aws

개발자 가이드

아마존 GameLift

아마존 GameLift: 개발자 가이드

Table of Contents

Amazon GameLift란 무엇인가요?

Amazon GameLift를 사용하면 세션 기반 멀티플레이어 게임을 위해 클라우드에서 저비용 전용 서버를 배포, 운영 및 확장할 수 있습니다. AWS 글로벌 컴퓨팅 인프라를 기반으로 구축된 Amazon GameLift는 고성능, 높은 신뢰성을 갖춘 게임 서버를 제공하는 동시에 리소스 사용량을 동적으로 조정하여 전 세계 플레이어 수요를 충족할 수 있습니다.

Amazon GameLift 사용

Amazon GameLift는 다음과 같은 사용 사례 및 기타 사례를 지원합니다.

- 자체 사용자 지정 멀티플레이어 게임 서버를 사용하거나 바로 사용할 수 있는 Realtime 서버를 사용하여 게임을 호스팅합니다.
- Amazon Elastic Compute Cloud(Amazon EC2) 스팟 인스턴스를 사용하여 저렴한 호스팅 리소스를 실행할 수 있습니다.
- 사용량에 따라 게임에 필요한 호스팅 리소스의 양을 자동으로 조정합니다.
- Amazon GameLift FleetIQ를 사용하여 Amazon EC2 컴퓨팅 리소스를 모두 한 곳에서 관리할 수 있습니다.
- Amazon GameLift FlexMatch를 사용하여 멀티플레이어 게임에서 플레이어를 매칭합니다.
- Amazon GameLift Anywhere를 사용하여 게임 서버 및 클라이언트 빌드를 반복적으로 테스트합니다.
- Amazon GameLift Anywhere를 사용하면 자체 하드웨어를 사용하면서 모든 것을 한 곳에서 관리할 수 있습니다.

> **Tip**
>
> Amazon GameLift 게임 서버 호스팅을 사용해 보려면 Amazon GameLift 시작하기 섹션을 참조하세요.

Amazon GameLift 솔루션 시작하기

게임 개발자용 Amazon GameLift 솔루션

- 사용자 지정 서버용 Amazon GameLift 호스팅

- Realtime 서버를 사용한 Amazon GameLift 호스팅
- Amazon EC2에서 호스트하기 위한 Amazon GameLift FleetIQ
- 매치메이킹을 위한 Amazon GameLift FlexMatch
- Amazon GameLift Anywhere 하드웨어 호스팅

사용자 지정 서버용 Amazon GameLift 호스팅

Amazon GameLift는 자체 사용자 지정 게임 서버를 호스팅하는 데 필요한 작업을 대체합니다. Auto Scaling 기능을 사용하면 필요한 것보다 더 많은 리소스에 비용을 지불하지 않아도 됩니다. 또한 Auto Scaling을 사용하면 대기 시간을 최소화하면서 신규 플레이어가 게임에 항상 참여할 수 있도록 할 수 있습니다.

Amazon GameLift 호스팅에 대한 자세한 내용은 Amazon GameLift 작동 방식 섹션을 참조하세요.

주요 기능

- Auto Scaling, 다중 위치 대기열, 게임 세션 배치 등의 Amazon GameLift 관리 기능을 사용합니다.
- Amazon Linux 또는 Windows Server 운영 체제에서 실행할 게임 서버를 배포합니다.
- 게임 세션 및 플레이어 세션을 관리합니다.
- 서버 프로세스에 사용자 지정 상태 추적을 설정하여 문제를 검색하고 성능이 저하된 프로세스의 문제를 해결합니다.
- Amazon GameLift에 대한 AWS CloudFormation 템플릿을 사용하여 게임 리소스를 관리합니다.

Realtime 서버를 사용한 Amazon GameLift 호스팅

Realtime 서버를 사용하여 사용자 지정으로 구축된 게임 서버가 필요 없는 게임을 지원합니다. 이 경량 서버 솔루션은 게임에 맞게 구성할 수 있는 게임 서버를 제공합니다.

Realtime 서버를 사용한 Amazon GameLift 호스팅에 대한 자세한 내용은 Amazon GameLift Realtime 서버와 게임 통합 섹션을 참조하세요.

주요 기능

- Auto Scaling, 다중 위치 대기열, 게임 세션 배치 등의 Amazon GameLift 관리 기능을 사용합니다.
- Amazon GameLift 호스팅 리소스를 사용하여 해당 플릿에 대한 AWS 컴퓨팅 하드웨어 유형을 선택합니다.

- 게임 클라이언트 및 서버 상호 작용을 위한 전체 네트워크 스택을 활용합니다.
- 사용자 지정 가능한 서버 로직으로 핵심 게임 서버 기능을 얻습니다.
- Realtime 구성 및 서버 로직에 대한 실시간 업데이트를 수행합니다.

