::AttributeValue attribute;
      switch (value.m_type) {
        case FAttributeType::STRING:
          attribute =
Aws::GameLift::Server::Model::AttributeValue(TCHAR_TO_UTF8(*value.m_S));
        break;
        case FAttributeType::DOUBLE:
          attribute = Aws::GameLift::Server::Model::AttributeValue(value.m_N);
        break;
        case FAttributeType::STRING_LIST:
          attribute =
Aws::GameLift::Server::Model::AttributeValue::ConstructStringList();
          for (auto sl : value.m_SL) {
            attribute.AddString(TCHAR_TO_UTF8(*sl));
          };
        break;
        case FAttributeType::STRING_DOUBLE_MAP:
          attribute =
Aws::GameLift::Server::Model::AttributeValue::ConstructStringDoubleMap();
          for (auto sdm : value.m_SDM) {
            attribute.AddStringAndDouble(TCHAR_TO_UTF8(*sdm.Key), sdm.Value);
          };
        break;
      }
      sdkPlayer.WithPlayerAttribute((TCHAR_TO_UTF8(*attributeEntry.Key)), attribute);
    }
    sdkRequest.AddPlayer(sdkPlayer);
  }
  auto outcome = Aws::GameLift::Server::StartMatchBackfill(sdkRequest);
  if (outcome.IsSuccess()) {
    return FGameLiftStringOutcome(outcome.GetResult().GetTicketId());
  }
  else {
    return FGameLiftStringOutcome(FGameLiftError(outcome.GetError()));
```

```
  }
  #else
  return FGameLiftStringOutcome("");
  #endif
}
```

StopMatchBackfill()

Bricht eine aktive Match-Backfill-Anforderung ab. Weitere Informationen finden Sie unter FlexMatch Backfill-Feature.

Syntax

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

Parameter

FStopMatchBackfillRequest

Ein StopMatchBackfillRequest Objekt, das das zu stornierende Matchmaking-Ticket identifiziert:

- Die Ticket-ID, die der Backfill-Anforderung zugewiesen ist.
- Der Matchmaker, an den die Backfill-Anforderung gesendet wurde.
- Die Spielsitzung, die der Backfill-Anforderung zugeordnet ist.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Fehlschlag mit einer Fehlermeldung besteht.

Beispiel

```
FGameLiftGenericOutcome FGameLiftServerSDKModule::StopMatchBackfill(const
 FStopMatchBackfillRequest& request)
{
  #if WITH_GAMELIFT
  Aws::GameLift::Server::Model::StopMatchBackfillRequest sdkRequest;
  sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
  sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

 sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn
```

```
  auto outcome = Aws::GameLift::Server::StopMatchBackfill(sdkRequest);
  if (outcome.IsSuccess()) {
    return FGameLiftGenericOutcome(nullptr);
  }
  else {
    return FGameLiftGenericOutcome(FGameLiftError(outcome.GetError()));
  }
  #else
  return FGameLiftGenericOutcome(nullptr);
  #endif
}
```

GetComputeCertificate()

Ruft den Pfad zum TLS-Zertifikat ab, das zum Verschlüsseln der Netzwerkverbindung zwischen Ihrer Amazon GameLift Anywhere-Rechenressource und Amazon verwendet wird GameLift. Sie können den Zertifikatpfad verwenden, wenn Sie Ihr Rechengerät bei einer Amazon GameLift Anywhere-Flotte registrieren. Weitere Informationen finden Sie unter RegisterCompute.

Syntax

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

Rückgabewert

Gibt ein `GetComputeCertificateResponse` Objekt zurück, das Folgendes enthält:

- CertificatePath: Der Pfad zum TLS-Zertifikat auf Ihrer Rechenressource.
- HostName: Der Hostname Ihrer Rechenressource.

Beispiel

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
{
  #if WITH_GAMELIFT
  auto outcome = Aws::GameLift::Server::GetComputeCertificate();
  if (outcome.IsSuccess()) {
    auto& outres = outcome.GetResult();
    FGameLiftGetComputeCertificateResult result;
    result.m_certificate_path = UTF8_TO_TCHAR(outres.GetCertificatePath());
    result.m_computeName = UTF8_TO_TCHAR(outres.GetComputeName());
```

```
    return FGameLiftGetComputeCertificateOutcome(result);
  }
  else {
    return FGameLiftGetComputeCertificateOutcome(FGameLiftError(outcome.GetError()));
  }
  #else
  return FGameLiftGetComputeCertificateOutcome(FGameLiftGetComputeCertificateResult());
  #endif
}
```

GetFleetRoleCredentials()

Ruft IAM-Rollen-Anmeldeinformationen ab, die Amazon GameLift zur Interaktion mit anderen autorisierenAWS-Services. Weitere Informationen finden Sie unter Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten.

Syntax

```
FGameLiftGetFleetRoleCredentialsOutcome
 FGameLiftServerSDKModule::GetFleetRoleCredentials(const
 FGameLiftGetFleetRoleCredentialsRequest &request)
```

Parameter

FGameLiftGetFleetRoleCredentialsRequest

Rückgabewert

Gibt ein the section called “FGameLiftGetFleetRoleCredentialsOutcome”-Objekt zurück.

Beispiel

```
FGameLiftGetFleetRoleCredentialsOutcome
 FGameLiftServerSDKModule::GetFleetRoleCredentials(const
 FGameLiftGetFleetRoleCredentialsRequest &request)
{
  #if WITH_GAMELIFT
  Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest sdkRequest;
  sdkRequest.SetRoleArn(TCHAR_TO_UTF8(*request.m_roleArn));
  sdkRequest.SetRoleSessionName(TCHAR_TO_UTF8(*request.m_roleSessionName));

  auto outcome = Aws::GameLift::Server::GetFleetRoleCredentials(sdkRequest);
```

```
  if (outcome.IsSuccess()) {
    auto& outres = outcome.GetResult();
    FGameLiftGetFleetRoleCredentialsResult result;
    result.m_assumedUserRoleArn = UTF8_TO_TCHAR(outres.GetAssumedUserRoleArn());
    result.m_assumedRoleId = UTF8_TO_TCHAR(outres.GetAssumedRoleId());
    result.m_accessKeyId = UTF8_TO_TCHAR(outres.GetAccessKeyId());
    result.m_secretAccessKey = UTF8_TO_TCHAR(outres.GetSecretAccessKey());
    result.m_sessionToken = UTF8_TO_TCHAR(outres.GetSessionToken());
    result.m_expiration = FDateTime::FromUnixTimestamp(outres.GetExpiration());
    return FGameLiftGetFleetRoleCredentialsOutcome(result);
  }
  else {
    return FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftError(outcome.GetError()));
  }
  #else
  return
 FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftGetFleetRoleCredentialsResult());
  #endif
}
```

Amazon GameLift Server SDK (Unreal)-Referenz: Datentypen

Sie können diese Amazon GameLift -Unreal-Server-SDK-Referenz verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten GameLift. Weitere Informationen zum Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu und Informationen zur Verwendung des Server-Plugins Unreal SDK finden Sie unter Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt.

Datentypen

- FProcessParameters
- UpdateGameSession
- GameSession
- FServerParameters
- FStartMatchBackfillRequest
- FPlayer
- FGameLiftDescribePlayerSessionsRequest
- FStopMatchBackfillRequest
- FAttributeValue

- FGameLiftGetFleetRoleCredentialsRequest
- FGameLiftLongOutcome
- FGameLiftStringOutcome
- FGameLiftDescribePlayerSessionsOutcome
- FGameLiftDescribePlayerSessionsResult
- FGenericOutcome
- FGameLiftPlayerSession
- FGameLiftGetComputeCertificateOutcome
- FGameLiftGetComputeCertificateResult
- FGameLiftGetFleetRoleCredentialsOutcome
- FGetFleetRoleCredentialsResult
- FGameLiftError
- Aufzählungen

> **Note**
>
> In diesem Thema wird die Amazon GameLift -C++-API beschrieben, die Sie beim Erstellen für die Unreal Engine verwenden können. Insbesondere gilt diese Dokumentation für Code, den Sie mit der `-DBUILD_FOR_UNREAL=1` Option kompilieren.

FProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die an Amazon GameLift in einem gesendet werdenProcessReady().

Eigenschaften	Beschreibung
LogParameters	Ein Objekt mit Verzeichnispfaden zu Dateien, die während einer Spielsitzung generiert werden. Amazon GameLift kopiert und speichert die Dateien für den zukünftigen Zugriff. Typ: `TArray<FString>`

	Required: No
OnHealthCheck	Die Rückruffunktion, die Amazon GameLift aufruft, um einen Zustandsstatusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf und wartet 60 Sekunden auf eine Antwort. Der Serverprozess gibt zurück`TRUE`, wenn er fehlerfrei ist, `FALSE` wenn er nicht fehlerfrei ist. Wenn keine Antwort zurückgegeben wird, GameLift zeichnet Amazon den Serverprozess als nicht fehlerfrei auf. Diese Eigenschaft ist eine Delegierungsfunktion, die als definiert ist`DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)` ; Typ: `FOnHealthCheck` Required: No
OnProcessTerminate	Die Callback-Funktion, die Amazon GameLift aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 5 Minuten, bis der Serverprozess heruntergefahren wird und mit einem -ProcessEnding()Aufruf antwortet, bevor der Serverprozess heruntergefahren wird. Typ: `FSimpleDelegate` Erforderlich: Ja

OnStartGameSession	Die Callback-Funktion, die Amazon GameLift aufruft, um eine neue Spielsitzung zu aktiviere n. Amazon GameLift ruft diese Funktion als Antwort auf eine Client-Anforderung aufCreateGameSession. Die Callback-Funktion übergibt ein GameSession Objekt, wie in der Amazon GameLift -API-Referenz definiert. Diese Eigenschaft ist eine Delegieru ngsfunktion, die als definiert ist. `DECLARE_D ELEGATE_OneParam(FOnStartGa meSession, Aws::GameLift::Ser ver::Model::GameSession);` Typ: `FOnStartGameSession` Erforderlich: Ja
OnUpdateGameSession	Die Callback-Funktion, die Amazon GameLift aufruft, um ein aktualisiertes Spielsitzungsobjek t an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anforderung verarbeitet wurde, um aktualisierte Matchmaker-Daten bereitzus tellen. Es übergibt ein GameSession Objekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID. Diese Eigenschaft ist eine Delegieru ngsfunktion, die als definiert ist. `DECLARE_D ELEGATE_OneParam(FOnUpdateG ameSession, Aws::GameLift::Ser ver::Model::UpdateGameSession);` Typ: `FOnUpdateGameSession` Required: No

Port	Die Portnummer, die der Serverprozess auf neue Player-Verbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersi tzungsobjekten enthalten, die die Spielsitz ungen bei der Verbindung mit einem Serverpro zess verwenden. Typ: `int` Erforderlich: Ja

UpdateGameSession

Dieser Datentyp wird auf ein Spielsitzungsobjekt aktualisiert, das den Grund für die Aktualisierung der Spielsitzung und die zugehörige Backfill-Ticket-ID enthält, wenn Backfill verwendet wird, um Spielersitzungen in der Spielsitzung zu füllen.

Eigenschaften	Beschreibung
GameSession	Ein von der Amazon GameLift API definiert es GameSession Objekt. Das -`GameSessi on` Objekt enthält Eigenschaften, die eine Spielsitzung beschreiben. Typ: `Aws::GameLift::Server::Game Session` Required: No
UpdateReason	Der Grund, warum die Spielsitzung aktualisiert wird. Typ: `enum class UpdateReason` • MATCHMAKING_DATA_UPDATED

Eigenschaften	Beschreibung
	• BACKFILL_FEHLGESCHLAGEN • BACKFILL_TIMED_OUT • BACKFILL_CANCELLED Required: No
BackfillTicketId	Die ID des Backfill-Tickets, das versucht, die Spielsitzung zu aktualisieren. Typ: `char[]` Required: No

GameSession

Dieser Datentyp enthält Details zu einer Spielsitzung.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für die Spielsitzung. Ein Spielsitzungs-ARN hat das folgende Format: `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` . Typ: `char[]` Required: No
Name	Eine beschreibende Bezeichnung der Spielsitzung. Typ: `char[]` Required: No

Eigenschaften	Beschreibung
FleetId	Eine eindeutige Kennung für die Flotte, auf der die Spielsitzung ausgeführt wird. Typ: `char[]` Required: No
MaximumPlayerSessionCount	Die maximale Anzahl von Spielerverbindungen zur Spielsitzung. Typ: `int` Required: No
Port	Die Portnummer für die Spielsitzung. Um eine Verbindung zu einem Amazon- GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: `int` Required: No
IpAddress	Die IP-Adresse der Spielsitzung. Um eine Verbindung zu einem Amazon- GameLift Spielserver herzustellen, benötigt eine App sowohl die IP-Adresse als auch die Portnummer. Typ: `char[]` Required: No

Eigenschaften	Beschreibung
GameSessionData	Eine Reihe von benutzerdefinierten Spielsitzungseigenschaften, die als einzelner Zeichenfolgenwert formatiert sind. Typ: `char[]` Required: No
MatchmakerData	Informationen über den Matchmaking-Prozess, der zum Erstellen der Spielsitzung verwendet wurde, in JSON-Syntax, formatiert als Zeichenfolge. Zusätzlich zur verwendeten Matchmaking-Konfiguration enthält sie Daten zu allen Spielern, die dem Match zugewiesen sind, einschließlich Spielerattributen und Teamzuweisungen. Typ: `char[]` Required: No
GameProperties	Eine Reihe von benutzerdefinierten Eigenschaften für eine Spielsitzung, formatiert als Schlüssel-Wert-Paare. Diese Eigenschaften werden mit einer Anfrage zum Starten einer neuen Spielsitzung übergeben. Typ: `GameProperty[]` Required: No

Eigenschaften	Beschreibung
DnsName	Die DNS-ID, die der Instance zugewiesen ist, die die Spielsitzung ausführt. Werte haben das folgende Format: • TLS-fähige Flotten: `<unique identifier>.<region identifier>.amazongamelift.com` . • Nicht TLS-fähige Flotten: `ec2-<unique identifier>.compute.amazonaws.com` . Wenn Sie eine Verbindung zu einer Spielsitzung herstellen, die auf einer TLS-fähigen Flotte ausgeführt wird, müssen Sie den DNS-Namen verwenden, nicht die IP-Adresse. Typ: `char[]` Required: No

FServerParameters

Informationen, die verwendet werden, um die Verbindung zwischen einem Amazon GameLift Anywhere-Server und dem Amazon- GameLift Service aufrechtzuerhalten. Diese Informationen werden beim Starten neuer Serverprozesse mit verwendetInitSDK(). Verwenden Sie für Server, die auf von Amazon GameLift verwalteten EC2-Instances gehostet werden, ein leeres Objekt.

Eigenschaften	Beschreibung
webSocketUrl	Amazon `GameLiftServerSdkEndpoint` GameLift gibt zurück, wenn Sie `RegisterCompute` für eine Amazon GameLift Anywhere-Rechenressource verwenden. Typ: `char[]`

Eigenschaften	Beschreibung
	Erforderlich: Ja
processId	Eine eindeutige Kennung, die für den Serverprozess registriert ist, der Ihr Spiel hostet. Typ: `char[]` Erforderlich: Ja
hostId	Die `HostID` wird `ComputeName` verwendet, als Sie Ihre Datenverarbeitung registriert haben. Weitere Informationen finden Sie unter RegisterCompute. Typ: `char[]` Erforderlich: Ja
fleetId	Die eindeutige Kennung der Flotte, bei der die Datenverarbeitung registriert ist. Weitere Informationen finden Sie unter RegisterCompute. Typ: `char[]` Erforderlich: Ja
authToken	Das von Amazon generierte Authentifizierungstoken GameLift , das Ihren Server bei Amazon authentifiziert GameLift. Weitere Informationen finden Sie unter GetComputeAuthToken. Typ: `char[]` Erforderlich: Ja

FStartMatchBackfillRequest

Informationen, die zum Erstellen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt diese Informationen Amazon GameLift in einem StartMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Kennung für Spielsitzungen. Die API-Operation gibt die Kennung im ARN-Format `GetGameSessionId` zurück. Typ: `char[]` Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung in Form eines ARN, die der Matchmaker für diese Anforderung verwenden soll. Der Matchmaker-ARN für die ursprüngliche Spielsitzung befindet sich im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Weitere Informationen zu Matchmaker-Daten finden Sie unter Arbeiten mit Matchmaker-Daten. Typ: `char[]` Erforderlich: Ja
Players	Eine Reihe von Daten, die alle Spieler darstellen, die sich in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Typ: `TArray<FPlayer>` Erforderlich: Ja
TicketId	Eine eindeutige Kennung für ein Matchmaking- oder Match-Backfill-Anforderungsticket.

Eigenschaften	Beschreibung
	Wenn Sie keinen Wert angeben, GameLift generiert Amazon einen. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen. Typ: `char[]` Required: No

FPlayer

Dieser Datentyp stellt einen Spieler im Matchmaking dar. Beim Starten einer Matchmaking-Anfrage hat ein Spieler eine Spieler-ID, Attribute und möglicherweise Latenzdaten. Amazon GameLift fügt Teaminformationen hinzu, nachdem ein Match durchgeführt wurde.

Eigenschaften	Beschreibung
LatencyInMS	Eine Reihe von Werten, die in Millisekunden ausgedrückt werden und die Latenz angeben, die ein Spieler erfährt, wenn er mit einem Standort verbunden ist. Wenn diese Eigenschaft verwendet wird, wird der Spieler nur mit den aufgelisteten Standorte n abgeglichen. Wenn ein Matchmaker eine Regel hat, die die Latenz der Spieler auswertet , müssen die Spieler die zu vergleichende Latenz melden. Typ: `TMap>FString, int32<` Required: No
PlayerAttributes	Eine Sammlung von Schlüssel-Wert-Paaren, die Spielerinformationen für die Verwendung im Matchmaking enthalten. Spielerattributschlüssel

Eigenschaften	Beschreibung
	müssen mit dem in einem Matchmaking-Regels atz PlayerAttributes verwendeten übereinst immen. Weitere Informationen zu Spielerattributen finden Sie unter AttributeValue. Typ: `TMap>FString, FAttribut eValue<` Required: No
PlayerId	Eine eindeutige Kennung für einen Spieler. Typ: `std::string` Required: No
Team	Der Name des Teams, dem der Spieler in einem Spiel zugewiesen ist. Sie definieren den Teamnamen im Matchmaking-Regelsatz. Typ: `FString` Required: No

FGameLiftDescribePlayerSessionsRequest

Ein Objekt, das angibt, welche Player-Sitzungen abgerufen werden sollen. Der Serverprozess stellt diese Informationen mit einem DescribePlayerSessions() Aufruf von Amazon bereit GameLift.