Amazon EC2에서 호스트하기 위한 Amazon GameLift FleetIQ

Amazon GameLift FleetIQ를 사용하면 Amazon EC2 및 Amazon EC2 Auto Scaling의 호스팅 리소스를 직접 사용할 수 있습니다. 이를 통해 저렴하고 복원력이 뛰어난 게임 호스팅을 위한 Amazon GameLift 최적화의 이점을 제공합니다. 이 솔루션은 완전 관리형 Amazon GameLift 솔루션이 제공하는 것보다 더 많은 유연성을 필요로 하는 게임 개발자를 위한 것입니다.

Amazon GameLift FleetIQ가 게임 호스팅을 위해 Amazon EC2 및 EC2 Auto Scaling과 함께 작동하는 방식에 대한 자세한 내용은 Amazon GameLift FleetIQ 개발자 가이드를 참조하세요.

주요 기능

- FleetIQ 알고리즘을 사용하여 스팟 인스턴스 밸런싱을 최적화합니다.
- 플레이어 라우팅 기능을 사용하여 게임 서버 리소스를 효율적으로 관리하고 게임 참여에 대한 플레이어 경험을 개선합니다.
- 플레이어 사용량에 따라 자동으로 호스팅 용량을 조정합니다.
- 자체 AWS 계정 계정에서 Amazon EC2 인스턴스를 직접 관리합니다.
- Windows, Linux, 컨테이너 및 Kubernetes 등 지원되는 게임 서버 실행 파일 형식 중 하나를 사용합니다.

매치메이킹을 위한 Amazon GameLift FlexMatch

FlexMatch를 사용하여 게임의 멀티플레이어 매치를 정의하기 위해 사용자 지정 규칙 세트를 구축합니다. FlexMatch는 규칙 세트를 사용하여 각 매치의 호환 가능한 플레이어를 비교하고 플레이어에게 이상적인 멀티플레이어 경험을 제공합니다.

FlexMatch에 대한 자세한 내용은 Amazon GameLift FlexMatch란 무엇인가요?를 참조하세요.

주요 기능

- 매치 생성 속도와 매치 품질의 균형을 맞춥니다.
- 정의된 특성에 따라 플레이어 또는 팀을 매칭합니다.

- 지연 시간을 기준으로 플레이어를 매치에 배치하는 규칙을 정의합니다.

Amazon GameLift Anywhere 하드웨어 호스팅

Amazon GameLift Anywhere를 사용하여 사용자 환경 내 어디에서나 하드웨어를 Amazon GameLift 게임 호스팅에 통합합니다. 매치메이커와 게임 세션 대기열에서 Anywhere 플릿 및 EC2 플릿을 통합하여 하드웨어 전반에 걸쳐 매치메이킹과 게임 배치를 관리할 수 있습니다.

Anywhere과 함께 테스트하는 방법에 대한 자세한 내용은 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 섹션을 참조하세요. Anywhere 플릿 설정에 대한 자세한 내용은 Amazon GameLift 플릿 설정 섹션을 참조하세요.

주요 기능

- 게임 서버 및 클라이언트 빌드를 빠르고 반복적으로 테스트할 수 있습니다.
- Amazon GameLift 도구 세트를 사용하여 자체 하드웨어에 게임을 배포할 수 있습니다.
- 플레이어와 가장 가까운 곳, 어디에서나 하드웨어를 사용할 수 있습니다.

Amazon GameLift 액세스

이러한 도구를 사용하여 Amazon GameLift를 사용할 수 있습니다.

Amazon GameLift SDK

Amazon GameLift SDK에는 게임 클라이언트, 게임 서버, 게임 서비스에서 Amazon GameLift와 통신하는 데 필요한 라이브러리가 포함되어 있습니다. 자세한 내용은 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.

Amazon GameLift Realtime Client SDK

Realtime Client SDK를 사용하면 게임 클라이언트가 Realtime 서버에 연결하여 게임 세션에 참여하고 다른 플레이어와 동기화된 상태를 유지할 수 있습니다. SDK를 다운로드하고 Realtime Servers Client API(C#)를 사용하여 API를 호출하는 방법에 대해 자세히 알아보세요.

Amazon GameLift 콘솔

Amazon GameLift의 AWS Management Console을 사용하여 게임 배포를 관리하고 리소스를 구성하고 플레이어 사용량 및 성능 지표를 추적합니다. Amazon GameLift 콘솔은 AWS Command Line Interface(AWS CLI)를 통한 프로그래밍 방식의 리소스 관리를 대신할 GUI를 제공합니다.

AWS CLI

Amazon GameLift API를 포함하여 AWS SDK에 호출하려면 이 명령줄 도구를 사용합니다. AWS CLI 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서 AWS CLI의 시작하기를 참조하세요.

Amazon GameLift 요금

Amazon GameLift는 사용 기간별로 인스턴스에 대해 요금을 부과하고, 전송된 데이터 양을 기준으로 대역폭을 청구합니다. Amazon GameLift에 관련된 전체적인 요금 및 가격 목록은 Amazon GameLift 요금을 참조하세요.

Amazon GameLift를 통한 게임 호스팅 또는 매치메이킹 비용 계산에 대한 자세한 내용은 AWS Pricing Calculator 사용 방법을 설명하는 Amazon GameLift의 추정 요금 산출 섹션을 참조하세요.

Amazon GameLift 작동 방식

게임 호스팅에 대한 핵심 구성 요소를 다루며 Amazon GameLift를 통해 플레이어가 멀티플레이어 게임 서버를 사용하게 되는 방식을 설명합니다.

Amazon GameLift에서 게임을 호스팅할 준비가 되었나요? Amazon GameLift 관리형 호스팅 로드맵 섹션을 확인합니다.

핵심 구성 요소

Amazon GameLift가 게임을 호스팅하도록 설정하려면 다음 구성 요소가 필요합니다. 관리형 Amazon을 사용한 게임 아키텍처 GameLift의 다이어그램은 이러한 구성 요소 간의 관계를 시각화합니다.

- 게임 서버는 플릿에서 실행되는 게임의 서버 소프트웨어입니다. 게임 서버 빌드 또는 스크립트를 Amazon GameLift에 업로드하고 Amazon GameLift에 알립니다. Amazon GameLift Anywhere 또는 Amazon GameLift FleetIQ를 사용하는 경우 게임 서버 빌드를 컴퓨팅 리소스에 직접 업로드합니다.
- 게임 세션은 플레이어가 참여하는 진행 중인 게임입니다. 수명 및 플레이어 수와 같은 게임 세션의 기본 특성을 정의합니다. 그런 다음 플레이어는 게임 서버에 연결하여 게임 세션에 참여합니다.
- 게임 클라이언트는 플레이어의 디바이스에서 실행되는 게임 소프트웨어입니다. 게임 클라이언트는 Amazon GameLift에서 수신한 연결 정보를 기반으로 백엔드 서비스를 통해 게임 서버에 연결하여 게임 세션에 참여합니다.

- 백엔드 서비스는 Amazon GameLift와 관련된 작업을 처리하는 추가 사용자 지정 서비스입니다. 가장 좋은 방법은 백엔드 서비스가 Amazon GameLift와의 모든 게임 클라이언트 통신을 처리하는 것입니다.

게임 서버 호스팅

Amazon GameLift를 사용하면 관리형 Amazon GameLift, Amazon GameLift FleetIQ 및 Amazon GameLift Anywhere의 세 가지 방법으로 게임 서버를 호스팅할 수 있습니다. Amazon GameLift FleetIQ에 대한 자세한 내용은 Amazon GameLift FleetIQ란 무엇인가요?를 참조하세요.

게임의 요구 사항에 맞게 플릿을 설계할 수 있습니다. 플릿 설계에 대한 자세한 내용은 Amazon GameLift 플릿 설계 가이드 섹션을 참조하세요.

관리형 Amazon GameLift

관리형 Amazon GameLift를 사용하면 인스턴스라고 하는 Amazon GameLift 가상 컴퓨팅 리소스에서 게임 서버를 호스팅할 수 있습니다. 인스턴스 플릿을 생성하고 게임 서버를 실행하도록 배포하여 호스팅 리소스를 설정합니다.

Amazon GameLift Anywhere

Amazon GameLift Anywhere를 사용하면 관리하는 컴퓨팅에서 게임 서버를 호스팅할 수 있습니다. 컴퓨팅을 참조하는 Anywhere 플릿을 생성하여 호스팅 리소스를 설정합니다.

플릿 별칭

별칭은 플릿 간에 전송할 수 있는 명칭으로, 이를 사용하면 플릿 위치를 일반화하는 편리한 방법입니다. 별칭을 사용하여 게임 클라이언트를 변경하지 않고도 게임 클라이언트에서 사용하는 플릿을 전환할 수 있습니다. 콘텐츠를 가리키는 터미널 별칭을 만들 수도 있습니다.

게임 세션 실행

플릿에 게임 서버 빌드를 배포하고 Amazon GameLift가 각 인스턴스에서 게임 서버 프로세스를 시작하면 플릿에서 게임 세션을 호스팅할 수 있습니다. Amazon GameLift는 게임 클라이언트 서비스가 백엔드 서비스 또는 Amazon GameLift에 배치 요청을 보낼 때 새 게임 세션을 시작합니다.

게임 세션 배치 및 FleetIQ 알고리즘

대기열은 FleetIQ 알고리즘을 사용하여 새 게임 세션을 호스팅할 수 있는 사용 가능한 게임 서버를 선택합니다. 게임 세션 배치의 주요 구성 요소는 Amazon GameLift 게임 세션 대기열입니다. 게임 세션

대기열에 플릿 목록을 할당하여 대기열에서 게임 세션을 배치할 수 있는 위치를 결정합니다. 게임 세션 대기열 및 게임에 맞게 대기열을 설계하는 방법에 대한 자세한 내용은 게임 세션 대기열 설계 섹션을 참조하세요.