Eigenschaften	Beschreibung
GameSessionId	Eine eindeutige Kennung für Spielsitzungen. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitz ung anzufragen.

Eigenschaften	Beschreibung
	Das ID-Format für Spielsitzungen ist `FString`. ist `GameSessionID` eine benutzerdefinierte ID-Zeichenfolge oder ein Typ: `std::string` Required: No
PlayerSessionId	Die eindeutige Kennung für eine Player-Si tzung. Verwenden Sie diesen Parameter, um eine einzelne spezifische Player-Sitzung anzufordern. Typ: `FString` Required: No
PlayerId	Die eindeutige Kennung für einen Spieler. Verwenden Sie diesen Parameter, um alle Player-Sitzungen für einen bestimmten Player anzufordern. Siehe Generieren Sie Spieler-IDs. Typ: `FString` Required: No

Eigenschaften	Beschreibung
PlayerSessionStatusFilter	Der Status der Player-Sitzung, nach dem Ergebnisse gefiltert werden sollen. Zu den möglichen Player-Sitzungsstatus gehören: • RESERVED – Die Player-Sitzungsanf orderung wurde empfangen, aber der Player hat keine Verbindung zum Serverprozess hergestellt oder validiert. • ACTIVE – Der Player wurde vom Serverpro zess validiert und ist verbunden. • COMPLETED – Die Player-Verbindung wurde unterbrochen. • TIMEDOUT – Eine Player-Sitzungsanf orderung wurde empfangen, aber der Player hat keine Verbindung hergestellt oder wurde nicht innerhalb des Timeout-Limits (60 Sekunden) validiert. Typ: `FString` Required: No
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `FString` Required: No

Eigenschaften	Beschreibung
Limit	Die maximale Anzahl der auszugebenden Ergebnisse. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `int` Required: No

FStopMatchBackfillRequest

Informationen, die zum Abbrechen einer Matchmaking-Backfill-Anforderung verwendet werden. Der Spieleserver teilt diese Informationen dem Amazon- GameLift Service in einem StopMatchBackfill() Anruf mit.

Eigenschaften	Beschreibung
GameSessionArn	Eine eindeutige Spielsitzungs-ID der abgebroch enen Anfrage. Typ: `FString` Erforderlich: Ja
MatchmakingConfigurationArn	Eine eindeutige Kennung des Matchmakers, an den diese Anforderung gesendet wurde. Typ: `FString` Erforderlich: Ja
TicketId	Eine eindeutige Kennung des zu stornierenden Backfill-Anforderungstickets. Typ: `FString` Erforderlich: Ja

FAttributeValue

Verwenden Sie diese Werte in FPlayer Attribut-Schlüssel-Wert-Paaren. Mit diesem Objekt können Sie einen Attributwert mit einem der gültigen Datentypen angeben: Zeichenfolge, Zahl, Zeichenfolgen-Array oder Datenzuordnung. Jedes `AttributeValue` Objekt kann nur eine der verfügbaren Eigenschaften verwenden.

Eigenschaften	Beschreibung
attrType	Gibt den Typ des Attributwerts an. Typ: Ein `FAttributeType` Aufzählungswert. Required: No
S	Stellt einen Zeichenfolgenattributwert dar. Typ: `FString` Required: No
N	Stellt einen numerischen Attributwert dar. Typ: `double` Required: No
SL	Stellt ein Array von Zeichenfolgenattributwerten dar. Typ: `TArray<FString>` Required: No
SDM	Stellt ein Wörterbuch mit Zeichenfolgenschlüsseln und doppelten Werten dar. Typ: `TMap<FString, double>` Required: No

FGameLiftGetFleetRoleCredentialsRequest

Dieser Datentyp stellt Rollenanmeldeinformationen bereit, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen auf den Spieleserver ausweiten. Weitere Informationen finden Sie unter Einrichten einer IAM-Servicerolle für Amazon GameLift.

Eigenschaften	Beschreibung
RoleArn	Der Amazon-Ressourcenname (ARN) der Servicerolle, die den eingeschränkten Zugriff auf Ihre AWS Ressourcen erweitert. Typ: `FString` Required: No
RoleSessionName	Der Name der Sitzung, die die Verwendung der Rollenanmeldeinformationen beschreibt. Typ: `FString` Required: No

FGameLiftLongOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `long` Required: No
ResultWithOwnership	Das Ergebnis der -Aktion, umgewandelt in einen rvalue, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `long&&`

Eigenschaften	Beschreibung
	Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGameLiftStringOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: `FString` Required: No
ResultWithOwnership	Das Ergebnis der -Aktion, umgewandelt in einen rvalue, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `FString&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool`

Eigenschaften	Beschreibung
	Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGameLiftDescribePlayerSessionsOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “FGameLiftDescribePlayerSessionsResult” Required: No
ResultWithOwnership	Das Ergebnis der -Aktion, umgewandelt in einen rvalue, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `FGameLiftDescribePlayerSessionsResult&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war.

Eigenschaften	Beschreibung
	Typ: the section called “FGameLiftError” Required: No

FGameLiftDescribePlayerSessionsResult

Eigenschaften	Beschreibung
PlayerSessions	Typ: `TArray<FGameLiftPlayerSession>` Erforderlich: Ja
NextToken	Das Token, das den Beginn der nächsten Ergebnisseite angibt. Um den Anfang der Ergebnismenge anzugeben, geben Sie keinen Wert an. Wenn Sie eine Player-Sitzungs-ID angeben, wird dieser Parameter ignoriert. Typ: `FString` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGenericOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGameLiftPlayerSession

Eigenschaften	Beschreibung
CreationTime	Typ: `long` Erforderlich: Ja
FleetId	Typ: `FString` Erforderlich: Ja
GameSessionId	Typ: `FString` Erforderlich: Ja
IpAddress	Typ: `FString` Erforderlich: Ja
PlayerData	Typ: `FString`

Eigenschaften	Beschreibung
	Erforderlich: Ja
PlayerId	Typ: `FString` Erforderlich: Ja
PlayerSessionId	Typ: `FString` Erforderlich: Ja
Port	Typ: `int` Erforderlich: Ja
Status	Typ: Eine `PlayerSessionStatus` Aufzählung von . ??? Erforderlich: Ja
TerminationTime	Typ: `long` Erforderlich: Ja
DnsName	Typ: `FString` Erforderlich: Ja

FGameLiftGetComputeCertificateOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “FGameLiftGetComputeCertificateResult” Required: No

Eigenschaften	Beschreibung
ResultWithOwnership	Das Ergebnis der -Aktion, umgewandelt in einen rvalue, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `FGameLiftGetComputeCertificateResult&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGameLiftGetComputeCertificateResult

Der Pfad zum TLS-Zertifikat auf Ihrer Datenverarbeitung und der Hostname der Datenverarbeitung.

Eigenschaften	Beschreibung
CertificatePath	Typ: `FString` Erforderlich: Ja
ComputeName	Typ: `FString` Erforderlich: Ja

FGameLiftGetFleetRoleCredentialsOutcome

Dieser Datentyp stammt aus einer Aktion und erzeugt ein Objekt mit den folgenden Eigenschaften:

Eigenschaften	Beschreibung
Ergebnis	Das Ergebnis der Aktion. Typ: the section called “FGetFleetRoleCredentialsResult” Required: No
ResultWithOwnership	Das Ergebnis der -Aktion, umgewandelt in einen rvalue, sodass der aufrufende Code den Besitz des Objekts übernehmen kann. Typ: `FGameLiftGetFleetRoleCredentialsResult&&` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “FGameLiftError” Required: No

FGetFleetRoleCredentialsResult

Eigenschaften	Beschreibung
AccessKeyId	Die Zugriffsschlüssel-ID für die Authentif izierung und den Zugriff auf Ihre -AWSResso urcen. Typ: `FString` Required: No
AssumedRoleId	Die ID des Benutzers, zu dem die Servicerolle gehört. Typ: `FString` Required: No
AssumedRoleUserArn	Der Amazon-Ressourcenname (ARN) des Benutzers, zu dem die Servicerolle gehört. Typ: `FString` Required: No
Ablauf	Die Zeit bis zum Ablauf Ihrer Sitzungsa nmeldeinformationen. Typ: `FDateTime` Required: No
SecretAccessKey	Die ID des geheimen Zugriffsschlüssels für die Authentifizierung. Typ: `FString` Required: No

Eigenschaften	Beschreibung
SessionToken	Ein Token zur Identifizierung der aktuellen aktiven Sitzung, die mit Ihren -AWSRessourcen interagiert. Typ: `FString` Required: No
Herzlichen Glückwunsch	Ob die Aktion erfolgreich war oder nicht. Typ: `bool` Erforderlich: Ja
Fehler	Der Fehler, der auftrat, wenn die Aktion nicht erfolgreich war. Typ: the section called “GameLiftError” Required: No

FGameLiftError

Eigenschaften	Beschreibung
ErrorType	Der Fehlertyp. Typ: Eine `GameLiftErrorType` Aufzählung von . ??? Required: No
ErrorName	Der Name des Fehlers. Typ: `std::string` Required: No

Eigenschaften	Beschreibung
ErrorMessage	Die Fehlermeldung. Typ: `std::string` Required: No

Aufzählungen

Für das Amazon GameLift Server SDK (Unreal) definierte Enums sind wie folgt definiert:

FAttributeType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

Zeichenfolgenwert, der den Fehlertyp angibt. Gültige Werte sind:

- SERVICE_CALL_FAILED – Ein Aufruf an einen -AWSService ist fehlgeschlagen.
- LOCAL_CONNECTION_FAILED – Die lokale Verbindung zu Amazon GameLift ist fehlgeschlagen.
- NETWORK_NOT_INITIALIZED – Das Netzwerk wurde nicht initialisiert.
- GAMESESSION_ID_NOT_SET – Die Spielsitzungs-ID wurde nicht festgelegt.
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_AUSNAHME
- ALREADY_INITIALIZED – Der Amazon GameLift Server oder Client wurde bereits mit Initialize() initialisiert.
- FLEET_MISMATCH – Die Zielflotte stimmt nicht mit der Flotte einer gameSession oder playerSession überein.
- GAMELIFT_CLIENT_NOT_INITIALIZED – Der Amazon- GameLift Client wurde nicht initialisiert.

- GAMELIFT_SERVER_NOT_INITIALIZED – Der Amazon- GameLift Server wurde nicht initialisiert.
- GAME_SESSION_ENDED_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung beendet wurde.
- GAME_SESSION_NOT_READY – Die Amazon GameLift Server-Spielsitzung wurde nicht aktiviert.
- GAME_SESSION_READY_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass die Spielsitzung bereit ist.
- INITIALIZATION_MISMATCH – Eine Client-Methode wurde nach Server::Initialize() oder umgekehrt aufgerufen.
- NOT_INITIALIZED – Der Amazon GameLift Server oder Client wurde nicht mit Initialize() initialisiert.
- NO_TARGET_ALIASID_SET – Eine Ziel-aliasId wurde nicht festgelegt.
- NO_TARGET_FLEET_SET – Eine Zielflotte wurde nicht festgelegt.
- PROCESS_ENDING_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess endet.
- PROCESS_NOT_ACTIVE – Der Serverprozess ist noch nicht aktiv, nicht an ein gebunden GameSessionund kann weder akzeptieren noch verarbeiten PlayerSessions.
- PROCESS_NOT_READY – Der Serverprozess ist noch nicht bereit, aktiviert zu werden.
- PROCESS_READY_FAILED – Das Amazon GameLift Server SDK konnte den Service nicht kontaktieren, um zu melden, dass der Prozess bereit ist.
- SDK_VERSION_DETECTION_FAILED – SDK-Versionserkennung fehlgeschlagen.
- STX_CALL_FAILED – Ein Aufruf der XStx-Server-Backend-Komponente ist fehlgeschlagen.
- STX_INITIALIZATION_FAILED – Die XStx-Server-Backend-Komponente konnte nicht initialisiert werden.
- UNEXPECTED_SpeedER_SESSION – Der Server ist auf eine nicht registrierte Player-Sitzung gestoßen.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT

- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE – Abrufbares Senden einer Nachricht an den GameLift Service WebSocket.
- WEBSOCKET_SEND_MESSAGE_FAILURE – Es konnte keine Nachricht an den GameLift Service gesendet werden WebSocket.
- MATCH_BACKFILL_REQUEST_VALIDATION – Die Validierung der Anforderung ist fehlgeschlagen.
- microSDER_SESSION_REQUEST_VALIDATION – Die Validierung der Anforderung ist fehlgeschlagen.

EPlayerSessionCreationPolicy

Zeichenfolgenwert, der angibt, ob die Spielsitzung neue Spieler akzeptiert. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.
- NOT_SET – Die Spielsitzung ist nicht so eingestellt, dass neue Spielersitzungen akzeptiert oder abgelehnt werden.

EPlayerSessionStatus

- ACTIVE
- COMPLETED
- NOT_SET
- RESERVIERT
- TIMEDOUT

Referenz zum Amazon GameLift Unreal Engine-Server-SDK 3.x

Sie können diese Referenz zum Amazon GameLift Unreal Engine Server-SDK 3.x verwenden, um Ihr Multiplayer-Spiel für die Verwendung mit Amazon vorzubereiten. GameLift Einzelheiten zum Integrationsprozess finden Sie unterFüge Amazon GameLift zu deinem Gameserver hinzu.

Themen

- Amazon GameLift Server-SDK-Referenz für Unreal Engine: Aktionen
- Amazon GameLift Server-SDK-Referenz für Unreal Engine: Datentypen

Amazon GameLift Server-SDK-Referenz für Unreal Engine: Aktionen

Diese Amazon GameLift Server-SDK-Referenz kann Ihnen helfen, Ihre Unreal Engine-Spieleprojekte für die Verwendung mit Amazon GameLift vorzubereiten. Weitere Informationen zu dem Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Diese API ist in `GameLiftServerSDK.h` und `GameLiftServerSDKModels.h` definiert.

So richten Sie das Unreal Engine-Plug-in ein und zeigen Codebeispiele an Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt:

- Aktionen
- Datentypen

AcceptPlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spielersitzungs-ID eine Verbindung zum Serverprozess hergestellt hat und eine Überprüfung benötigt. Amazon GameLift überprüft, ob die Spielersitzungs-ID gültig ist, d. h., ob die Spieler-ID einen Spielerplatz in der Spielsitzung reserviert hat. Nach der Bestätigung GameLift ändert Amazon den Status des Spielerslots von RESERVED auf ACTIVE.

Syntax

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die vom GameLift Amazon-Dienst als Antwort auf einen Aufruf der GameLift Amazon-API-Aktion des AWS SDK ausgegeben wurde CreatePlayerSession. Der Spielclient verweist auf diese ID, wenn er sich mit dem Serverprozess verbindet.

Typ: FString

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

ActivateGameSession()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess eine Spielsitzung aktiviert hat und nun bereit ist, Spielerverbindungen zu empfangen. Diese Aktion muss im Rahmen der `onStartGameSession()`-Callback-Funktion aufgerufen werden, nachdem die Initialisierung der Spielsitzung vollständig abgeschlossen ist.

Syntax

```
FGameLiftGenericOutcome ActivateGameSession()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

DescribePlayerSessions()

Ruft Sitzungsdaten der Spieler, einschließlich Einstellungen, Metadaten zu der Sitzung und Spielerdaten ab. Verwenden Sie diese Aktion, um für eine einzelne Spielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen zu einer einzelnen Spieler-ID abzurufen.

Syntax

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
 FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameter

describePlayerSessionsAnfrage

Ein F DescribePlayerSessionsRequest-Objekt, mit dem beschrieben wird, welche Spielersitzungen abgerufen werden sollen.

Erforderlich: Ja

Rückgabewert

Wenn sie erfolgreich ausgeführt wird, gibt die Funktion ein F DescribePlayerSessionsRequest-Objekt mit einer Menge von Spielersitzungsobjekten zurück, die den Anforderungsparametern entsprechen. Player-Sitzungsobjekte haben eine Struktur, die mit dem GameLift PlayerSessionAmazon-API-Datentyp des AWS SDK identisch ist.

GetGameSessionId()

Ruft die ID der Spielsitzung ab, die derzeit von dem Serverprozess gehostet wird, wenn der Serverprozess aktiv ist.

Syntax

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die Spielsitzungs-ID als `FGameLiftStringOutcome`-Objekt zurück. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

GetInstanceCertificate()

Ruft den Speicherort eines PEM-kodierten TLS-Zertifikats ab, das der Flotte und ihren Instanzen zugeordnet ist. AWS Certificate Managergeneriert dieses Zertifikat, wenn Sie eine neue Flotte erstellen, wobei die Zertifikatskonfiguration auf GENERATED gesetzt ist. Verwenden Sie dieses Zertifikat, um eine sichere Verbindung mit einem Spiele-Client herzustellen und die Client-/Serverkommunikation zu verschlüsseln.

Syntax

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Bei Erfolg wird ein `GetInstanceCertificateOutcome` Objekt zurückgegeben, das den Speicherort der TLS-Zertifikatsdatei und der Zertifikatskette der Flotte enthält, die auf der Instanz gespeichert sind. Eine Stammzertifikatsdatei, die aus der Zertifikatskette extrahiert wurde, wird ebenfalls auf der Instanz gespeichert. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Weitere Informationen über das Zertifikat und die Zertifikatskettendaten finden Sie unter GetCertificateAntwortelemente in der AWS Certificate Manager API-Referenz.

GetSdkVersion()

Gibt die aktuelle Versionsnummer des SDK zurück, das in den Serverprozess integriert ist.

Syntax

```
FGameLiftStringOutcome GetSdkVersion();
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

War der Aufruf erfolgreich, gibt die Funktion die aktuelle SDK-Version als `FGameLiftStringOutcome`-Objekt zurück. Die zurückgegebene Zeichenfolge enthält nur die Versionsnummer (z. B. „3.1.5"). Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben.

Beispiel

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialisiert das Amazon GameLift SDK. Diese Methode sollte beim Start aufgerufen werden, bevor eine andere Initialisierung GameLift im Zusammenhang mit Amazon erfolgt.