게임에 대한 플레이어 연결

게임 세션 배치 프로세스의 일부로써 대기열 또는 게임 세션은 선택된 게임 서버에 새로운 게임 세션을 시작하라는 메시지를 표시합니다. 게임 서버는 메시지에 응답하고 플레이어 연결을 수락할 준비가 되면 Amazon GameLift에 다시 보고합니다. 그러면 Amazon GameLift는 백엔드 서비스 또는 게임 클라이언트 서비스에 연결 정보를 전송합니다. 그런 다음 게임 클라이언트는 이 정보를 사용하여 게임 세션에 직접 연결하고 게임플레이를 시작합니다.

플릿 용량 조정

플릿이 활성화되고 게임 세션을 호스팅할 준비가 되면 플레이어 요구에 맞게 플릿 용량을 조정할 수 있습니다. 들어오는 모든 플레이어가 게임을 빨리 찾는 것과 유휴 상태인 리소스에 과도하게 소비하는 것 사이에서 균형을 찾는 것이 좋습니다.

Amazon GameLift는 매우 효과적인 자동 크기 확장 도구를 제공하거나 플릿 용량을 수동으로 설정할 수 있습니다. 자세한 내용은 Amazon GameLift 호스팅 용량 확장 섹션을 참조하세요.

Auto Scaling

Amazon GameLift는 두 가지 Auto Scaling 방법을 제공합니다.

- 대상 기반 Auto Scaling
- 규칙 기반 정책을 사용한 Auto Scaling

추가 조정 기능

- 게임 세션 보호 - 스케일 다운 이벤트 동안 활성 플레이어를 호스팅한 게임 세션에서 Amazon GameLift가 종료되는 것을 방지합니다.
- 크기 조정 제한 - 플릿의 최소 및 최대 인스턴스 수를 설정하여 전체 인스턴스 사용량을 제어합니다.
- Auto Scaling 일시 중지 - Auto Scaling 정책을 변경하거나 삭제하지 않으면서 플릿 위치 수준에서 Auto Scaling을 일시 중지합니다.
- 크기 조정 지표 - 플릿의 용량 및 조정 이벤트 기록을 추적합니다.

Amazon GameLift 모니터링

플릿을 실행 중인 경우, Amazon GameLift가 다양한 정보를 수집하여 이미 배포한 게임 서버의 성능을 모니터링하는 데 도움을 줍니다. 이 정보를 이용하여 리소스 사용을 최적화하고 문제를 해결하는 한편, 플레이어의 게임 플레이에 대한 통찰을 얻을 수 있습니다. Amazon GameLift는 다음을 수집합니다.

- 플릿, 위치, 게임 세션 및 플레이어 세션 세부 정보
- 사용량 지표
- 서버 프로세스 상태
- 게임 세션 로그

Amazon GameLift 모니터링에 대한 자세한 내용은 아마존 모니터링 GameLift 섹션을 참조하세요.

다른 AWS 리소스 사용

게임 서버 및 애플리케이션은 다른 AWS 리소스와 통신할 수 있습니다. 예를 들어 플레이어 인증 또는 소셜 네트워킹의 다양한 웹 서비스를 이용할 수 있습니다. 게임 서버를 통해 AWS 계정가 관리하는 AWS 리소스에 액세스하려면 Amazon GameLift가 AWS 리소스에 액세스할 수 있도록 명시적으로 허용합니다.

Amazon GameLift에서는 이러한 유형의 액세스를 관리할 수 있는 다양한 옵션을 제공합니다. 자세한 내용은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

플레이어의 게임 연결 방법

게임 세션은 Amazon GameLift에서 실행되는 게임의 인스턴스입니다. 게임을 플레이하려면 플레이어가 기존 게임 세션을 검색하여 참여하거나 새 게임 세션을 만들고 참여할 수 있습니다. 플레이어는 게임 세션에 대해 플레이어 세션을 만들어 참여합니다. 게임 세션이 플레이어에게 열려 있는 경우 Amazon GameLift는 플레이어를 위한 슬롯을 예약하고 연결 정보를 제공합니다. 그러면 플레이어가 게임 세션에 연결하고 예약된 슬롯을 클레임할 수 있습니다.

게임 세션 및 사용자 지정 게임 서버가 있는 플레이어 세션을 만들고 관리하는 방법에 대한 자세한 내용은 GameLift Amazon을 게임 클라이언트에 추가 섹션을 참조하세요. 플레이어를 Realtime 서버에 연결하는 방법에 대한 자세한 내용은 Realtime 서버용 게임 클라이언트 통합 섹션을 참조하세요.

Amazon GameLift는 게임 및 플레이어 세션과 관련된 여러 가지 기능을 제공합니다.

여러 위치에서 최상의 가용 리소스에서 게임 세션 호스트

Amazon GameLift가 새 게임 세션을 호스트할 리소스를 선택하는 방법을 구성할 때 여러 가지 옵션 중에서 선택합니다. 여러 위치에서 플릿을 운영하는 경우 위치에 관계없이 모든 플릿에 새 게임 세션을 배치하는 게임 세션 대기열을 설계할 수 있습니다.

게임 세션에 대한 플레이어 액세스를 제어

연결된 플레이어 수에 관계없이 새 플레이어의 참여 요청을 허용하거나 거부하도록 게임 세션을 구성합니다.

사용자 지정 게임 및 플레이어 데이터 사용

게임 세션 객체 및 플레이어 세션 객체에 사용자 지정 데이터를 추가합니다. Amazon GameLift는 새 게임 세션을 시작할 때 게임 세션 데이터를 게임 서버로 전달합니다. Amazon GameLift는 플레이어가 게임 세션에 연결할 때 플레이어 세션 데이터를 게임 서버로 전달합니다.

사용 가능한 게임 세션을 필터링 및 정렬

세션 검색 및 정렬을 사용하여 잠재 플레이어에게 가장 적합한 게임 상대를 검색하거나 플레이어가 사용 가능한 게임 세션 목록을 선택할 수 있도록 허용합니다. 세션 검색 및 정렬을 사용하여 세션 특성을 기반으로 게임 세션을 찾을 수 있습니다. 자체 사용자 지정 게임 데이터를 기반으로 검색하고 정렬할 수도 있습니다.