Syntax

```
FGameLiftGenericOutcome InitSDK()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

ProcessEnding()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess heruntergefahren wird. Diese Methode sollte aufgerufen werden, nachdem alle anderen Aufgaben ausgeführt sind, und insbesondere auch alle aktiven Spielsitzungen beendet sind. Diese Methode sollte mit einem Beendigungscode 0 beendet werden. Andere Beendigungscodes sorgen für eine Ereignisnachricht zum unsauberen Beenden des Prozesses.

Syntax

```
FGameLiftGenericOutcome ProcessEnding()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

ProcessReady()

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess bereit ist, Spielesitzungen zu hosten. Rufen Sie diese Methode auf, nachdem Sie die Einrichtungsaufgaben erfolgreich aufgerufen InitSDK() und abgeschlossen haben, die erforderlich sind, bevor der Serverprozess eine Spielsitzung hosten kann. Diese Methode sollte nur einmal pro Prozess aufgerufen werden.

Syntax

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

Parameter

F ProcessParameters

Ein F ProcessParameters-Objekt, das die folgenden Informationen über den Serverprozess mitteilt:

- Namen der im Spieleservercode implementierten Callback-Methoden, die der GameLift Amazon-Dienst zur Kommunikation mit dem Serverprozess aufruft.
- Die Portnummer, auf der der Serverprozess horcht.
- Pfad zu allen spielsitzungsspezifischen Dateien, die Amazon erfassen und GameLift speichern soll.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Beispiel

Vgl. den Beispielcode in Verwendung des Unreal Engine-Plug-ins.

RemovePlayerSession()

Benachrichtigt den GameLift Amazon-Dienst, dass ein Spieler mit der angegebenen Spieler-Sitzungs-ID die Verbindung zum Serverprozess getrennt hat. Als Reaktion darauf GameLift ändert Amazon den Spielerplatz auf verfügbar, sodass er einem neuen Spieler zugewiesen werden kann.

Syntax

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

Parameter

playerSessionId

Eindeutige ID, die vom GameLift Amazon-Dienst als Antwort auf einen Aufruf der GameLift Amazon-API-Aktion des AWS SDK ausgegeben wurde CreatePlayerSession. Der Spielclient verweist auf diese ID, wenn er sich mit dem Serverprozess verbindet.

Typ: FString

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

StartMatchBackfill()

Sendet eine Anforderung zur Suche nach neuen Spielern für offene Slots in einer Spielsitzung, die mit FlexMatch erstellt wurde. Siehe auch die AWS SDK-Aktion StartMatchBackfill(). Mit dieser Aktion können Match-Backfill-Anforderungen von einem Game-Server-Prozess initiiert werden, der die Spielsitzung hostet. Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Diese Aktion ist asynchron. Wenn neue Spieler erfolgreich gematcht werden, liefert der GameLift Amazon-Service mithilfe der Callback-Funktion aktualisierte Matchmaker-Daten. `OnUpdateGameSession()`

Ein Serverprozess kann immer nur eine aktive Match-Backfill-Anforderung haben. Um eine neue Anforderung zu senden, rufen Sie zuerst StopMatchBackfill() auf, um die ursprüngliche Anforderung abzubrechen.

Syntax

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
 &startBackfillRequest);
```

Parameter

F StartMatchBackfillRequest

Ein F StartMatchBackfillRequest-Objekt, das die folgenden Informationen übermittelt:

- Eine Ticket-ID für die Zuordnung zur Backfill-Anforderung. Diese Information ist optional. Wenn keine ID angegeben GameLift wird, generiert Amazon automatisch eine.
- Der Matchmaker, an den die Anfrage gesendet werden soll. Der vollständige ARN der Konfiguration ist erforderlich. Dieser Wert kann aus den Matchmaker-Daten der Spielsitzung abgerufen werden.
- Die ID der Spielsitzung für das Backfill.
- Verfügbare Matchmaking-Daten für die aktuellen Spieler der Spielsitzung.

Erforderlich: Ja

Rückgabewert

Wenn erfolgreich, wird das Match-Backfill-Ticket als `FGameLiftStringOutcome`-Objekt zurückgegeben. Wenn die Funktion nicht erfolgreich ausgeführt wird, wird eine Fehlermeldung zurückgegeben. Der Ticketstatus kann mit der AWS SDK-Aktion DescribeMatchmaking() verfolgt werden.

StopMatchBackfill()

Bricht eine aktive mit dem Befehl StartMatchBackfill() erstellte Match-Backfill-Anforderung ab. Siehe auch die AWS SDK-Aktion StopMatchmaking(). Erfahren Sie mehr über die FlexMatchBackfill-Funktion.

Syntax

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

Parameter

StopMatchBackfillRequest

Ein F StopMatchBackfillRequest-Objekt, das das abzubrechende Matchmaking-Ticket identifiziert:

- Ticket-ID der abzubrechenden Backfill-Anforderung
- Matchmaker, an den die Backfill-Anforderung gesendet wurde
- Spielsitzung für die Backfill-Anforderung

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

TerminateGameSession()

Diese Methode ist mit Version 4.0.1 veraltet. Stattdessen sollte der Serverprozess ProcessEnding() nach dem Ende einer Spielsitzung aufgerufen werden.

Benachrichtigt den GameLift Amazon-Dienst, dass der Serverprozess die aktuelle Spielsitzung beendet hat. Diese Aktion wird aufgerufen, wenn der Serverprozess aktiv bleibt und bereit ist, eine neue Spielsitzung zu veranstalten. Es sollte erst aufgerufen werden, nachdem das Verfahren zur Beendigung der Spielsitzung abgeschlossen ist, da es Amazon signalisiertGameLift, dass der Serverprozess sofort für das Hosten einer neuen Spielsitzung verfügbar ist.

Diese Aktion wird nicht aufgerufen, wenn der Serverprozess nach Beendigung der Spielsitzung heruntergefahren wird. Rufen Sie stattdessen an, ProcessEnding() um zu signalisieren, dass sowohl die Spielsitzung als auch der Serverprozess beendet werden.

Syntax

```
FGameLiftGenericOutcome TerminateGameSession()
```

Parameter

Diese Aktion hat keine Parameter.

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

UpdatePlayerSessionCreationPolicy()

Aktualisiert die Kapazität der aktuellen Spielsitzung zur Aufnahme neuer Spielersitzungen. Eine Spielsitzung kann so eingerichtet werden, dass Sie alle neuen Spieler-Sitzungen akzeptiert oder ablehnt. (Siehe auch die `UpdateGameSession()`Aktion in der Amazon GameLift Service API-Referenz).

Syntax

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
 policy)
```

Parameter

Richtlinie

Wert, der angibt, ob die Spielsitzung neue Spieler akzeptiert.

Eingabe: `EPlayerSessionCreationPolicy` enum. Gültige Werte sind:

- ACCEPT_ALL – Akzeptiert alle neuen Spielersitzungen.
- DENY_ALL – Verwehrt neue Spielersitzungen.

Erforderlich: Ja

Rückgabewert

Gibt ein generisches Ergebnis zurück, das aus Erfolg oder Misserfolg mit einer Fehlermeldung besteht.

Amazon GameLift Server-SDK-Referenz für Unreal Engine: Datentypen

Diese Amazon GameLift Server-SDK-Referenz kann Ihnen helfen, Ihre Unreal Engine-Spieleprojekte für die Verwendung mit Amazon GameLift vorzubereiten. Weitere Informationen zu dem Integrationsprozess finden Sie unter Füge Amazon GameLift zu deinem Gameserver hinzu.

Diese API ist in `GameLiftServerSDK.h` und `GameLiftServerSDKModels.h` definiert.

So richten Sie das Unreal Engine-Plug-in ein und zeigen Codebeispiele an Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt:

- Aktionen
- Datentypen

F DescribePlayerSessionsRequest

Dieser Datentyp wird verwendet, um anzugeben, welche Spielersitzung(en) abgerufen werden sollen. Sie können ihn wie folgt verwenden:

- Geben Sie eine PlayerSessionId an, um eine bestimmte Spielersitzung anzufordern.
- Geben Sie einGameSessionId, um alle Spielersitzungen in der angegebenen Spielsitzung anzufordern.
- Geben Sie einPlayerId, um alle Spielersitzungen für den angegebenen Spieler anzufordern.

Verwenden Sie für große Sammlungen von Spielersitzungen die Paginierungsparameter, um Ergebnisse in aufeinander folgenden Blöcken abzurufen.

Inhalt

GameSessionId

Eindeutiger Bezeichner für die Spielsitzung. Verwenden Sie diesen Parameter, um alle Spielersitzungen für die angegebene Spielsitzung anzufragen. Das Format der Spielsitzungs-ID ist wie folgt: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. Der Wert von **<**ID-String**>** ist entweder eine benutzerdefinierte ID-Zeichenfolge (sofern bei der Erstellung der Spielsitzung angegeben) oder eine generierte Zeichenfolge.

Typ: Zeichenfolge

Required: No

Limit

Maximale Anzahl der zurückzugebenden Ergebnisse. Verwenden Sie diesen Parameter mit NextToken, um Ergebnisse als Satz aufeinanderfolgender Seiten zu erhalten. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Ganzzahl

Required: No

NextToken

Token, der den Beginn der nächsten Seite mit Ergebnissen anzeigt. Verwenden Sie den Token, der bei einem früheren Aufruf dieser Aktion zurückgegeben wurde. Um den Beginn des Ergebnissatzes anzugeben, geben Sie keinen Wert an. Wenn eine Spielersitzungs-ID angegeben ist, wird dieser Parameter ignoriert.

Typ: Zeichenfolge

Required: No

PlayerId

Eindeutiger Bezeichner für den Spieler. Player-IDs sind vom Entwickler definiert. Siehe Generieren Sie Spieler-IDs.

Typ: Zeichenfolge

Required: No

PlayerSessionId

Eindeutiger Bezeichner für eine Spielersitzung.

Typ: Zeichenfolge

Required: No

PlayerSessionStatusFilter

Status der Spielersitzung für die Filterung der Ergebnisse. Mögliche Spielersitzungsstatus sind u. a.:

- RESERVED – Die Anfrage nach einer Spielersitzung ist eingegangen, der Spieler ist jedoch noch nicht mit dem Serverprozess verbunden und/oder noch nicht validiert.
- ACTIVE – Der Spieler wurden vom Serverprozess validiert und ist derzeit verbunden.
- COMPLETED – Die Spielerverbindung wurde beendet.
- TIMEDOUT – Eine Anfrage nach einer Spielersitzung ist eingegangen, der Spieler hat jedoch keine Verbindung hergestellt und/oder wurde nicht innerhalb des Timeout-Zeitraums (60 Sekunden) validiert.

Typ: Zeichenfolge

Required: No

F ProcessParameters

Dieser Datentyp enthält den Satz von Parametern, die bei einem ProcessReady() Anruf an den GameLift Amazon-Dienst gesendet werden.

Inhalt

port

Nummer des Ports, den der Serverprozess für neue Spielerverbindungen überwacht. Der Wert muss innerhalb des Port-Bereich liegen, der für eine Flotte definiert wurde, die diesen Spiel-Server-Build verwendet. Diese Portnummer ist in Spielsitzungs- und Spielersitzungsobjekten enthalten, die die Spielsitzungen bei der Verbindung mit einem Serverprozess verwenden.

Typ: Ganzzahl

Erforderlich: Ja

logParameters

Objekt mit einer Liste der Verzeichnispfade zu den Spielsitzungsprotokolldateien.

Typ: TArray<FString>

Required: No

onStartGameSitzung

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um eine neue Spielsitzung zu aktivieren. Amazon GameLift ruft diese Funktion als Antwort auf die Kundenanfrage auf CreateGameSession. Die Callback-Funktion verwendet ein GameSessionObjekt (definiert in der Amazon GameLift Service API-Referenz).

Typ: F OnStartGameSession

Erforderlich: Ja

onProcessTerminate

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um das Herunterfahren des Serverprozesses zu erzwingen. Nach dem Aufruf dieser Funktion GameLift wartet Amazon fünf Minuten, bis der Serverprozess heruntergefahren ist, und antwortet mit einem ProcessEnding() Anruf, bevor der Serverprozess heruntergefahren wird.

Typ: F SimpleDelegate

Required: No

onHealthCheck

Name der Rückruffunktion, die der GameLift Amazon-Dienst aufruft, um einen Statusbericht vom Serverprozess anzufordern. Amazon GameLift ruft diese Funktion alle 60 Sekunden auf. Nach dem Aufruf dieser Funktion GameLift wartet Amazon 60 Sekunden auf eine Antwort. Wenn keine Antwort empfangen wird, zeichnet Amazon den Serverprozess als fehlerhaft auf.

Typ: F OnHealthCheck

Required: No

onUpdateGameSitzung

Name der Callback-Funktion, die der GameLift Amazon-Dienst aufruft, um ein aktualisiertes Spielsitzungsobjekt an den Serverprozess zu übergeben. Amazon GameLift ruft diese Funktion auf, wenn eine Match-Backfill-Anfrage bearbeitet wurde, um aktualisierte Matchmaker-Daten bereitzustellen. Es übergibt ein GameSessionObjekt, eine Statusaktualisierung (`updateReason`) und die Match-Backfill-Ticket-ID.

Typ: F OnUpdateGameSession

Required: No

F StartMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung zu versenden. Die Informationen werden in einem StartMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner für die Spielsitzung. Die API-Aktion GetGameSessionId() gibt den Bezeichner im ARN-Format zurück.

Typ: FString

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner in Form eines ARN für den Matchmaker, der für diese Anfrage verwendet werden soll. Um den Matchmaker zu finden, der für die Erstellung der ursprünglichen Spielsitzung

verwendet wurde, suchen Sie im Spielsitzungsobjekt in der Dateneigenschaft Matchmaker. Weitere Informationen zu Matchmaker-Daten finden Sie unter Mit Matchmaker-Daten arbeiten.

Typ: FString

Erforderlich: Ja

Players

Ein Datensatz, der alle Spieler repräsentiert, die sich gerade in der Spielsitzung befinden. Der Matchmaker verwendet diese Informationen, um nach neuen Spielern zu suchen, die zu den aktuellen Spielern passen. Eine Beschreibung des Player-Objektformats finden Sie im Amazon GameLift API-Referenzhandbuch. Um Spielerattribute, IDs und Teamzuweisungen zu finden, suchen Sie im Spielsitzungsobjekt in der Matchmaker-Dateneigenschaft. Wenn der Matchmaker die Latenz nutzt, erfassen Sie die aktualisierte Latenz für die aktuelle Region und nehmen Sie sie in die Daten jedes Spielers auf.

Typ: TArray<FPlayer>

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner für ein Matchmaking- oder Match-Backfill-Request-Ticket. Wenn hier kein Wert angegeben GameLift wird, generiert Amazon einen in Form einer UUID. Verwenden Sie diesen Bezeichner, um den Status des Backfill-Tickets zu verfolgen oder die Anfrage bei Bedarf abzubrechen.

Typ: FString

Required: No

F StopMatchBackfillRequest

Dieser Datentyp wird verwendet, um eine Matchmaking-Backfill-Anforderung abzubrechen.
Die Informationen werden in einem StopMatchBackfill() Anruf an den GameLift Amazon-Dienst übermittelt.

Inhalt

GameSessionArn

Eindeutiger Bezeichner der Spielsitzung für die Abbruchanfrage.

Typ: FString

Erforderlich: Ja

MatchmakingConfigurationArn

Eindeutiger Bezeichner des Matchmakers, an den diese Anfrage gesendet wurde.

Typ: FString

Erforderlich: Ja

TicketId

Eindeutiger Bezeichner des abzubrechenden Tickets für die Backfill-Anforderung.

Typ: FString

Erforderlich: Ja

Platzierungsveranstaltungen für Spielsitzungen

Amazon GameLift sendet Ereignisse für jede Platzierungsanfrage für eine Spielsitzung aus, während diese bearbeitet wird. Sie können diese Ereignisse in einem Amazon SNS SNS-Thema veröffentlichen, wie unter beschriebenRichten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein. Diese Ereignisse werden auch nahezu in Echtzeit und nach bestem Wissen und Gewissen an Amazon CloudWatch Events gesendet.

Dieses Thema beschreibt die Struktur der Platzierungsveranstaltungen für Spielsitzungen und bietet ein Beispiel für jeden Ereignistyp. Weitere Informationen zum Status von Anfragen zur Platzierung von Spielsitzungen finden Sie GameSessionPlacementin der Amazon GameLift API-Referenz.

Syntax für Platzierungsereign

Ereignisse werden als JSON-Objekte dargestellt. Die Struktur der CloudWatch Ereignisse entspricht dem Ereignismuster mit ähnlichen Feldern auf oberster Ebene und dienstspezifischen Details.

Zu den Feldern der obersten Ebene gehören die folgenden Felder (weitere Informationen finden Sie unter Ereignismuster):

version

Dieses Feld ist immer auf 0 (Null) gesetzt.

id

Eindeutige Tracking-ID für das Ereignis.

detail-type

Wert ist immer`GameLift Queue Placement Event`.

Quelle

Wert ist immer`aws.gamelift`.

Konto

Das AWS Konto, das zur Verwaltung von Amazon verwendet wird GameLift.

time

Zeitstempel des Ereignisses.

Region

Die AWS Region, in der die Platzierungsanfrage bearbeitet wird. Dies ist die Region, in der sich die aktuell verwendete Warteschlange für Spielsitzungen befindet.

Ressourcen

ARN-Wert der Warteschlange für die Spielsitzung, die die Platzierungsanfrage verarbeitet.

PlacementFulfilled

Die Platzierungsanfrage wurde erfolgreich erfüllt. Eine neue Spielsitzung wurde gestartet und für jeden Spieler, der in der Platzierungsanfrage für die Spielsitzung aufgeführt ist, wurden neue Spielersitzungen erstellt. Verbindungsinformationen für Spieler sind verfügbar.

Syntax im Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

port

Die Portnummer für die neue Spielsitzung.

gameSessionArn

Die ARN-ID für die neue Spielsitzung.

ipAddress

Die IP-Adresse der Spielsitzung.

DNS-Name

Die DNS-ID, die der Instanz zugewiesen wurde, auf der die neue Spielsitzung ausgeführt wird. Das Wertformat ist unterschiedlich, je nachdem, ob die Instanz, auf der die Spielsitzung ausgeführt wird, TLS-fähig ist. Wenn Spieler auf einer TLS-fähigen Flotte eine Verbindung zu einer Spielsitzung herstellen, müssen sie den DNS-Namen verwenden, nicht die IP-Adresse.

TLS-fähige Flotten:. `<unique identifier>.<region identifier>.amazongamelift.com`

Nicht TLS-fähige Flotten:. `ec2-<unique identifier>.compute.amazonaws.com`

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage erfüllt wurde.

gameSessionRegion

AWS Region der Flotte, die die Spielsitzung veranstaltet. Dies entspricht dem Regionstoken in der`GameSessionArn`.

placedPlayerSessions

Die Sammlung von Spielersitzungen, die für jeden Spieler in der Platzierungsanfrage für die Spielsitzung erstellt wurden.

Beispiel

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
```

```
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementFulfilled",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "port": "6262",
    "gameSessionArn": "arn:aws:gamelift:us-west-2::gamesession/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/4444dddd-55ee-66ff-77aa-8888bbbb99cc",
    "ipAddress": "98.987.98.987",
    "dnsName": "ec2-12-345-67-890.us-west-2.compute.amazonaws.com",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z",
    "gameSessionRegion": "us-west-2",
    "placedPlayerSessions": [
      {
        "playerId": "player-1"
        "playerSessionId": "psess-123213123232412412 3123"
      }
    ]
  }
}
```

PlacementCancelled

Die Platzierungsanfrage wurde mit einem Anruf beim GameLift Service storniert StopGameSessionPlacement.

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage storniert wurde.

Beispiel

```
{
```

```
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {

    "type": "PlacementCancelled",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

PlacementTimedOut

Die Platzierung der Spielsitzung wurde nicht erfolgreich abgeschlossen, bevor das Zeitlimit der Warteschlange abgelaufen ist. Die Platzierungsanfrage kann bei Bedarf erneut eingereicht werden.

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage storniert wurde.

Beispiel

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
```

```
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {

    "type": "PlacementTimedOut",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

PlacementFailed

Amazon GameLift konnte die Anfrage für die Spielsitzung nicht erfüllen. Dies wird in der Regel durch einen unerwarteten internen Fehler verursacht. Die Platzierungsanfrage kann bei Bedarf erneut eingereicht werden.

Detail:

Platzierungs-ID

Eine eindeutige Kennung, die der Platzierungsanfrage für die Spielsitzung zugewiesen wurde.

startTime

Zeitstempel, der angibt, wann diese Anfrage in die Warteschlange gestellt wurde.

endTime

Zeitstempel, der angibt, wann diese Anfrage fehlgeschlagen ist.