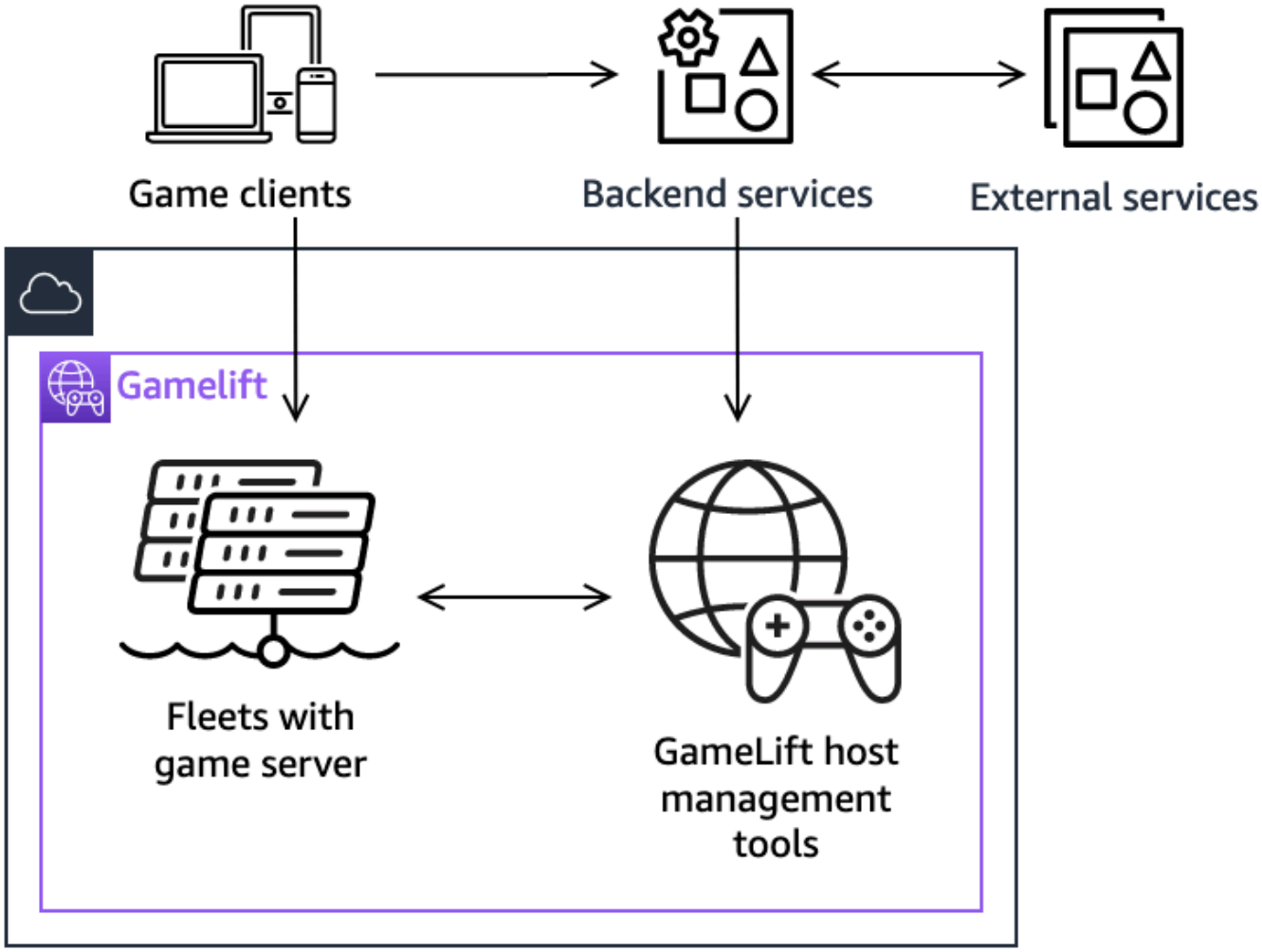
게임 및 플레이어 사용량 데이터를 추적

완료된 게임 세션의 로그를 자동으로 저장하도록 합니다. Amazon GameLift를 게임 서버에 통합할 때 로그 스토리지를 설정할 수 있습니다. 자세한 내용은 Amazon GameLift에서 서버 메시지 로깅 섹션을 참조하세요.

Amazon GameLift 콘솔을 사용하여 세션 메타데이터, 설정, 플레이어 세션 데이터 등의 게임 세션에 대한 자세한 정보를 볼 수 있습니다. 자세한 정보는 게임 및 플레이어 세션에서 데이터 보기 및 지표 섹션을 참조하세요.

관리형 Amazon을 사용한 게임 아키텍처 GameLift

다음 다이어그램은 관리형 Amazon GameLift 솔루션을 사용하여 호스팅되는 게임 아키텍처의 주요 구성 요소를 보여줍니다.



이 아키텍처의 주요 구성 요소에 포함되는 사항들은 다음과 같습니다.

게임 클라이언트

GameLiftAmazon에서 호스팅되는 게임에 참여하려면 게임 클라이언트가 먼저 사용 가능한 게임 세션을 찾아야 합니다. 게임 클라이언트는 기존 게임 세션을 검색하거나 매치메이킹을 요청하거나 백엔드 서비스를 GameLift 통해 Amazon과 통신하여 새 게임 세션을 시작합니다. 백엔드 서비스가 GameLift Amazon에 요청을 보내면 이에 대한 응답으로 서비스가 게임 세션 정보를 수신하여 게임 클라이언트에 다시 전달합니다. 그러면 게임 클라이언트가 게임 서버에 연결됩니다. 자세한 정보는 아마존에서 게임을 준비하기 GameLift을 참조하세요.

백엔드 서비스

백엔드 서비스는 AWS SDK에서 Amazon GameLift 서비스 API 작업을 GameLift 호출하여 게임 클라이언트와 Amazon 간의 통신을 처리합니다. 또한 플레이어 인증 및 권한 부여, 인벤토리 또는 통

화 제어와 같은 기타 게임별 작업에 대한 백엔드 서비스도 사용할 수 있습니다. 자세한 정보는 게임 클라이언트 서비스 설계을 참조하세요.

외부 서비스

게임에서 구독 멤버십 확인 등과 같은 작업 시 외부 서비스를 사용할 수 있습니다. 외부 서비스는 백엔드 서비스와 GameLift Amazon을 통해 게임 서버에 정보를 전달할 수 있습니다.

게임 서버

GameLiftAmazon에 게임 서버 소프트웨어를 GameLift 업로드하면 Amazon이 호스팅 시스템에 배포하여 게임 세션을 호스팅하고 플레이어 연결을 수락합니다. 게임 서버는 Amazon과 GameLift 통신하여 게임 세션을 시작하고, 새로 연결된 플레이어를 확인하고, 게임 세션, 플레이어 연결 및 사용 가능한 리소스의 상태를 보고합니다.

사용자 지정 게임 서버는 Amazon GameLift Server SDK를 GameLift 사용하여 Amazon과 통신합니다. 게임 클라이언트는 백엔드 서비스를 GameLift 통해 Amazon으로부터 연결 세부 정보를 받은 후 게임 서버에 직접 연결합니다. 자세한 정보는 사용자 지정 게임 서버와 게임 통합을 참조하세요.

Realtime 서버는 사용자 지정 스크립트를 실행하는 게임 서버입니다. 게임에 참가할 때 게임 클라이언트는 Realtime Client SDK를 사용하여 Realtime 서버에 직접 연결됩니다. 자세한 정보는 Amazon GameLift Realtime 서버와 게임 통합을 참조하세요.

호스트 관리 도구

호스팅 리소스를 설정하고 관리할 때 게임 소유자는 호스팅 관리 도구를 사용하여 게임 서버 빌드나 스크립트, 플릿, 매치메이킹, 대기열을 관리합니다. AWS SDK와 콘솔의 Amazon GameLift 도구 세트는 호스팅 리소스를 관리할 수 있는 다양한 방법을 제공합니다. 문제 해결을 위해 개별 게임 서버에 원격으로 액세스할 수 있습니다.