Beispiel

```
{
  "version": "0",
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",
  "detail-type": "GameLift Queue Placement Event",
```

```
  "source": "aws.gamelift",
  "account": "252386620677",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {

    "type": "PlacementFailed",
    "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

Generierung von GameLift Amazon-Preisschätzungen

Mit AWS Pricing Calculator können Sie eine Preisschätzung für Amazon erstellen GameLift. Sie benötigen keine AWS-Konto oder vertiefte Kenntnisse, AWS um den Taschenrechner verwenden zu können.

AWS Pricing CalculatorDer Rechner führt Sie durch die Entscheidungen, die sich auf die Servicekosten auswirken, um Ihnen eine Vorstellung davon zu geben, wie viel Amazon für Ihr Spieleprojekt kosten GameLift könnte. Wenn Sie sich noch nicht sicher sind, wie Sie Amazon verwenden möchtenGameLift, verwenden Sie die Standardwerte, um eine Schätzung zu erstellen. Bei der Planung der Nutzung in der Produktion kann Ihnen der Rechner dabei helfen, mögliche Szenarien zu testen und genauere Schätzungen zu erstellen.

Sie können AWS Pricing Calculator damit Schätzungen für die folgenden GameLift Amazon-Hosting-Optionen erstellen:

- Schätzen Sie das GameLift Amazon-Hosting
- Schätzen Sie Amazon GameLift Standalone FlexMatch

Schätzen Sie das GameLift Amazon-Hosting

Diese Option bietet einen Kostenvoranschlag für das Hosten Ihrer Spiele auf von Amazon GameLift verwalteten Servern, einschließlich der Kosten für die Nutzung der Serverinstanzen und die Datenübertragung. FlexMatchMatchmaking ist in den Kosten für Amazon GameLift Managed Hosting enthalten.

Wenn du Spielserver in mehr als einer AWS Region oder auf mehr als einem Instanztyp hostest oder planst, diese zu hosten, erstelle eine Schätzung für jede Region und jeden Instanztyp.

GameLiftAmazon-Instanzen

In diesem Abschnitt können Sie die Art und Anzahl der Rechenressourcen abschätzen, die Sie benötigen, um Spielsitzungen für Ihre Spieler zu veranstalten. Amazon GameLift verwendet Amazon Elastic Compute Cloud (Amazon EC2) -Instances zur Verwaltung von Spieleservern. In Amazon GameLift stellen Sie eine Flotte von Instances mit einem bestimmten Instance-Typ und Betriebssystem bereit. Wenn Sie mehrere Flotten haben oder planen, mehrere Flotten zu haben, erstellen Sie für jede Flotte eine Schätzung.

Öffnen Sie zunächst die GameLiftSeite Amazon konfigurieren vonAWS Pricing Calculator. Fügen Sie eine Beschreibung hinzu, wählen Sie eine Region aus und wählen Sie dann Estimate Amazon GameLift Hosting (Instance + Data Transfer Out). Füllen Sie unter GameLiftAmazon-Instances die folgenden Felder aus:

- Spitzenwert bei gleichzeitigen Spielern (Spitzen-CCU)

  Dies ist die maximale Anzahl von Spielern, die sich gleichzeitig mit Ihren Spielservern verbinden können. Dieses Feld gibt an, wie viel Hosting-Kapazität Amazon GameLift benötigt, um die Spitzennachfrage der Spieler zu decken. Gib die tägliche Höchstzahl an Spielern ein, die du erwartest, indem du Instanzen in deiner ausgewählten AWS Region verwendest.

  Wenn du beispielsweise 1.000 Spielern gleichzeitig eine Verbindung zu deinem Spiel ermöglichen möchtest, behalte den Standardwert von bei**1000**.

- Durchschnittliche CCU pro Stunde als Prozentsatz der täglichen Spitzen-CCU

  Dies ist die durchschnittliche Anzahl gleichzeitiger Spieler pro Stunde über einen Zeitraum von 24 Stunden. Wir verwenden diesen Wert, um abzuschätzen, wie viel Hosting-Kapazität Amazon für Ihre Spieler bereitstellen GameLift muss. Wenn Sie sich nicht sicher sind, welchen Prozentwert Sie verwenden sollen, behalten Sie den Standardwert **50** Prozent bei. Für Spiele mit stabiler Spielernachfrage empfehlen wir, einen Wert von **70** Prozent einzugeben.

  Wenn Ihr Spiel beispielsweise eine durchschnittliche Stunden-CCU von 6.000 und eine Spitzen-CCU von 10.000 hat, geben Sie den Wert in Prozent **60** ein.

- Spielsitzungen pro Instanz

  Dies ist die Anzahl der Spielesitzungen, die jede deiner Gameserverinstanzen gleichzeitig hosten kann. Zu den Faktoren, die sich auf diese Zahl auswirken können, gehören die Ressourcenanforderungen Ihres Spielservers, die Anzahl der Spieler, die in jeder Spielsitzung gehostet werden müssen, und die Leistungserwartungen der Spieler. Wenn Sie die Anzahl der gleichzeitigen Spielsitzungen für Ihr Spiel kennen, geben Sie diesen Wert ein. Sie können auch den Standardwert von beibehalten**20**.

- Spieler pro Spielsitzung

  Dies ist die durchschnittliche Anzahl von Spielern, die sich zu einer Spielsitzung verbinden, wie in Ihrem Spieldesign definiert. Wenn du Spielmodi mit unterschiedlichen Spielerzahlen hast, schätze die durchschnittliche Anzahl von Spielern pro Spielsitzung im gesamten Spiel. Der Standardwert ist **8**.

- Puffer im Leerlauf der Instanz%

  Dies ist der Prozentsatz der ungenutzten Hosting-Kapazität, die als Reserve für plötzliche Nachfragespitzen der Spieler bereitgehalten werden muss. Die Puffergröße ist ein Prozentsatz der Gesamtzahl der Instances in einer Flotte. Der Standardwert ist **`10`** Prozent.

  Bei einem Puffer von 20 Prozent Leerlauf verwaltet eine Flotte, die Spieler mit 100 aktiven Instances unterstützt, beispielsweise 20 inaktive Instances.
- Spot-Instanz%

  GameLiftAmazon-Flotten können eine Kombination aus On-Demand-Instances und Spot-Instances verwenden. On-Demand-Instances bieten zwar eine zuverlässigere Verfügbarkeit, Spot-Instances bieten jedoch eine äußerst kosteneffiziente Alternative. Wir empfehlen, eine Kombination zu verwenden, um sowohl die Kosteneinsparungen als auch die Verfügbarkeit zu optimieren. Informationen darüber, wie Amazon Spot-Instances GameLift verwendet, finden Sie unterOn-Demand-Instances versus Spot-Instances.

  Geben Sie für dieses Feld den Prozentsatz der Spot-Instances ein, die in einer Flotte verwaltet werden sollen. Wir empfehlen einen Spot-Instance-Prozentsatz zwischen 50 und 85 Prozent. Der Standardwert ist **`50`** Prozent.

  Wenn Sie beispielsweise eine Flotte mit 100 Instances bereitstellen und den **`40`** Prozentsatz angeben, verwaltet Amazon GameLift 60 On-Demand-Instances und 40 Spot-Instances.
- Instance-Typ

  GameLiftAmazon-Flotten können eine Reihe von Amazon EC2-Instance-Typen verwenden, die sich in Rechenleistung, Arbeitsspeicher, Speicher und Netzwerkfunktionen unterscheiden. Wenn Sie eine GameLift Amazon-Flotte konfigurieren, wählen Sie einen Instance-Typ, der den Anforderungen Ihres Spiels am besten entspricht. Informationen zur Auswahl eines Instance-Typs bei Amazon GameLift finden Sie unterAuswahl von GameLift Amazon-Rechenressourcen.

  Wenn Sie den Instance-Typ kennen, den Sie in Ihrer GameLift Amazon-Flotte verwenden oder verwenden möchten, wählen Sie diesen Typ. Wenn Sie sich nicht sicher sind, welchen Typ Sie wählen sollen, sollten Sie c5.large wählen. Dies ist ein Hochverfügbarkeitstyp mit durchschnittlicher Größe und durchschnittlichen Funktionen.
- Betriebssystem

  Dieses Feld gibt das Betriebssystem an, auf dem Ihre Spieleserver laufen — entweder Linux oder Windows. Der Standardwert ist Linux.

Ausgehende Datenübertragung (DTO)

In diesem Abschnitt können Sie die Kosten für den Traffic zwischen Ihren Spielclients und den Spielservern abschätzen. Datenübertragungsgebühren fallen nur für ausgehenden Datenverkehr an. Die eingehende Datenübertragung ist kostenlos.

Erweitern Sie auf der GameLiftSeite Amazon konfigurieren von AWS Pricing Calculator die Option Ausgehende Datenübertragung (DTO) und füllen Sie dann die folgenden Felder aus:

- Art der DTO-Schätzung

  Du kannst DTO auf eine der beiden folgenden Arten schätzen, je nachdem, wie du die Datenübertragung für dein Spiel verfolgst.

  - Pro Monat (in GB) — Wenn du den monatlichen Traffic deiner Spieleserver verfolgst, wähle diesen Typ.
  - Pro Spieler — Wenn Sie die Datenübertragung pro Spieler verfolgen, wählen Sie diesen Typ. Dies ist der Standardtyp.

    Im folgenden Feld schätzen Sie den DTO pro Spieler auf der Grundlage der Anzahl der Spielerstunden, die Sie im vorherigen Abschnitt berechnet haben.
- DTO pro Monat (in GB)

  Wenn Sie den DTO-Schätzungstyp Pro Monat (in GB) ausgewählt haben, geben Sie Ihre geschätzte monatliche DTO-Nutzung in GB für jede Instance pro Region ein.
- DTO pro Spieler

  Wenn du den DTO-Schätzungstyp Pro Spieler ausgewählt hast, gib die geschätzte DTO-Nutzung deines Spiels pro Spieler in KB/Sek ein. Der Standardwert ist **4**.

Wenn Sie mit der Konfiguration Ihrer GameLift Amazon-Preisschätzung fertig sind, wählen Sie Zu meiner Schätzung hinzufügen. Weitere Informationen zum Erstellen und Verwalten von Schätzungen in AWS Pricing Calculator finden Sie im AWS Pricing CalculatorBenutzerhandbuch unter Erstellen einer Schätzung, Konfiguration eines Dienstes und Hinzufügen weiterer Dienste.

Schätzen Sie Amazon GameLift Standalone FlexMatch

Diese Option bietet einen Kostenvoranschlag für die Nutzung von FlexMatch Matchmaking als eigenständigen Dienst und das Hosten Ihrer Spiele mit einer anderen Gameserverlösung. Dazu

gehören GameLift selbstverwaltetes Hosting von Amazon mit FleetIQ und lokales Hosting oder primitive peer-to-peer Datentypen für Cloud-Computing. Die FlexMatch Kosten für Standalone-Geräte richten sich nach der verwendeten Rechenleistung.

Wenn Sie mehr als einen Matchmaker in verschiedenen AWS Regionen haben oder planen, dies zu tun, erstellen Sie eine Schätzung für jede Region.

> **Note**
>
> Amazon GameLift FlexMatch ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Seoul), Asien-Pazifik (Sydney), Asien-Pazifik (Tokio), Europa (Frankfurt), Europa (Irland).

Öffnen Sie zunächst die GameLiftSeite Amazon konfigurieren vonAWS Pricing Calculator. Fügen Sie eine Beschreibung hinzu, wählen Sie eine Region aus und wählen Sie dann Estimate Amazon GameLift Standalone FlexMatch aus. Füllen Sie GameLift FlexMatch unter Amazon die folgenden Felder aus:

- Spitzenwert bei gleichzeitigen Spielern (Spitzen-CCU)

  Dies ist die maximale Anzahl von Spielern, die sich gleichzeitig mit Ihren Spielservern verbinden und Matchmaking anfordern können. Gib die tägliche Höchstzahl an Spielern ein, die du erwartest, in den Spielsitzungen in der von dir ausgewählten Region zu spielen.

  Wenn Sie beispielsweise bis zu 1.000 Spieler gleichzeitig treffen möchten, behalten Sie den Standardwert von bei**`1000`**.

- Durchschnittliche CCU pro Stunde als Prozentsatz der täglichen Spitzen-CCU

  Dies ist die durchschnittliche Anzahl gleichzeitiger Spieler pro Stunde über einen Zeitraum von 24 Stunden. Dieser Wert hilft bei der Schätzung des Umfangs Ihrer Matchmaking-Anfragen. Wenn Sie sich nicht sicher sind, welchen Prozentwert Sie verwenden sollen, behalten Sie den Standardwert **`50`** Prozent bei. Für Spiele mit stabiler Spielernachfrage empfehlen wir, einen Wert von **`70`** Prozent einzugeben.

  Wenn Ihr Spiel beispielsweise eine durchschnittliche Stunden-CCU von 6.000 und eine Spitzen-CCU von 10.000 hat, geben Sie den Wert in Prozent **`60`** ein.

- Anzahl der Spieler pro Spiel

Dies ist die durchschnittliche Anzahl von Spielern, die einer Spielsitzung entsprechen, wie in Ihrem Spieldesign definiert. Wenn du Spielmodi mit unterschiedlichen Spielerzahlen hast, schätze die durchschnittliche Anzahl von Spielern pro Spielsitzung im gesamten Spiel. Der Standardwert ist **8**.

- Spieldauer (in Minuten)

  Dies ist die durchschnittliche Zeit, die Spieler von Anfang bis Ende in einer Spielsitzung verbringen. Dieser Wert hilft zu bestimmen, wie oft Spieler möglicherweise ein neues Spiel benötigen. Geben Sie eine durchschnittliche Spieldauer in Minuten für Ihre Spieler ein. Der Standardwert ist **1**.

- Komplexität der Matchmaking-Regeln

  Die Komplexität der Matchmaking-Regeln bezieht sich auf die Anzahl und Art der Regeln, die Sie verwenden, um Spieler zusammenzubringen. Der Grad der Komplexität Ihres Regelsatzes hilft dabei, die Menge an Rechenleistung zu bestimmen, die für jedes Spiel benötigt wird.

  - Geringere Komplexität — Wählen Sie diese Option, wenn Ihr Matchmaking-Regelsatz nur wenige Regeln enthält, einfachere Regeltypen verwendet (z. B. Vergleichsregeln) und Regeln enthält, die erfolgreiche Matches mit weniger Versuchen ermöglichen.
  - Höhere Komplexität — Wählen Sie diese Option, wenn Ihr Matchmaking-Regelsatz mehrere Regeln enthält, komplexere Regeltypen verwendet (wie Entfernungs- oder Latenzregeln) und restriktive Regeln enthält, die zu mehr Fehlschlägen führen und mehr Abgleichsversuche erfordern.

  Weitere Informationen zur Regelkomplexität und Preisgestaltung finden Sie unter Amazon GameLift FlexMatch auf der GameLift Amazon-Preisseite.

Wenn Sie mit der Konfiguration Ihrer GameLift FlexMatch Amazon-Preisschätzung fertig sind, wählen Sie Zu meiner Schätzung hinzufügen. Weitere Informationen zum Erstellen und Verwalten von Schätzungen in AWS Pricing Calculator finden Sie im AWS Pricing CalculatorBenutzerhandbuch unter Erstellen einer Schätzung, Konfiguration eines Dienstes und Hinzufügen weiterer Dienste.

Kontingente und unterstützte Regionen

Informationen zu AWS GameLift Amazon-Servicekontingenten finden Sie unter GameLiftAmazon-Kontingente.

Informationen zum Beantragen von Kontingenterhöhungen für AWS Ressourcen finden Sie unter AWSServicekontingente.

Eine Liste der AWS-Regionen unterstützten GameLift Amazon-Regionen finden Sie unter GameLiftAmazon-Regionen.

GameLift Versionshinweise von Amazon

Die GameLift Amazon-Versionshinweise enthalten Einzelheiten zu neuen Funktionen, Updates und Korrekturen im Zusammenhang mit dem Service.

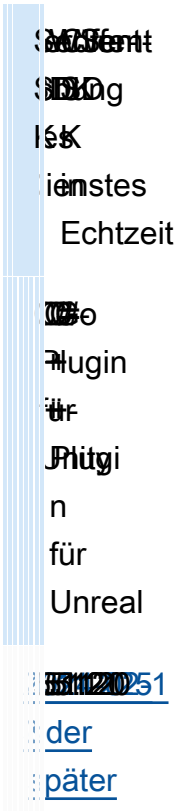
SDK-Versionen

In den folgenden Tabellen sind alle GameLift Amazon-Versionen mit SDK-Versionsinformationen aufgeführt. Es ist nicht erforderlich, vergleichbare SDKs für Ihre Spieleserver- und Client-Integrationen zu verwenden. Frühere Versionen eines SDK unterstützen jedoch möglicherweise nicht vollständig die neuesten Funktionen eines anderen.

Weitere Informationen zu Amazon GameLift SDKs finden Sie unterEntwicklungsunterstützung mit Amazon GameLift.

Die neuesten Amazon GameLift SDKs finden Sie auf der Amazon GameLift SDKs-Download-Website.

Aktuelle Version



Frühere Versionen

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
2023-12-14	1.11.225 oder später	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
02.11.2023	1.11.193 oder später	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
28.09.2023	1.11.144 oder später	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
17.08.2023	1.11.144 oder später	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
27.07.2023	1.11.111 oder später	5.1.0	5.1.0	5.0.2	5.0.0	1.2.0
29.06.2023	1.11.111 oder später		5.0.4	5.0.2	5.0.0	1.2.0

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
15.06.2023	1.11.87 oder später		5.0.4	5.0.2	5.0.0	1.2.0
25.05.2023	1.11.87 oder später		5.0.3	5.0.2	5.0.0	1.2.0
20.04.2023	1.11.63 oder später				5.0.0	1.2.0
13.04.2023	1.10.21 oder später				5.0.0	1.2.0
2023-02-09	1.10.21 oder später			3.4.0	5.0.0	1.2.0
2023-01-31	1.10.21 oder später			3.4.0	5.0.0	1.2.0
01.12.2022	1.10.21 oder später			3.4.0		1.2.0
25.08.2022	1.9.333 oder später		3.4.2	3.4.0		1.2.0
28.10.2021	1.9.133 oder später		3.4.2	3.4.0		1.2.0

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
03.06.2021	1.8.168 oder später		3.4.2	3.4.0		1.2.0
23.03.2021	1.8.168 oder später		3.4.1	3.3.3		1.1.0
16.03.2021	1.8.163 oder später		3.4.1	3.3.3		1.1.0
09.02.2021	1.8.139 oder später		3.4.1	3.3.3		1.1.0
22.12.2020	1.8.95 oder später		3.4.1	3.3.3		1.1.0
24.11.2020	1.8.95 oder später		3.4.1	3.3.2		1.1.0
11.11.2020	1.8.36 oder später		3.4.1	3.3.2		1.1.0
17.09.2020	1.8.36 oder später		3.4.1	3.3.2		1.1.0
2020-08-27	1.7.310 oder höher		3.4.0	3.3.1		1.1.0
16.04.2020	1.7.310 oder höher		3.4.0	3.3.1		1.1.0

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
02.04.2020	1.7.310 oder höher		3.4.0			1.1.0
19.12.2019	1.7.249 oder höher		3.4.0			1.1.0
14.11.2019	1.7.210 oder höher		3.4.0			1.1.0
2019-10-24	1.7.210 oder höher		3.4.0			1.1.0
2019-09-03	1.7.175 oder höher		3.4.0			1.1.0
2019-07-09	1.7.140 oder höher		3.3.0			1.0.0
2019-04-25	1.7.91 oder höher		3.3.0			1.0.0
07.03.2019	1.7.65 oder höher		3.3.0			
07.02.2019	1.7.45 oder höher		3.3.0			
2018-12-14	1.6.20 oder höher		3.3.0			

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
27.09.2018	1.6.20 oder höher		3.2.1			
14.06.2018	1.4.47 oder höher		3.2.1			
10.05.2018	1.4.47 oder höher		3.2.1			
2018-02-15	1.3.58 oder höher		3.2.1			
2018-02-08	1.3.52 oder höher		3.2.0			
01.09.2017	1.1.43 oder höher		3.1.7			
2017-08-16	1.1.31 oder höher		3.1.7			
16.05.2017	1.0.122 oder höher		3.1.5			
11.04.2017	1.0.103 oder höher		3.1.5			
2017-02-21	1.0.72 oder höher		3.1.5			

Veröffentlichung des Dienstes	AWS SDK	Server-SDK				Client SDK in Echtz
		C#-Plugin für Unity	C++	C++-Plugin für Unreal	Go	
18.11.2016	1.0.31 oder höher		3.1.0			
13.10.2016	1.0.17 oder höher		3.1.0			
2016-09-01	0.14.9 oder höher		3.1.0			
2016-08-04	0.12.16 oder höher		3.0.7			

Versionshinweise

Die folgenden Versionshinweise sind in chronologischer Reihenfolge, wobei die neuesten Updates zuerst aufgeführt sind. Amazon GameLift wurde erstmals 2016 veröffentlicht. Versionshinweise mit einem Datum vor den hier aufgeführten finden Sie unter den Links zum Veröffentlichungsdatum in SDK-Versionen.

24. April 2024: Amazon GameLift bringt Containerflotten auf den Markt

Amazon bietet jetzt GameLift eine Vorschau auf Containerflotten, die Ihnen verbesserte Portabilität, Skalierbarkeit, Fehlertoleranz und Agilität bieten.

In Containerflotten hosten Amazon EC2 EC2-Instances einen oder mehrere Ihrer Container. Zu diesen Containern gehören Ihr Spieleserver sowie alles, was er benötigt, einschließlich Abhängigkeiten und Konfigurationen. Beispiele für Abhängigkeiten sind SDKs und Softwarepakete. Nachdem Sie Ihren Container in Ihr privates Amazon Elastic Container Registry hochgeladen haben, GameLift füllt Amazon Ihre Flotte mit dem Container.

Um in einer Containerflotte zu funktionieren, muss Ihr Spieleserver unter Linux laufen und in das Server SDK 5.x integriert sein. In einer Containerflotte haben Sie eine fein abgestimmte Kontrolle über die Hosting-Ressourcen, sodass Sie den Verbrauch von Ressourcen wie CPU-Einheiten und Arbeitsspeicher optimieren können. Sie können auch mehrere Spieleserver in einem Container hosten, um den Ressourcenverbrauch zu reduzieren.

In einer Containerflotte erhalten Sie viele der gleichen Vorteile wie andere Flotten, wie z. B. On-Demand-Instance-Typen, Skalierung (automatisch und manuell), Warteschlangen und Spielersuche. Sie erhalten außerdem dieselben Kennzahlen wie bei anderen Flottenarten, zusätzlich zu einigen neuen Kennzahlen für Container. Mit Containerflotten erreichen Sie Spieler an folgenden Standorten und Regionen auf globaler Ebene:

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

Um noch mehr Regionen und lokale Zonen zu erreichen, erstellen Sie Containerflotten mit mehreren Standorten.

Weitere Informationen:

- Hosting-Verwaltung mit GameLift Amazon-Containern, Amazon GameLift Developer Guide