설정

멀티플레이어 게임을 호스팅하도록 Amazon GameLift를 사용하려면 AWS 계정를 설정하는 방법에 대해 도움을 받습니다.

> **Tip**
>
> Amazon GameLift 게임 서버 호스팅을 사용해 보려면 Amazon GameLift 시작하기 섹션을 참조하세요.

주제

설정 AWS 계정

Amazon 사용을 시작하려면 GameLift 다음을 생성하고 설정하십시오 AWS 계정. AWS 계정를 만드는 데는 비용이 들지 않습니다. 이 섹션에서는 계정 생성, 사용자 설정, 권한 구성 과정을 안내합니다.

주제

가입하세요. AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

Amazon의 사용자 권한 관리 GameLift

Amazon GameLift 리소스에 필요한 경우 추가 사용자를 생성하거나 액세스 권한을 기존 사용자로 확장하십시오. 모범 사례(IAM의 보안 모범 사례)로써 모든 사용자에게 최소 권한 권한을 적용합니다. 권한 구문에 대한 지침은 Amazon GameLift에 사용되는 IAM 권한 예제을 참조하세요.

다음 지침에 따라 AWS 계정의 사용자 관리 방식에 따라 사용자 권한을 설정하십시오.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 내 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용 설명서의 권한 세트 생성의 지침을 따르세요.
- ID 제공자를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기(연합)의 지침을 따르세요.

- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용 설명서에서 IAM 사용자의 역할 생성의 지침을 따르세요.
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따르세요.

IAM 사용자와 작업할 때는 항상 개별 사용자가 아닌 역할이나 사용자 그룹에 권한을 할당하는 것이 가장 좋습니다.

사용자의 프로그래밍 방식 액세스 설정

사용자가 AWS 외부 사용자와 상호 작용하려는 경우 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명하십시오. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface • AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

액세스 키를 사용하는 경우 액세스 키 관리 AWS 모범 사례를 참조하십시오.

게임의 프로그래밍 방식 액세스 설정

대부분의 게임은 백엔드 서비스를 사용하여 AWS SDK를 GameLift 사용하여 Amazon과 통신합니다. 예를 들어 (게임 클라이언트를 대신하여 행동하는) 벡엔드 서비스를 사용하여 게임 세션을 요청하고 플레이어를 게임에 참여시키는 등의 작업을 요청합니다. 이러한 서비스에는 Amazon GameLift 서비스 API에 대한 호출을 인증하기 위한 프로그래밍 액세스 및 보안 자격 증명이 필요합니다.

Amazon의 GameLift 경우 AWS Identity and Access Management (IAM) 에서 플레이어 사용자를 생성하여 이 액세스를 관리합니다. 다음 옵션 중 하나를 사용하여 플레이어 사용자 권한을 관리합니다.

- 플레이어 사용자 권한이 있는 IAM 역할을 생성하고 필요할 때 플레이어 사용자가 역할을 맡을 수 있도록 허용합니다. 백엔드 서비스는 GameLift Amazon에 요청하기 전에 이 역할을 맡는 코드를 포함해야 합니다. 보안 모범 사례에 따라 역할은 제한된 임시 액세스를 제공합니다. AWS 리소스 (IAM 역할) 또는 외부 (IAM Roles Anywhere) 에서 실행되는 워크로드에 역할을 사용할 수 있습니다. AWS

- 플레이어 사용자 권한을 사용하여 IAM 사용자 그룹을 생성하고 플레이어 사용자를 그룹에 추가합니다. 이 옵션은 플레이어 사용자에게 장기 자격 증명을 제공하며, 이 자격 증명은 백엔드 서비스가 GameLift Amazon과 통신할 때 저장하고 사용해야 합니다.

권한 정책 구문은 플레이어 사용자 권한 예제 섹션을 참조하세요.

워크로드에서 사용할 권한을 관리하는 방법에 대한 자세한 내용은 IAM 자격 증명: IAM의 임시 보안 인증을 참조하세요.

Amazon GameLift에 사용되는 IAM 권한 예제

이 예제의 구문을 사용하여 Amazon GameLift 리소스에 액세스해야 하는 사용자에 대한 AWS Identity and Access Management(IAM) 권한을 설정합니다. 사용자 권한 관리에 대한 자세한 내용은 Amazon의 사용자 권한 관리 GameLift 섹션을 참조하세요. IAM Identity Center 외부 사용자의 권한을 관리할 때는 항상 개별 사용자가 아닌 IAM 역할 또는 사용자 그룹에 권한을 할당하는 것이 가장 좋습니다.

Amazon GameLift FleetIQ를 독립형 솔루션으로 사용하는 경우 Amazon GameLift FleetIQ의 AWS 계정 설정을 참조하세요.

관리자 권한 예제

이러한 예제는 사용자에게 Amazon GameLift 게임 호스팅 리소스를 관리할 수 있는 전체 액세스 권한을 제공합니다.