- CreateContainerGroupDefinition, Amazon GameLift API-Referenz

13. Februar 2024: Amazon GameLift führt Verbesserungen an SDKs ein und vereinfacht die Installation des GameLift Amazon-Plugins für Unreal Engine

Aktualisierte SDK-Versionen:

- Go Server SDK, Version 5.1.0
- C# Server-SDK, Version 5.1.2
- C++-Server-SDK, Version 5.1.2

Wir haben die folgenden Verbesserungen vorgenommen:

- Die Zuverlässigkeit des SDK wurde verbessert, indem eine automatische Wiederverbindung im Falle einer Netzwerkunterbrechung hinzugefügt wurde.
- [Go] Sie können jetzt `InitSDK()` mit oder ohne Serverparameter aufrufen. Spieleserver, die auf von Amazon GameLift verwalteten EC2-Flotten laufen, lesen die Serverparameter direkt aus Umgebungsvariablen. Spieleserver auf GameLift Anywhere Amazon-Flotten müssen `InitSDK()` mit Serverparametern aufgerufen werden.

Aktualisierte Plugin-Versionen:

- GameLift Amazon-Plugin für Unreal Engine, Version 1.1.0
- GameLift Amazon-Plugin für Unity, Version 2.1.0
- C++-Server-SDK-Plugin für Unreal, Version 5.1.1
- C# Server-SDK-Plugin für Unity, Version 5.1.2

Wir haben die folgenden Verbesserungen vorgenommen:

- [GameLift Amazon-Plugin für Unreal Engine] Die Installationsanweisungen wurden aktualisiert und die Verpackung vereinfacht. Dieses Plugin enthält jetzt die neueste Version des C++ Server SDK für Unreal.
- Die Plugins wurden aktualisiert, um die neueste Version des GameLift Server-SDK zu unterstützen.

Weitere Informationen:

- Integration von Spielen mit dem GameLift Amazon-Plugin für Unreal Engine, Amazon GameLift Developer Guide
- GameLift Amazon-Plugin- und SDK-Downloads

14. Dezember 2023: Amazon GameLift bietet die Möglichkeit, die Spieleigenschaften aktiver Spielsitzungen zu aktualisieren

Sie konnten bereits beim Erstellen von Spielsitzungen Spieleigenschaften festlegen und Spielsitzungen nach bestimmten Eigenschaften durchsuchen. Jetzt kannst du diese Eigenschaften auch in einer aktiven Spielsitzung hinzufügen und aktualisieren.

Zum Beispiel stimmen deine Spieler auf einer Karte ab, auf der sie spielen möchten. Ihr Spielclient ruft `UpdateGameSession` auf, um einen `GameProperty` Wert zu ändern`{"Key": "map", "Value":"jungle"}`. Ihr Spiel implementiert dann die neue Map für die Spieler in der Spielsitzung.

Mithilfe des `SearchGameSessions` Vorgangs können Spieladministratoren auch nützliche Daten aus den Spieleigenschaften abrufen. Administratoren können beispielsweise Spielsitzungen auflisten, die den `Status` Wert `ACTIVE` und diese Spieleigenschaft haben:`{"Key": "map", "Value":"desert"}`.

Weitere Informationen:

- the section called “Amazon GameLift zu einem Spieleclient hinzufügen”, GameLift Amazon-Entwicklerhandbuch
- GameProperty, Amazon GameLift API-Referenz
- UpdateGameSession, Amazon GameLift API-Referenz
- SearchGameSessions, Amazon GameLift API-Referenz

21. November 2023: Amazon GameLift startet Unterstützung für Infrastructure-as-Code-Tools wie Terraform und Pulumi powered by AWS Cloud Control API

Sie können jetzt Ihren gesamten GameLift Amazon-Ressourcenstapel mithilfe von Infrastructure as Code (IaC) -Tools verwalten. Zu diesen Tools gehören AWS CloudFormation auch Tools von Drittanbietern wie Terraform und Pulumi. Mit dieser zusätzlichen Unterstützung können Sie sich jetzt auf die Entwicklung Ihres Spiels konzentrieren und DevOps Strategien nutzen, um sich um das Ressourcenmanagement, CI/CD und die Bereitstellung für Ihre Kunden zu kümmern.

Sie können jetzt auch alle GameLift Amazon-Ressourcentypen mithilfe der AWS Cloud Control API bereitstellen und konfigurieren. Sie können weiterhin mit Ressourcen arbeiten, indem Sie die GameLift Amazon-APIs oder die AWS CloudFormation Vorlagen für Amazon verwenden GameLift.

Einzelheiten zu den GameLift Amazon-Ressourcen, die über IaC verfügbar sind, finden Sie in der GameLift Amazon-Ressourcentyp-Referenz GameLift Amazon-Ressourcentyp-Referenz.

Darüber hinaus können Sie Ihre Flotten jetzt mithilfe von AWS CloudFormation Vorlagen oder der AWS Cloud Control-API automatisch skalieren, indem Sie die neue Fleet-Eigenschaft verwenden:. `ScalingPolicies`

Die Cloud Control API bietet Entwicklern einen Standardsatz von APIs zum Erstellen, Lesen, Aktualisieren, Löschen und Auflisten von Ressourcen (CRUDL) für Hunderte von AWS Diensten und mehrere Tools von Drittanbietern wie Terraform und Pulumi.

Weitere Informationen:

- AWS CloudFormation
- AWS Cloud Control API
- AWS CC Terraform-Anbieter
- Pulumi

16. November 2023: Amazon GameLift aktualisiert eigenständiges Plugin für Unity

Aktualisierte SDK-Versionen: GameLift Amazon-Plugin für Unity, Version 2.0.0

Das GameLift Amazon-Plugin für Unity bietet Tools und Workflows, die die Schritte zur Inbetriebnahme Ihres Unity-Spiels für Cloud-Hosting bei Amazon GameLift vereinfachen. Amazon GameLift ist ein vollständig verwalteter Service, mit dem Spieleentwickler dedizierte Spieleserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können.

Mit dieser Version wird das Plugin für Unity aktualisiert, um die neuesten GameLift Amazon-Funktionen zu nutzen, einschließlich Server-SDK Version 5.x und Unterstützung für lokale Tests mit Amazon GameLift Anywhere. Das Plugin ist mit den Unity-Versionen Unity 2021.3 LTS und 2022.3 LTS kompatibel.

Zu den wichtigsten Funktionen des Plugins gehören:

- Geführte UI-Workflows im Unity-Editor für die folgenden Szenarien:
  - Testen Sie Ihre Spieleintegration mit Amazon, GameLift indem Sie Ihre lokale Workstation als Host verwenden. Dieser Workflow hilft Ihnen dabei, eine GameLift Anywhere Amazon-Flotte für Ihren lokalen Computer einzurichten, Instanzen Ihres Spieleservers und Clients zu starten, eine Spielsitzung über Amazon GameLift anzufordern und dem Spiel beizutreten.
  - Stellen Sie eine Cloud-Hosting-Lösung für Ihren integrierten Spieleserver mit Amazon GameLift Managed EC2 und unterstützenden AWS Ressourcen bereit. Dieser Workflow hilft Ihnen bei der Konfiguration Ihres Spiels für Cloud-Hosting und bietet drei Bereitstellungsszenarien:
    - Setze den Spieleserver auf einer einzigen Flotte ein.
    - Setze den Spielserver für eine Reihe kostengünstiger Spot-Flotten in mehreren AWS Regionen ein.

    - Stellen Sie den Spieleserver mit einem FlexMatch Matchmaker bereit.
- Möglichkeit, Benutzerprofile einzurichten, die auf einen AWS Kontobenutzer verweisen, und eine AWS Standardregion festzulegen. Sie können mehrere Profile verwalten, um in verschiedenen AWS Konten, Kontonutzern und Regionen zu arbeiten.
- Besondere Annehmlichkeiten, die dazu beitragen, die GameLift Integrations- und Bereitstellungsprozesse von Amazon zu optimieren, darunter:
    - Jede Hosting-Lösung umfasst unterstützende AWS Ressourcen, darunter einen Amazon Cognito Cognito-Benutzerpool, der eindeutige Spieler-IDs und Spielervalidierungen bereitstellt. Die Lösungen umfassen auch einen Amazon S3 S3-Bucket für Speicher, Amazon SNS SNS-Ereignisbenachrichtigungen, AWS Lambda Funktionen und andere Ressourcen.
    - Für den Anywhere Workflow automatisiert das Plugin die erforderlichen Serverparametereinstellungen.
    - Für den Amazon EC2 EC2-Workflow bietet jede Bereitstellungslösung einen integrierten Client-Backend-Service, der Lambda-Funktionen verwendet. Der Backend-Service befindet sich zwischen dem Spielclient und dem GameLift Amazon-Service und verwaltet alle direkten Anrufe an den GameLift Amazon-Service.
- Inhalt für Integrationstests, einschließlich Ressourcen und Code für ein einfaches Beispiel-Multiplayer-Spiel zur Veranschaulichung der Integration von Spieleserver und Spielclient.
- Plugin-Dokumentation mit ausführlichen Integrationsanleitungen und Beispielcode.

In allen Bereitstellungsszenarien, auch für EC2-Flotten Anywhere und Amazon EC2-Flotten, werden AWS CloudFormation Vorlagen verwendet, um die AWS Ressourcen für die Lösung Ihres Spiels zu beschreiben und bereitzustellen. Diese Vorlagen sind im GameLift Amazon-Plugin-Download enthalten. Sie können sie unverändert verwenden oder für Ihr Spiel anpassen.

Weitere Informationen:

- Leitfaden für das Amazon GameLift -Plugin für Unity für Server-SDK 5.x, GameLift Amazon-Entwicklerhandbuch
- Laden Sie das Plugin von herunter GitHub
- Über GameLift Amazon-Hosting
- GameLift Amazon-Forum

2. November 2023: Amazon GameLift fügt Unterstützung für gemeinsame Anmeldeinformationen hinzu

Aktualisierte SDK-Versionen: AWS SDK 1.11.193

Die neue Amazon-Funktion für GameLift gemeinsame Anmeldeinformationen ermöglicht es Anwendungen, die auf verwalteten EC2-Flotten bereitgestellt werden, mit anderen AWS Ressourcen zu interagieren. Dieses Update betrifft Anwendungen, die Sie zusammen mit Gameserver-Binärdateien bündeln und bereitstellen, die in das Server-SDK Version 5.x oder höher integriert sind. (Programmdateien von Spieleservern können bereits mithilfe der Server-SDK 5.x-Aktion Anmeldeinformationen anfordern.) `GetFleetRoleCredentials()`

Wenn Sie beispielsweise Ihren Gameserver-Build mit einem CloudWatch Amazon-Agenten bereitstellen möchten, um EC2-Instance-Metriken und andere Daten zu sammeln, benötigt der Agent die Erlaubnis, mit Ihren CloudWatch Ressourcen zu interagieren. Dazu müssen Sie zunächst eine AWS Identity and Access Management IAM-Rolle mit Berechtigungen zur Nutzung der CloudWatch Ressourcen einrichten und dann eine Flotte konfigurieren, für die die IAM-Rolle und die gemeinsamen Anmeldeinformationen aktiviert sind. Wenn Amazon Ihren Spieleserver-Build auf jeder EC2-Instance GameLift bereitstellt, generiert Amazon eine Datei mit gemeinsamen Anmeldeinformationen und speichert sie auf der Instance. Alle Anwendungen auf der Instance können die gemeinsamen Anmeldeinformationen verwenden. Amazon aktualisiert die temporären Anmeldeinformationen während der gesamten Lebensdauer der Instance GameLift automatisch.

Sie können gemeinsame Anmeldeinformationen aktivieren, wenn Sie eine verwaltete EC2-Flotte mit den folgenden Methoden erstellen:

- Im Workflow zur Flottenerstellung in der GameLift Amazon-Konsole.
- Beim Aufrufen der Amazon GameLift Service API-Operation `CreateFleet` mit dem neuen Parameter`InstanceRoleCredentialsProvider`.
- Beim Aufrufen der AWS CLI-Operation `aws gamelift create-fleet` mit dem Parameter`instance-role-credentials-provider`.

Weitere Informationen:

- Kommunizieren Sie mit anderen AWS Ressourcen aus Ihren Flotten, Amazon GameLift Developer Guide
- CreateFleet InstanceRoleCredentialsProvider, Amazon GameLift API-Referenz

- Eine IAM-Servicerolle einrichten, Amazon GameLift Developer Guide

28. September 2023: Amazon GameLift veröffentlicht neues eigenständiges Plugin für Unreal Engine

Aktualisierte SDK-Versionen: GameLift Amazon-Plugin für Unreal Engine Version 1.0.0

Das GameLift Amazon-Plugin für Unreal Engine bietet Tools und Workflows, die deine Schritte zur Inbetriebnahme eines Spiels mit Amazon GameLift für Cloud-Hosting vereinfachen. Amazon GameLift ist ein vollständig verwalteter Service, mit dem Spieleentwickler dedizierte Spieleserver für sitzungsbasierte Multiplayer-Spiele verwalten und skalieren können. Das Plugin unterstützt die UE-Versionen 5.0, 5.1 und 5.2. Zu den wichtigsten Funktion gehören:

- Geführte Benutzeroberflächenworkflows im Unreal Editor] führen Sie durch die folgenden Pfade:
  - Testen Sie Ihre Spieleintegration mit Amazon, GameLift indem Sie Ihre lokale Workstation als Host verwenden. Dieser Workflow hilft Ihnen dabei, eine GameLift Anywhere Amazon-Flotte für Ihren lokalen Computer einzurichten, Instanzen Ihres Spieleservers und Clients zu starten, eine Spielsitzung über Amazon GameLift anzufordern und Verbindungsinformationen für die neue Spielsitzung abzurufen.
  - Stellen Sie eine Amazon EC2 EC2-Cloud-Hosting-Lösung für Ihren integrierten Spieleserver bereit. Dieser Workflow hilft Ihnen bei der Konfiguration Ihres Spiels für Cloud-Hosting und bietet drei verschiedene Einsatzszenarien: Bereitstellung auf einer einzelnen Flotte, Bereitstellung auf einer Reihe von Spot-Flotten in mehreren Regionen oder Bereitstellung auf einer Gruppe von Flotten mit einem Matchmaker. FlexMatch Die Lösung für jedes Bereitstellungsszenario umfasst GameLift Amazon-Ressourcen und unterstützende AWS Ressourcen.
- Möglichkeit, Benutzerprofile einzurichten, die mit einem AWS Kontobenutzer verknüpft sind, und eine AWS Standardregion zu definieren. Sie können mehrere Profile verwalten, um in verschiedenen AWS Konten, Kontonutzern und Regionen zu arbeiten.
- Besondere Annehmlichkeiten, die dazu beitragen, die GameLift Integrations- und Bereitstellungsprozesse von Amazon zu optimieren, darunter:
  - Jede Hosting-Lösung umfasst unterstützende AWS Ressourcen, darunter einen grundlegenden Amazon Cognito Cognito-Benutzerpool, der eindeutige Spieler-IDs bereitstellt, einen Amazon S3-Bucket für Speicher, Amazon SNS-Ereignisbenachrichtigungen und AWS Lambda Funktionen.
  - Für den Anywhere Workflow automatisiert das Plugin die erforderlichen Serverparametereinstellungen mithilfe von Befehlszeilenargumenten.

  - Für den Amazon EC2 EC2-Workflow bietet jede Bereitstellungslösung einen integrierten Client-Backend-Service, der Lambda-Funktionen verwendet. Der Backend-Service empfängt Anfragen von Spieleclients und leitet sie an den GameLift Amazon-Service weiter.
- Inhalt für Integrationstests, darunter eine Starter-Spielkarte und zwei Testkarten mit grundlegenden Entwürfen und Benutzeroberflächenelementen.
- Plugin-Dokumentation mit ausführlichen Integrationsanleitungen und Beispielcode.

Alle Bereitstellungsszenarien, auch für Anywhere und Amazon EC2 EC2-Flotten, verwenden AWS CloudFormation Vorlagen zur Beschreibung der Lösungen. Das Plugin verwendet diese Vorlagen bei der Bereitstellung von GameLift Amazon-Ressourcen für Ihr Spiel. Diese Vorlagen sind im GameLift Amazon-Plugin-Download enthalten und können bearbeitet werden. Sie können sie unverändert verwenden oder für Ihr Spiel ändern.

Weitere Informationen:

- Integrieren von Spielen mit dem Amazon GameLift -Plugin für Unreal Engine, GameLift Amazon-Entwicklerhandbuch
- Laden Sie das Plugin von herunter GitHub
- Über GameLift Amazon-Hosting
- GameLift Amazon-Forum

17. August 2023: Amazon GameLift bietet Gameserver-Hosting mit AWS Graviton-Prozessoren an

Aktualisierte SDK-Versionen: AWS SDK 1.11.144

Mit Amazon können GameLift Sie Ihre Spiele jetzt mithilfe von EC2-Instances mit AWS Graviton-Prozessoren in der Cloud hosten. Graviton-Instances wurden AWS mit ARM64-basierten Prozessoren entwickelt und bieten das beste Preis-Leistungs-Verhältnis für Cloud-Workloads mit EC2, mit einer Verbesserung von bis zu 40% gegenüber vergleichbaren x86-basierten Instances. Die neuesten Graviton3-Prozessoren bieten eine um bis zu 25% bessere Rechenleistung als frühere Versionen.

Bei Amazon GameLift können Sie jetzt aus diesen neuen Instances der AWS Graviton-Familie wählen:

- Graviton2-basierte Instances: c6g, c6gn, r6g, m6g, g5g

- Graviton3-basierte Instanzen: c7g, r7g, m7g

Weitere Informationen:

- AWS Graviton-Prozessor: Erfahren Sie mehr über die Vorteile und praktischen Einsatzmöglichkeiten von Graviton-basierten EC2-Instances.
- Erste Schritte mit Graviton: Verschaffen Sie sich einen Überblick über die Graviton-basierten Instances und erfahren Sie, wie Anwendungen auf ihnen je nach Betriebssystem, Sprache und Laufzeit ausgeführt werden.

> **Note**
>
> Graviton Arm-Instances erfordern einen GameLift Amazon-Server, der auf einem Linux-Betriebssystem basiert. Server-SDK 5.1.1 oder neuer ist für C++ und C# erforderlich. Server-SDK 5.0 oder neuer ist für Go erforderlich. Diese Instances bieten keine out-of-the-box Unterstützung für die Mono-Installation auf Amazon Linux 2023 (AL2023) oder Amazon Linux 2 (AL2).

27. Juli 2023: Amazon GameLift veröffentlicht das Server-SDK 5.1.0 mit zusätzlicher Unterstützung für die Unity-Entwicklung

Aktualisierte SDK-Versionen: Server-SDK SDK for C++, C#/Unity, Unreal 5.1.0

Die neueste Version des Amazon GameLift Server SDK enthält Updates für C++, C# und das Unreal-Plugin sowie ein neues Plugin zur Verwendung mit der Unity-Game-Engine. Spieleentwickler integrieren das GameLift Amazon-Server-SDK in Spieleserver, die sie für das Hosting bei Amazon einsetzen GameLift.

Die neueste Server-SDK-Version enthält die folgenden Updates, die eine Reihe von Kundenanfragen beinhalten:

- Sprachspezifische SDK-Pakete herunterladen — Die aktualisierte GameLiftAmazon-Download-Website enthält SDK-Pakete für jede Sprache. Sie können aktuelle oder frühere Versionen herunterladen.
- Neues C#-Server-SDK-Plugin für Unity — Das neue Server-SDK-Paket für Unity enthält fertige C#-Bibliotheken, die Sie mit dem Paketmanager im Unity Editor installieren können (siehe den

neuen Unity-Integrationsleitfaden). Diese Bibliotheken enthalten die erforderlichen Abhängigkeiten durch. UnityNuGet Sie können dieses Plugin mit Unity 2020.3 LTS, 2021.3 LTS und 2022.3 LTS für Windows und Mac OS verwenden. Es unterstützt die Profile .NET Framework und .NET Standard von Unity mit.NET Standard 2.1 und .NET 4.x.

- Konsolidierte .NET-Lösung für C# — Das Server-SDK für C# unterstützt jetzt .NET Framework 4.6.2 (aktualisiert von 4.6.1) und.NET 6.0 in einer einzigen Lösung. .NET Standard 2.1 ist mit den von Unity erstellten Bibliotheken verfügbar.
- Server-SDK 5.1.0 aktualisiert
  - [C++, C#, Unreal] Sie können jetzt `InitSDK()` mit oder ohne Serverparameter aufrufen. Spieleserver, die auf von Amazon GameLift verwalteten EC2-Flotten laufen, lesen die Serverparameter direkt aus Umgebungsvariablen. Spieleserver auf GameLift Anywhere Amazon-Flotten müssen `InitSDK()` mit Serverparametern aufgerufen werden.
  - [C++, C#, Unreal] Bei Server-SDK-Aufrufen wurden die Fehlermeldungen verbessert.
  - [C++ SDK] Um die Build-Zeiten für das Server-SDK zu verkürzen, `-DRUN_CLANG_FORMAT` ist das Build-Flag standardmäßig deaktiviert. Sie können es mit aktivieren`-DRUN_CLANG_FORMAT=1`.
  - [C++ SDK] Beim Erstellen der Bibliotheken ohne die Standardbibliotheken (`-DGAMELIFT_USE_STD=0`) werden `InitSDK()` keine `std::` Datentypen mehr verwendet.
- Erweiterte Dokumentation zum Server-SDK 5.x
  - Aktualisierte Server-SDK-Referenzhandbücher für C++, C#/Unity und Unreal, einschließlich erweiterter Abdeckung aller Datentypen.
    - Amazon GameLift Server SDK 5.x-Referenz für C# und Unity
    - Amazon GameLift Server SDK 5.x-Referenz für C++
    - Referenz zum Amazon GameLift Unreal Engine-Server-SDK 5.x
  - Neue Versionen der Server-SDK 5-Integrationsleitfäden für Unity- und Unreal-Plugins
    - Integrieren Sie Amazon GameLift in ein Unity-Projekt
    - Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt
- Zusätzliche Aktualisierungen der Dokumentation
  - Überarbeitete Dokumentation für Amazon GameLift Service API-Operationen GetComputeAccessund GetInstanceAccesszur Erläuterung der Fernzugriffsverfahren auf der Grundlage der verwendeten Amazon GameLift Server-SDK-Version.

- Die Beschreibungen wurden überarbeitet GameSessionPlacement, um zu dokumentieren, dass Informationen zur Spielsitzung vorübergehend sind, wenn sich eine Platzierung im Status „ausstehend“ befindet.

13. Juli 2023: Amazon GameLift fügt Flottenhardwaremetriken hinzu

Sie können jetzt Hardware-Leistungskennzahlen für Ihre von Amazon GameLift verwalteten EC2-Flotten verfolgen. Zu den Metriken gehören EC2-Instance-Metriken für die CPU-Auslastung, das Netzwerkdatenverkehrsvolumen und die Lese-/Schreibaktivität der Festplatte. Für Amazon GameLift beschreiben diese Metriken alle aktiven Instances an einem Flottenstandort. Sie können diese Flottenhardwaremetriken mithilfe eines CloudWatch Amazon-Dashboards im anzeigen AWS Management Console. Sie können sie auch in der GameLift Amazon-Konsole unter Flottendetails anzeigen.

Weitere Informationen:

- Überwachen Sie Amazon GameLift mit Amazon CloudWatch(Metriken für Flotten), GameLift Amazon-Entwicklerhandbuch

29. Juni 2023: Amazon GameLift startet Unterstützung für Amazon Linux 2023

Aktualisierte SDK-Versionen: AWS SDK 1.11.111

GameLift Amazon-Kunden können jetzt das Betriebssystem Amazon Linux 2023 verwenden, um ihre Spieleserver zu hosten. AL2023 bietet mehrere Verbesserungen gegenüber AL2, einschließlich der Sicherheit. Dieses Betriebssystem ist in allen Regionen AWS-Regionen mit Ausnahme der chinesischen Regionen verfügbar.

Kunden können die neueren Linux-Betriebssysteme verwenden und weiterhin wichtige Sicherheitsupdates erhalten, wenn der Support für Amazon Linux (AL1) im Dezember 2023 endet. Die Support für Amazon Linux 2 wird bis 2025 fortgesetzt.

Weitere Informationen:

- Häufig gestellte Fragen zu Amazon GameLift Linux Server
- Vergleich von Amazon Linux 2 und Amazon Linux 2023
- Amazon GameLift API-Referenzlinks:
  - AWS SDK-Aktion `CreateBuild`

- CLI-Befehl `upload-build`
- CLI-Befehl `create-build`

25. Mai 2023: Amazon GameLift FleetIQ fügt einen Filter hinzu, um Platzierungen von Spielsitzungen auf Instances, die auslaufen, auszuschließen

Aktualisierte SDK-Versionen: SDK 1.11.87 AWS

Wenn Sie Amazon GameLift FleetIQ für das Hosten von Spielen verwenden, können Sie jetzt verhindern, dass Spielsitzungen auf Instances platziert werden, die derzeit ausgelastet sind. Instances, die auslaufen, sind zum Herunterfahren gekennzeichnet, können aber trotzdem ausgewählt werden, um neue Spielesitzungen zu hosten, wenn keine anderen Hosting-Ressourcen verfügbar sind. Mit dieser neuen Funktion kannst du die Verwendung von Instanzen, die das System belasten, komplett ausschließen.

Nutze diese Funktion, wenn du anrufst`ClaimGameServer`, um nach verfügbaren Spieleservern zu suchen. Fügen Sie den neuen `FilterOption` Parameter hinzu und setzen Sie den Status der erlaubten Instanzen auf Nur AKTIV. Als Reaktion darauf betrachtet Amazon GameLift FleetIQ nur aktive Instances, wenn nach einem verfügbaren Spieleserver gesucht und dieser beansprucht wird.

Weitere Informationen:

- ClaimGameServerin der Amazon GameLift API-Referenz
- So funktioniert FleetIQ im Amazon GameLift FleetIQ Developer Guide

16. Mai 2023: Amazon GameLift unterstützt die Kennzeichnung der Kostenzuweisung für Flotten

GameLift Amazon-Kunden können jetzt AWS Billing Kostenzuweisungs-Tags verwenden, um ihre Hosting-Kosten für Spiele zu organisieren. Sie können einzelnen Amazon GameLift EC2-Flottenressourcen Kostenzuweisungs-Tags zuweisen, um zu verfolgen, wie Ihre Flotten zu den gesamten Hosting-Kosten beitragen.

Weitere Informationen:

- Verwalten der Kosten für das Hosten von Spielen
- Verwendung von AWS Kostenzuweisungs-Tags, Benutzerhandbuch AWS Billing

20. April 2023: Amazon GameLift startet Unterstützung für Windows Server 2016

Aktualisierte SDK-Versionen: AWS SDK 1.11.63

GameLift Amazon-Kunden können jetzt das Betriebssystem Windows Server 2016 verwenden, um ihre Spieleserver zu hosten. Dieses Betriebssystem ist in allen Versionen verfügbar AWS-Regionen. Kunden können das neuere Windows-Betriebssystem verwenden und weiterhin wichtige Sicherheitsupdates erhalten, da Microsoft seinen Support für Windows Server 2012 im Oktober 2023 einstellt.

Ab heute müssen Neukunden, die eine Windows-Laufzeitumgebung benötigen, Windows Server 2016 angeben, wenn sie neue Gameserver-Builds für das Hosting erstellen. Bestandskunden können weiterhin neue Builds und Flotten mit Windows Server 2012 erstellen, müssen jedoch die Migration mit Windows Server 2016 vor dem Ende des Supports von Microsoft am 10. Oktober 2023 abschließen.

Dieses Update beinhaltet die folgenden Serviceänderungen:

- Wenn Sie einen Gameserver-Build mit Amazon GameLift SDK- oder CLI-Befehlen erstellen, müssen Sie das Betriebssystem jetzt explizit festlegen. Es gibt keinen Standardwert mehr. Verwenden Sie den Wert, um Ihren Spieleserver auf Windows Server 2016 bereitzustellen`WINDOWS_2016`.
- Wenn Sie mit der GameLift Amazon-Konsole einen Gameserver-Build erstellen, müssen Sie ein Betriebssystem aus den verfügbaren Werten auswählen. Wenn Sie bereits Kunde mit aktiven Windows Server 2012-Flotten sind, können Sie entweder `WINDOWS_2012` oder `WINDOWS_2016` wählen.

Weitere Informationen:

- Amazon GameLift API-Referenzlinks:
  - CLI-Befehl `upload-build`
  - CLI-Befehl `create-build`
  - AWS SDK-Aktion `CreateBuild`
- GameLift Häufig gestellte Fragen zu Amazon für Windows 2012

13. April 2023: Amazon GameLift bringt das Server-SDK 5.x für Unreal auf den Markt

Aktualisierte SDK-Versionen: Server SDK 5.0.0 für Unreal

Die neueste Version des Amazon GameLift Lightweight Plug-ins für Unreal Engine basiert jetzt auf dem Amazon GameLift Server SDK 5.x. Um mit der Integration Ihrer Unreal Engine-Umgebung in Amazon zu beginnen, klicken GameLift Sie auf die folgenden Links.

Weitere Informationen:

- Integrieren Sie Amazon GameLift in ein Unreal Engine-Projekt
- Füge Amazon GameLift zu deinem Gameserver hinzu
- Amazon GameLift Server SDK 5.x-Referenz für C++

14. März 2023: Amazon GameLift bringt ein neues Konsolenerlebnis auf den Markt

Die neue GameLift Amazon-Konsole beinhaltet die folgenden Verbesserungen:

- Verbesserte Navigation — Der neue Navigationsbereich erleichtert die Navigation zwischen GameLift Amazon-Ressourcen.
- GameLift Amazon-Landingpage — Die neue Landingpage enthält Links zu hilfreicher Dokumentation, bietet einen allgemeinen Überblick über Amazon GameLift und bietet Unterstützung durch Links zur Dokumentation, häufig gestellte Fragen und AWS re:Post.
- Verbesserte CloudWatch Amazon-Metriken — GameLift Amazon-Metriken sind jetzt sowohl in der GameLift Amazon-Konsole als auch in Ihren CloudWatch Dashboards verfügbar. Dieses Update enthält auch neue Messwerte für Leistung, Auslastung und Spielersitzungen.

Weitere Informationen:

- Deine Spieldaten auf der Konsole ansehen
- Verwaltung der GameLift Amazon-Hosting-Ressourcen
- Einen FlexMatch Matchmaker aufbauen

14. Februar 2023: Amazon unterstützt GameLift jetzt serverseitige Verschlüsselung für Amazon SNS SNS-Themen

Die serverseitige Verschlüsselung (SSE) für SNS-Themen verschlüsselt Ihre vertraulichen Daten im Ruhezustand. SSE verwendet AWS Key Management Service (AWS KMS) -Schlüssel, um den Inhalt Ihrer SNS-Themen zu schützen.

Weitere Informationen:

- Richten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein
- FlexMatchMatchmaking-Ereignisse
- Verschlüsselung im Ruhezustand

9. Februar 2023: Das GameLift Amazon-Server-SDK unterstützt.NET 6 mit C #10

Aktualisierte SDK-Versionen: Server-SDK 5.0.0 für .NET 6. Es sind keine SDK-Updates erforderlich.

Wenn Sie die Unity Real-Time Development Platform verwenden, verwenden Sie weiterhin das GameLift Amazon-Server-SDK 5.0.0 mit.NET 4.6. Unity unterstützt.NET 6 nicht.

Weitere Informationen:

- Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift Getting Started herunter
- Amazon GameLift Server SDK 5.x-Referenz für C# und Unity

31. Januar 2023: Das Amazon GameLift Server SDK unterstützt die Go-Sprache

Aktualisierte SDK-Versionen: Server-SDK 5.0.0 für Go

Weitere Informationen:

- Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift Getting Started herunter
- Amazon GameLift Server-SDK-Referenz für Go

1. Dezember 2022: Amazon GameLift bringt Amazon GameLift Anywhere und Amazon GameLift Server SDK 5.0 auf den Markt

Aktualisierte SDK-Versionen: AWS SDK 1.10.21, Server-SDK 5.0.0 für C++ und C#

Amazon GameLift Anywhere verwendet Ihre Spieleserver-Ressourcen, um GameLift Amazon-Spieleserver zu hosten. Sie können Amazon verwenden GameLift Anywhere, um Ihre eigenen Rechenressourcen in Amazon GameLift Managed EC2 Compute zu integrieren, um Ihre Spieleserver auf mehrere Rechenarten zu verteilen. Sie können Amazon auch verwenden GameLift Anywhere, um Ihre Spieleserver iterativ zu testen, ohne den Build GameLift für jede Iteration auf Amazon hochladen zu müssen.

Höhepunkte:

- Neue GameLift Anywhere Amazon-Flotten- und Compute-Typen
- Registrierung Amazon GameLift Anywhere Amazon-Rechenressourcen
- Verbesserter Test-Iterationszyklus

Amazon GameLift Server SDK 5.0.0 führt Verbesserungen am bestehenden Server-SDK und einen neuen Ressourcentyp, Compute, ein. Das Server-SDK 5.0.0 unterstützt Amazon GameLift Anywhere und die Verwendung Ihrer eigenen Rechenressourcen für das Hosting von Spieleservern.

Weitere Informationen:

- Referenz zum Amazon GameLift Server-SDK
- Standort der Flotte
- Auswahl von GameLift Amazon-Rechenressourcen
- Erstellen Sie eine GameLift Anywhere Amazon-Flotte

25. August 2022: Amazon GameLift startet Unterstützung für Local Zones

Aktualisierte SDK-Versionen: AWS SDK 1.9.333

Amazon GameLift ist jetzt in acht Local Zones in den Vereinigten Staaten verfügbar, sodass Sie Ihre Flotten näher an den Spielern einsetzen können. Sie können alle verwalteten GameLift Amazon-Funktionen mit Local Zones nutzen, indem Sie die Local Zones zu Ihren Flotten hinzufügen.

Local Zones erweitern AWS Ressourcen und Dienste bis an den Rand der Cloud, in der Nähe von Zentren mit großer Bevölkerung, Industrie und Informationstechnologie (IT). Das bedeutet, dass Sie Anwendungen, die eine Latenz im einstelligen Millisekundenbereich erfordern, näher an Endbenutzern oder lokalen Rechenzentren bereitstellen können.

Weitere Informationen:

- Local Zones
- Standort der Flotte
- Erstellen Sie eine von Amazon GameLift verwaltete Flotte

28. Juni 2022: Amazon GameLift bringt ein neues Opt-In-Konsolenerlebnis auf den Markt

Die neue GameLift Amazon-Konsole beinhaltet die folgenden Verbesserungen:

- Verbesserte Navigation — Der neue Navigationsbereich erleichtert die Navigation zwischen GameLift Amazon-Ressourcen.
- GameLift Amazon-Landingpage — Die neue Landingpage enthält Links zu hilfreicher Dokumentation, bietet einen allgemeinen Überblick über Amazon GameLift und bietet Unterstützung durch Links zur Dokumentation, häufig gestellte Fragen und AWS re:Post.
- Verbesserte CloudWatch Amazon-Metriken — GameLift Amazon-Metriken sind jetzt sowohl in der GameLift Amazon-Konsole als auch in Ihren CloudWatch Dashboards verfügbar. Dieses Update enthält auch neue Messwerte für Leistung, Auslastung und Spielersitzungen.

Weitere Informationen:

- Deine Spieldaten auf der Konsole ansehen
- Verwaltung der GameLift Amazon-Hosting-Ressourcen
- Einen FlexMatch Matchmaker aufbauen

15. Februar 2022: FlexMatch Fügt eine zusammengesetzte Regel und weitere Verbesserungen hinzu

FlexMatch Benutzer haben jetzt Zugriff auf die folgenden Funktionen:

- Zusammengesetzte Regel — Unterstützung für zusammengesetzte Matchmaking-Regeln für Spiele mit 40 oder weniger Spielern hinzugefügt. Sie können jetzt logische Anweisungen verwenden, um eine zusammengesetzte Regel zu erstellen, um ein Spiel zu bilden. Ohne eine zusammengesetzte Regel in Ihrem Regelsatz müssen alle Regeln im Regelsatz wahr sein, um eine Übereinstimmung zu bilden. Bei zusammengesetzten Regeln können Sie mithilfe der folgenden logischen Operatoren auswählen, welche Regeln angewendet werden sollen: `and`or`,`not`, und`xor`.
- Flexible Teamauswahl — Die Eigenschaftsausdrücke für das Matchmaking wurden aktualisiert und unterstützen nun die Auswahl einer Teilmenge aller verfügbaren Teams.
- Längere Zeichenkettenlisten — Die maximale Anzahl von Zeichenketten in einer Liste von Zeichenketten mit Spielerattributwerten wurde von 10 auf 100 erhöht.

Weitere Informationen:

- GameLift FlexMatch Amazon-Entwicklerhandbuch:
  - FlexMatch Regeltypen
  - FlexMatch Eigenschaftsausdrücke
- AttributeValue: SL

28. Oktober 2021: Amazon GameLift bietet Unterstützung für Flotten mit mehreren Regionen in der Region Asien-Pazifik (Osaka); Amazon GameLift FleetIQ fügt Unterstützung für Graviton2-Prozessoren hinzu AWS

Aktualisierte SDK-Versionen: SDK 1.9.133 AWS

Amazon GameLift ist jetzt in der Region Asien-Pazifik (Osaka) verfügbar. Spieleentwickler können jetzt Instances in Osaka mithilfe einer Flotte aus GameLift mehreren Regionen bereitstellen.

Sie können jetzt von Graviton2 gehostete Spieleserver verwenden, die auf der ARM-basierten Prozessorarchitektur basieren, um im Vergleich zu den entsprechenden Intel-basierten Rechenoptionen eine höhere Leistung zu geringeren Kosten zu erzielen.

Höhepunkte:

- Amazon GameLift ist jetzt in der Region Asien-Pazifik (Osaka) verfügbar.
- Amazon GameLift FleetIQ Gameservergruppen können jetzt für die Verwaltung der Graviton2-Instance-Familien c6g, m6g und r6g konfiguriert werden.

Weitere Informationen:

- GameLift Amazon-Flotte mit mehreren Regionen
- CreateGameServerGroup
- AWS Graviton-Prozessor

20. September 2021: Amazon GameLift veröffentlicht Plugin für Unity

Das GameLift Amazon-Plugin für Unity Version 1.0.0 enthält Bibliotheken und eine native Benutzeroberfläche, die den Zugriff auf GameLift Amazon-Ressourcen und die Integration von Amazon GameLift in Ihr Unity-Spiel erleichtert. Sie können das GameLift Amazon-Plugin für Unity verwenden, um auf GameLift Amazon-APIs zuzugreifen und AWS CloudFormation Vorlagen für gängige Spieleszenarien bereitzustellen. Das Plugin enthält auch ein Beispielspiel, das mit den Beispielszenarien funktioniert. Sie können Amazon GameLift Local verwenden, um Nachrichten zu sehen, die zwischen dem Spieleclient und dem Spieleserver ausgetauscht werden, um zu erfahren, wie ein typisches Spiel mit Amazon GameLift interagiert.

Das Plugin für Unity unterstützt Unity 2019.4 LTS und 2020.3 LTS.

Höhepunkte:

- Erstelle ein Beispielspiel mit verschiedenen Szenarien, führe es aus und modifiziere es oder erstelle dein eigenes.
- Stellen Sie AWS CloudFormation Beispielszenarien für typische Spielszenarien bereit, darunter nur Authentifizierung, Flotte mit einer Region, Flotten mit mehreren Regionen mit Warteschlange und benutzerdefiniertem Matchmaker, Spot-Flotten mit Warteschlange und benutzerdefiniertem Matchmaker und. FlexMatch

Weitere Informationen:

- Integration von Spielen mit dem GameLift Amazon-Plugin für Unity

30. Juni 2021: FlexMatch fügt die BatchDistance-Regel hinzu

Sie können den Regeltyp BatchDistance verwenden, um eine Zeichenfolge oder ein numerisches Attribut anzugeben, was jedem Segment eine Vielzahl von Vorteilen bietet.

Höhepunkte:

- Bei großen Spielen (>40 Spieler) könnt ihr jetzt dieselbe Balance auf Basis von Fertigkeiten, Modi und Karten erreichen, anstatt die Spieler gleichmäßig nur nach ihren Fähigkeiten auszubalancieren. Stellt sicher, dass sich alle Spieler in einer Fähigkeitsgruppe befinden, gruppiert mehrere numerische Attribute wie Liga oder Spielstil und gruppiert nach Zeichenkettenattributen wie Karte oder Spielmodus. Du kannst im Laufe der Zeit auch Erweiterungen erstellen. Du kannst zum Beispiel eine Erweiterung erstellen, um einem größeren Spielniveau die Teilnahme am Spiel zu ermöglichen, je länger der Spieler wartet.

  Für Spiele mit weniger als 40 Spielern können Sie einen neuen vereinfachten Regelausdruck verwenden.

3. Juni 2021: Amazon GameLift Realtime Client SDK- und Server-SDK-Updates

Aktualisierte SDK-Versionen: Realtime Client SDK 1.2.0, Server SDK 3.4.0 für Unreal

Mit diesem neuesten SDK-Update können Sie jetzt IL2CPP in Ihre mobilen Anwendungen integrieren, die das RTS Client SDK verwenden, und bewährte Methoden für Frameworks befolgen. Sie können jetzt auch das Amazon GameLift Server SDK für Unreal Version 4.26 erstellen. Dieses Update enthält Komponenten, die sich in Ihren Windows- oder Linux-Gameserver integrieren lassen, darunter C++- und C#-Versionen des Amazon GameLift Server SDK, Amazon GameLift Local und ein Unreal Engine-Plugin.

Höhepunkte:

- Unterstützung für IL2CPP im RTS Client SDK und für die Erstellung der nativen Bibliotheken als Frameworks hinzugefügt, sodass Sie RTS-Clients für die neuesten Mobilgeräte erstellen können.
- Sie können DescribePlayerSessions() damit Informationen für eine Einzelspielersitzung, für alle Spielersitzungen in einer Spielsitzung oder für alle Spielersitzungen abrufen, die mit einer Einzelspieler-ID verknüpft sind.
- Sie können GetInstanceCertificate() damit den Dateispeicherort eines PEM-codierten TLS-Zertifikats abrufen, das der Flotte und ihren Instances zugeordnet ist.
- Server-SDK-Unterstützung für Unreal Version 4.26 wurde erstellt.
- Das vorhandene C#-SDK, Version 4.0.2, wurde als kompatibel mit Unity 2020.3 verifiziert. Es waren keine SDK-Updates erforderlich.

Weitere Informationen:

- GameLift Amazon-Entwicklerhandbuch:
  - DescribePlayerSessions()
  - GetInstanceCertificate()

23. März 2021: Amazon GameLift fügt Benachrichtigungen zur Platzierung von Spielsitzungen hinzu

Aktualisierte SDK-Versionen: AWS SDK 1.8.168

Du kannst jetzt Ereignisse verwenden, um die Platzierungsaktivitäten einer Spielsitzung in einer Warteschlange zu überwachen. Erstellen Sie ein Amazon Simple Notification Service (Amazon SNS) -Thema, um Ereignisbenachrichtigungen zu veröffentlichen, oder richten Sie die Ereignisverfolgung mithilfe von CloudWatch Events ein.

Höhepunkte:

- Für jede Warteschlange können Sie eine benutzerdefinierte Textzeichenfolge festlegen, die in allen Ereignisnachrichten enthalten sein soll.
- Wenn Sie ein Amazon SNS SNS-Thema verwenden, können Sie zusätzliche Zugriffsbedingungen festlegen, die die Veröffentlichung auf bestimmte Warteschlangen beschränken.

Weitere Informationen:

- GameLift Amazon-Entwicklerhandbuch:
  - Richten Sie eine Eventbenachrichtigung für die Platzierung von Spielsitzungen ein (neu)
  - Platzierungsveranstaltungen für Spielsitzungen (neu)
- API-Referenz (AWS SDK)
  - Neue Warteschlangenparameter für Spielsitzungen `NotificationTarget` und`CustomEventData`: GameSessionQueue, CreateGameSessionQueue, UpdateGameSessionQueue
- GameLiftAmazon-Forum

16. März 2021: Amazon GameLift fügt Flotten mit mehreren Regionen hinzu, sechs neue Regionen

Aktualisierte SDK-Versionen: SDK 1.8.163 AWS

Amazon GameLift Managed Hosting ist jetzt in 21 AWS Regionen verfügbar. Die neuen Regionen sind Kapstadt (`af-south-1`), Bahrain (`me-south-1`), Hongkong (`ap-east-1`), Mailand (`eu-south-1`), Paris (`eu-west-3`) und Stockholm (`eu-north-1`).

Mit der neuen Amazon-Funktion für Flotten mit GameLift mehreren Standorten können Sie jetzt eine einzige Flotte einrichten, um Ihre Spieleserver in einer oder allen der 20 von Amazon GameLift unterstützten Regionen (Region Peking ausgenommen) zu hosten. Diese Funktion zielt darauf ab, den Aufwand für die Einrichtung und Wartung von GameLift Amazon-Hosting-Ressourcen weltweit erheblich zu reduzieren. Flotten mit mehreren Standorten können in den folgenden AWS Regionen eingerichtet werden: `us-east-1` (Nord-Virginia), `us-west-2` (Oregon), `eu-central-1` (Frankfurt), `eu-west-1` (Irland), `ap-southeast-2` (Sydney), `ap-northeast-1` (Tokio) und `ap-northeast-2` (Seoul). In allen anderen Regionen können Sie nach Bedarf weiterhin Flotten mit einem Standort einrichten. Alle Flotten, die vor dieser Version erstellt wurden, sind Flotten mit nur einem Standort. Die Verwendung von Flotten mit mehreren Standorten hat keinen Einfluss auf Ihre Hosting-Kosten. Die GameLift Preise von Amazon basieren auf der Art, dem Standort und dem Volumen der Instances, die Sie verwenden. (Weitere Informationen finden Sie unter GameLiftAmazon-Preise.) AWS CloudFormation Unterstützung für Flotten mit mehreren Standorten wird in Kürze verfügbar sein.

> **Note**
>
> Flotten mit mehreren Standorten sind in den Regionen Chinas nicht verfügbar. GameLiftAmazon-Ressourcen, die sich in chinesischen Regionen befinden, können nicht mit Ressourcen in anderen GameLift Amazon-Regionen interagieren oder von diesen genutzt werden.

Höhepunkte:

- Fügen Sie bei einer Flotte mit mehreren Standorten explizit eine Liste mit abgelegenen Standorten hinzu. Amazon GameLift stellt Instances desselben Typs und derselben Konfiguration, einschließlich der Build- und Runtime-Konfiguration, in der Heimatregion der Flotte und an allen hinzugefügten Standorten bereit.
- Passen Sie die Kapazitätseinstellungen und die Skalierung für jeden Standort unabhängig voneinander an. Die Richtlinien für die automatische Skalierung gelten für eine gesamte Flotte, aber Sie können sie je nach Standort ein- oder ausschalten.

- Starte neue Spielsitzungen an bestimmten Flottenstandorten. Wenn du Warteschlangen für Spielsitzungen oder Spielersuche verwendest, um Spielsitzungen zu platzieren, kannst du jetzt nach Standort, Hosting-Kosten und Spielerlatenz priorisieren, wo neue Spielsitzungen beginnen.
- Rufen Sie Hosting-Metriken in der GameLift Amazon-Konsole ab, aggregiert für alle Standorte in einer Flotte oder aufgeschlüsselt nach jedem Flottenstandort.

Weitere Informationen:

- Amazon-Blog für Spieletechnologie
- API-Referenz (AWS SDK)
  - Betrieb neuer Flottenstandorte: CreateFleetLocationsDescribeFleetLocationAttributes, DescribeFleetLocationCapacity, DescribeFleetLocationUtilization, DeleteFleetLocations
  - Aktualisierter Flottenbetrieb mit neuer Unterstützung für mehrere Standorte: CreateFleet,, DescribeEC2 UpdateFleetCapacityInstanceLimits,,, DescribeInstancesStopFleetActionsStartFleetActions
  - Die Platzierung von Spielsitzungen wurde aktualisiert, mit neuer Priorität und Filterfunktion:,, CreateGameSessionQueueDescribeGameSessionQueuesUpdateGameSessionQueue
  - Die Operationen zur Erstellung von Spielsitzungen wurden aktualisiert, mit neuer Unterstützung für den Standort: CreateGameSession, DescribeGameSessions, DescribeGameSessionDetails, SearchGameSessions
- GameLift Amazon-Entwicklerhandbuch:
  - Amazon GameLift -Hosting-Standorte(aktualisiert)
  - Leitfaden zur GameLift Amazon-Flottenplanung (neu)

    Skalierung der GameLift Amazon-Hosting-Kapazität(aktualisiert)
  - Entwerfen Sie eine Warteschlange für Spielsitzungen (neu)
  - Flottendetails anzeigen(aktualisiert)
- GameLiftAmazon-Forum

9. Februar 2021: Amazon GameLift erweitert die Unterstützung für eigenständige AMD-Instances FlexMatch

Aktualisierte SDK-Versionen: AWS SDK 1.8.139

Diese Version enthält die folgenden Updates:

- Amazon GameLift FleetIQ Gameservergruppen können jetzt für die Verwaltung der AMD-Instance-Familien C5a, M5a und R5a konfiguriert werden. Die unterstützten Amazon EC2 EC2-Instance-Typen, wie sie für aufgeführt sind GameServerGroup InstanceDefinition, umfassen jetzt Folgendes:
  - c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
  - m5a.groß, m5a.xgroß, m5a.2xgroß, m5a.4xgroß, m5a.8xgroß, m5a.12xgroß, m5a.16xgroß, m5a.24xgroß
  - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

  Hinweis: AMD-Instances für FleetIQ sind derzeit nicht für die Verwendung in der Region China (Peking) AWS verfügbar. Weitere Informationen finden Sie unter Verfügbarkeit und Implementierungsunterschiede in China.
- Das von Amazon GameLift verwaltete Spielehosting unterstützt jetzt AMD-Instances in der Region China (Peking), die von Sinnet betrieben werden. Zu den neuen AMD-Instance-Familien gehören M5a und R5a. Zu den unterstützten EC2-Instance-Typen, wie sie für Fleet aufgeführt sind InstanceType, gehören jetzt die folgenden:
  - m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge, m5a.24xlarge
  - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge
- Amazon GameLift FlexMatch kann jetzt als eigenständige Matchmaking-Lösung in der Region China (Peking) verwendet werden, die von Sinnet betrieben wird. Kunden können in der Region Peking einen FlexMatch Matchmaker erstellen und den FlexMatchModeParameter auf STANDALONE konfigurieren. Weitere Informationen dazu FlexMatch, entweder mit Amazon GameLift Managed Hosting oder mit einer Hosting-Lösung, die nicht von Amazon GameLift stammt, finden Sie im Amazon GameLift FlexMatch Developer Guide.
- Bei der Einrichtung von Ereignisbenachrichtigungen für Amazon GameLift FlexMatch können Sie jetzt ein Amazon SNS FIFO-Thema als Benachrichtigungsziel festlegen. Weitere Informationen finden Sie hier:
  - MatchmakingConfiguration NotificationTarget, Amazon GameLift API-Referenz
  - FlexMatch Veranstaltungsbenachrichtigung einrichten, Amazon GameLift FlexMatch Developer Guide
  - Wir stellen vor: Amazon SNS FIFO — F irst-in-first-out Pub/Sub-Messaging, Nachrichtenblog AWS

22. Dezember 2020: Das Amazon GameLift Server SDK unterstützt Unreal Engine 4.25 und Unity 2020

Aktualisierte SDK-Versionen: Amazon GameLift Server SDK 4.0.2, Unreal-Plug-in-Version 3.3.3

Die neueste Version des Amazon GameLift Server SDK enthält die folgenden Komponenten:

- Das aktualisierte Unreal-Plugin wurde aus Gründen der Kompatibilität mit Unreal Engine 4.25 aktualisiert. Die API wurde nicht geändert.
- Das bestehende C#-SDK, Version 4.0.2, wurde als kompatibel mit Unity 2020 verifiziert. Es waren keine SDK-Updates erforderlich.

Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift getting started herunter.

24. November 2020: Amazon ist GameLift FlexMatch jetzt für Spiele verfügbar, die überall gehostet werden

Aktualisierte SDK-Versionen: AWS SDK 1.8.95

Amazon GameLift FlexMatch ist ein anpassbarer Matchmaking-Service für Multiplayer-Spiele. Ursprünglich für Nutzer von Amazon GameLift Managed Hosting konzipiert, FlexMatch kann es jetzt in Spiele integriert werden, die andere Hosting-Systeme verwenden peer-to-peer, darunter proprietäres lokales Computing und primitive Cloud-Computing-Typen. Spiele, die Amazon GameLift FleetIQ für das Hosten von Spielen auf Amazon EC2 verwenden, können jetzt Matchmaking mit implementieren. FlexMatch

FlexMatch bietet einen robusten Matchmaking-Algorithmus und eine Regelsprache, sodass Sie den Matchmaking-Prozess so anpassen können, dass Spieler auf der Grundlage der wichtigsten Spielermerkmale und der gemeldeten Latenz zusammengebracht werden. Darüber hinaus FlexMatch bietet es einen Workflow für Matchmaking-Anfragen, der Funktionen wie Spielerpartys, Spielerakzeptanz und Match-Backfill unterstützt. Wenn Sie Amazon GameLift Managed Hosting oder Realtime Server verwenden FlexMatch , verwendet der Matchmaker automatisch Amazon, um Hosting-Ressourcen GameLift zu finden und eine neue Spielsitzung für neu gegründete Spiele zu starten. Bei der Nutzung FlexMatch als eigenständiger Dienst sendet der Matchmaker die Spielergebnisse zurück an Ihr Spiel, das dann mit Ihrer Hosting-Lösung eine neue Spielsitzung starten kann.

API-Operationen für FlexMatch sind Teil der Amazon GameLift Service API, die im AWS SDK und der AWS Command Line Interface (AWS CLI) enthalten ist. Diese Version enthält die folgenden Updates zur Unterstützung von eigenständigem Matchmaking:

- Die API-Ressource `MatchmakingConfiguration` hat die folgenden Änderungen:
  - Neue Eigenschaft, `FlexMatchMode` gibt an, ob der Matchmaker mit Amazon GameLift Managed Hosting oder als eigenständiges Matchmaking verwendet wird.
  - Die Eigenschaft `GameSessionQueueArns` ist nicht erforderlich, wenn sie auf `FlexMatchMode` Standalone gesetzt ist.
  - Diese Eigenschaften werden beim eigenständigen Matchmaking nicht verwendet:`AdditionalPlayerCount`,, `BackfillModeGameProperties`,`GameSessionData`.
- Die automatische Backfill-Funktion ist beim eigenständigen Matchmaking nicht verfügbar.

24. November 2020: AMD-Instances jetzt bei Amazon verfügbar GameLift

Aktualisierte SDK-Versionen: AWS SDK 1.8.95

Die Liste der von Amazon EC2 EC2-Instance-Typen unterstützten Amazon EC2-Instance-Typen umfasst GameLift jetzt drei neue Instance-Familien: C5a, M5a und R5a. Diese Familien bestehen aus rechenoptimierten AMD-Instances, die von AMD EPYC-Prozessoren angetrieben werden, die mit Frequenzen von bis zu 3,3 laufen. GHz. Die AMD-Instances sind x86-kompatibel. Spiele, die derzeit auf Amazon laufen, GameLift können unverändert für AMD-Instance-Typen bereitgestellt werden. Die neuen Instances sind in den folgenden AWS Regionen verfügbar: USA Ost (Nord-Virginia und Ohio), USA West (Oregon und Nordkalifornien), Zentral-Kanada (Montreal), Südamerika (Sao Paulo), EU-Mitte (Frankfurt), EU West (London und Irland), Asien-Pazifik Süd (Mumbai), Asien-Pazifik-Nordosten (Seoul und Tokio) und Asien-Pazifik Südosten (Singapur und Sydney).

Zu den neuen AMD-Instances gehören:

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
- m5a.groß, m5a.xgroß, m5a.2xgroß, m5a.4xgroß, m5a.8xgroß, m5a.12xgroß, m5a.16xgroß, m5a.24xgroß
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

Weitere Informationen:

- Amazon-Blog für Spieletechnologie
- Preise Amazon GameLift Amazon-Instances
- Amazon EC2 EC2-Instances mit AMD EPYC-Prozessoren
- GameLiftAmazon-Forum

11. November 2020: Versionsupdate für das Amazon GameLift Server SDK

Aktualisierte SDK-Versionen: Amazon GameLift Server SDK 4.0.2

Die neue Server-SDK-Version 4.0.2 behebt ein bekanntes Problem mit dem API-Betrieb. `StartMatchBackfill()` Dieser Vorgang gibt jetzt eine korrekte Antwort auf eine Match-Backfill-Anfrage zurück.

Das Problem hatte keine Auswirkungen auf den Match-Backfill-Prozess, und die Funktionsweise dieser Funktion hat sich nicht geändert. Das Problem hat sich möglicherweise auf die Protokollnachrichten und die Fehlerbehandlung bei Match-Backfill-Anfragen ausgewirkt.

Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift getting started herunter.

5. November 2020: Neue FlexMatch Algorithmus-Anpassungen

FlexMatch Benutzer können jetzt die folgenden Standardverhaltensweisen für den Matchmaking-Prozess anpassen. Diese Anpassungen sind in einem Matchmaking-Regelsatz festgelegt. Es gibt keine Änderungen an den Amazon GameLift SDKs.

- Backfill-Tickets priorisieren: Sie können wählen, ob Sie bei der Suche nach akzeptablen Matches die Priorität von Backfill-Tickets erhöhen oder verringern möchten. Die Priorisierung von Backfill-Tickets ist nützlich, wenn die automatische Backfill-Funktion aktiviert ist. Verwenden Sie die Algorithmus-Eigenschaft. `backfillPriority`
- Vorsortierung zur Optimierung der Konsistenz und Effizienz von Spielen: Konfigurieren Sie Ihren Matchmaker so, dass der Ticketpool vorab sortiert wird, bevor Tickets zur Auswertung gebündelt werden. Indem du Tickets anhand der wichtigsten Spielerattribute vorsortierst, haben deine resultierenden Spiele in der Regel Spieler, die sich in diesen Attributen ähnlicher sind. Sie können den Bewertungsprozess auch effizienter gestalten, indem Sie die Sortierung nach denselben Attributen vornehmen, die auch in den Spielregeln verwendet werden. Verwenden Sie

die Algorithmuseigenschaft`sortByAttributes`, wenn die `strategy` Eigenschaft auf „sortiert“ gesetzt ist.

- Passen Sie an, wie die Wartezeiten für Erweiterungen ausgelöst werden: Wählen Sie, ob Sie Erweiterungen auf der Grundlage des Alters des neuesten (Standard) oder des ältesten Tickets in einem unvollständigen Spiel auslösen möchten. Das Auslösen mit dem ältesten Ticket führt dazu, dass Spiele schneller abgeschlossen werden, während das Auslösen mit dem neuesten Ticket zu einer höheren Spielqualität führt. Verwenden Sie die Algorithmus-Eigenschaft. `expansionAgeSelection`

17. September 2020: Amazon GameLift aktualisiert das Server-SDK

Aktualisierte SDK-Versionen: Amazon GameLift Server SDK 4.0.1

Das neue Server-SDK enthält die folgenden Updates:

- C#-API-Version 4.0.1
  - Der API-Vorgang TerminateGameSession() wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, ProcessEnding() um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
  - Ein bekanntes Problem mit dem GetInstanceCertificate() Vorgang wurde behoben.
  - Die Operation gibt GetTerminationTime() jetzt einen Wert vom Datentyp zurück AwsDateTimeOutcome.
- C++-API-Version 3.4.1
  - Der Vorgang TerminateGameSession() wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, ProcessEnding() um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
- Version 3.3.2 des Unreal Engine-Plug-ins
  - Der Vorgang TerminateGameSession() wird nicht mehr unterstützt. Ersetze ihn durch einen Aufruf, ProcessEnding() um sowohl eine Spielsitzung als auch den Serverprozess zu beenden.
  - Der Callback-Vorgang `OnUpdateGameSession` wurde hinzugefügt, um Match Backfill F ProcessParameters zu unterstützen.

Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift getting started herunter.

27. August 2020: Amazon GameLift FleetIQ für Spiele-Hosting mit Amazon EC2 (allgemeine Verfügbarkeit)

Aktualisierte SDK-Versionen: SDK 1.8.36 AWS

Die Amazon GameLift FleetIQ Lösung für kostengünstiges, cloudbasiertes Spielehosting auf Amazon EC2 ist jetzt allgemein verfügbar. Amazon GameLift FleetIQ bietet Entwicklern die Möglichkeit, Spieleserver direkt auf Amazon EC2-Spot-Instances zu hosten, indem ihre Rentabilität für das Hosting von Spielen optimiert wird. Spieleentwickler können Amazon GameLift FleetIQ für neue Spiele oder zur Erweiterung der Kapazität vorhandener Spiele verwenden. Diese Lösung unterstützt die Verwendung von Containern oder anderen AWS Diensten wie AWS Shield und Amazon Elastic Container Service (Amazon ECS).

Diese Version für allgemeine Verfügbarkeit umfasst die folgenden Updates für die Amazon GameLift FleetIQ-Lösung:

- Ein neuer API-Vorgang `DescribeGameServerInstances` gibt Informationen, einschließlich des Status, zu allen aktiven Instances für eine Amazon GameLift FleetIQ-Spielservergruppe zurück.
- Neue Balancing-Strategie`ON_DEMAND_ONLY`, konfiguriert eine Spieleservergruppe so, dass sie nur On-Demand-Instances verwendet. Sie können die Balancing-Strategie einer Spieleservergruppe jederzeit aktualisieren, sodass Sie je nach Bedarf zwischen Spot-Instances und On-Demand-Instances wechseln können.
- Die folgenden Vorschauelemente wurden aus Gründen der allgemeinen Verfügbarkeit entfernt:
  - Verwendung von benutzerdefinierten Sortierschlüsseln für Spielserverressourcen. Spieleserver können anhand des Zeitstempels der Registrierung sortiert werden.
  - Tagging für Spielserver-Ressourcen.

16. April 2020: Amazon GameLift aktualisiert das Server-SDK SDK for Unity und Unreal Engine

Aktualisierte SDK-Versionen: Amazon GameLift Server SDK 4.0.0, Amazon GameLift Local 1.0.5

Die neueste Version des Amazon GameLift Server SDK enthält die folgenden aktualisierten Komponenten:

- C# SDK Version 4.0.0 wurde für Unity 2019 aktualisiert.

- Die Unreal-Plug-in-Version 3.3.1 wurde für die Unreal Engine-Versionen 4.22, 4.23 und 4.24 aktualisiert.
- Amazon GameLift Local Version 1.0.5 wurde aktualisiert, um Integrationen zu testen, die das C#-Server-SDK Version 4.0.0 verwenden.

Laden Sie die neueste Version des Amazon GameLift Server SDK bei Amazon GameLift getting started herunter.

2. April 2020: Amazon GameLift FleetIQ ist für das Hosten von Spielen auf EC2 verfügbar (öffentliche Vorschau)

Aktualisierte SDK-Versionen: SDK 1.7.310 AWS

Die Amazon GameLift FleetIQ-Funktion optimiert die Rentabilität kostengünstiger Spot-Instances für die Verwendung mit Game-Hosting. Diese Funktion wurde jetzt für Kunden erweitert, die ihre Hosting-Ressourcen direkt und nicht über den verwalteten GameLift Amazon-Service verwalten möchten. Diese Lösung unterstützt die Verwendung von Containern oder anderen AWS Diensten wie AWS Shield und Amazon Elastic Container Service (Amazon ECS).

Weitere Informationen:

GameTech Blogbeitrag auf Amazon GameLift FleetIQ

19. Dezember 2019: Verbessertes AWS Ressourcenmanagement für GameLift Amazon-Ressourcen

Aktualisierte SDK-Versionen: AWS SDK 1.7.249

Sie können jetzt AWS Ressourcenmanagement-Tools mit GameLift Amazon-Ressourcen nutzen. Insbesondere allen wichtigen GameLift Amazon-Ressourcen — Builds, Skripten, Flotten, Warteschlangen für Spielsitzungen, Matchmaking-Konfigurationen und Matchmaking-Regelsätze — werden jetzt Werte für den Amazon-Ressourcennamen (ARN) zugewiesen. Ein Ressourcen-ARN bietet eine konsistente Kennung, die in allen AWS Regionen eindeutig ist. Sie können verwendet werden, um ressourcenspezifische AWS Identity and Access Management (IAM) Berechtigungsrichtlinien zu erstellen. Ressourcen wird jetzt ein ARN und auch die bereits vorhandene Ressourcen-ID zugewiesen, die nicht regionsspezifisch ist.

Darüber hinaus unterstützen GameLift Amazon-Ressourcen jetzt Tagging. Sie können Tags verwenden, um Ressourcen zu organisieren, IAM-Berechtigungsrichtlinien zu erstellen, um den

Zugriff auf Ressourcengruppen zu verwalten, AWS Kostenaufschlüsselungen anzupassen usw. Verwenden Sie bei der Verwaltung von Tags für GameLift Amazon-Ressourcen die GameLift Amazon-API-Aktionen `TagResource()``UntagResource()`, und`ListTagsForResource()`.

Weitere Informationen:

- TagResourcein der Amazon GameLift API-Referenz
- Tagging von AWS Ressourcen in der AWS Allgemeinen Referenz
- Amazon-Ressourcennamen in der AWS allgemeinen Referenz

14. November 2019: Neue AWS CloudFormation Vorlagen, Aktualisierungen in der Region China (Peking)

Aktualisierte SDK-Versionen: AWS SDK 1.7.210

AWS CloudFormation Vorlagen für Amazon GameLift

GameLift Amazon-Ressourcen können jetzt über erstellt und verwaltet werden AWS CloudFormation. Die vorhandenen AWS CloudFormation Build- und Flottenvorlagen wurden aktualisiert, um sie an die aktuellen Ressourcen anzupassen. Neue Vorlagen sind jetzt für Skripte, Warteschlangen, Matchmaking-Konfigurationen und Matchmaking-Regelsätze verfügbar. AWS CloudFormation Vorlagen vereinfachen die Verwaltung von Gruppen verwandter AWS Ressourcen erheblich, insbesondere bei der Bereitstellung von Spielen in mehreren Regionen.

Weitere Informationen:

- Referenz zum GameLift Amazon-Ressourcentyp im AWS CloudFormation Benutzerhandbuch
- Ressourcen verwalten mit AWS CloudFormationim Amazon GameLift Developer Guide

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.