Example Amazon GameLift 리소스 권한의 구문

다음 예제는 모든 Amazon GameLift 리소스에 대한 액세스를 확장합니다.

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

Example 기본적으로 활성화되지 않은 리전을 지원하는 Amazon GameLift 리소스 권한 구문

다음 예제는 기본적으로 활성화되지 않은 모든 Amazon GameLift 리소스 및 AWS 리전으로 액세스를 확장합니다. 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전 관리를 참조하세요.

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

Example Amazon GameLift 리소스 및 **`PassRole`** 권한의 구문

다음 예제는 모든 Amazon GameLift 리소스에 대한 액세스를 확장하고 사용자가 IAM 서비스 역할을 Amazon GameLift에 전달할 수 있도록 허용합니다. Amazon에 대한 IAM 서비스 역할 설정 GameLift에 설명된 대로 Amazon GameLift는 서비스 역할을 사용하여 사용자를 대신하여 다른 리소스 및 서비스에 제한적으로 액세스할 수 있습니다. 예를 들어, `CreateBuild` 요청에 응답할 때 Amazon GameLift는 Amazon S3 버킷의 빌드 파일에 액세스할 수 있어야 합니다. `PassRole` 작업에 대한 자세한 내용은 IAM 사용 설명서의 IAM: IAM 역할을 특정 AWS 서비스로 전달을 참조하세요.

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

```
        }
      }
    }
  ]
}
```

플레이어 사용자 권한 예제

이러한 예제를 사용하면 백엔드 서비스 또는 기타 엔티티가 Amazon GameLift API에 대한 API 호출을 수행할 수 있습니다. 게임 세션, 플레이어 세션 및 매치메이킹을 관리하는 일반적인 시나리오를 다룹니다. 자세한 내용은 게임의 프로그래밍 방식 액세스 설정 섹션을 참조하세요.

Example 게임 세션 배치 권한의 구문

다음 예제는 게임 세션 배치 대기열을 사용하여 게임 세션을 생성하고 플레이어 세션을 관리하는 Amazon GameLift API에 대한 액세스를 확장합니다.

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

Example 매치메이킹 권한의 구문

다음 예시는 FlexMatch 매치메이킹 활동을 관리하는 Amazon GameLift API에 대한 액세스를 확장합니다. FlexMatch는 신규 또는 기존 게임 세션에서 플레이어를 매칭하고 Amazon GameLift에서 호스팅되는 게임에 게임 세션 배치를 시작합니다. FlexMatch에 대한 자세한 내용은 Amazon GameLift FlexMatch란 무엇인가요?를 참조하세요.

```
{
```

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

Example 게임 세션 배치 권한의 구문

다음 예제는 특정 플릿에서 게임 세션과 플레이어 세션을 수동으로 생성하는 Amazon GameLift API에 대한 액세스를 확장합니다. 이 시나리오는 플레이어가 사용 가능한 게임 세션 목록에서 선택하여 참여할 수 있게 하는 게임(“목록 및 선택” 방법)과 같이 배치 대기열을 사용하지 않는 게임을 지원합니다.

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

Amazon에 대한 IAM 서비스 역할 설정 GameLift

일부 Amazon GameLift 기능을 사용하려면 소유한 AWS 리소스에 대한 제한된 액세스 권한을 확장해야 합니다. AWS Identity and Access Management(IAM) 역할을 생성하여 이 작업을 수행할 수 있습니다. IAM 역할은 계정에 생성할 수 있는, 특정 권한을 지닌 IAM 자격 증명입니다. IAM 역할은 AWS에서 자격 증명으로 할 수 있는 것과 없는 것을 결정하는 권한 정책을 갖춘 AWS 자격 증명이라는 점에서 IAM 사용자와 유사합니다. 그러나 역할은 한 사람하고만 연관되지 않고 해당 역할이 필요한 사람이라면 누구든지 맡을 수 있어야 합니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준 장기 자격 증명이 없습니다. 대신에 역할을 맡은 사람에게는 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다.

이 주제에서는 Amazon GameLift 관리형 플릿에서 사용할 수 있는 역할을 생성하는 방법을 다룹니다. Amazon GameLift FleetiQ를 사용하여 Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스에서 게임 호스팅을 최적화하는 경우 Amazon FleetiQ용 설정을 참조하십시오. AWS 계정 GameLift

다음 절차에서 사용자 지정 권한 정책과 Amazon이 역할을 GameLift 맡도록 허용하는 신뢰 정책을 사용하여 역할을 생성합니다.

사용자 지정 IAM 역할 생성

1단계: 권한 정책 생성.

JSON 정책 편집기를 사용하여 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.

   정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작하기를 선택합니다.
3. 페이지 상단에서 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
5. JSON 정책 문서를 입력하거나 붙여 넣습니다. IAM 정책 언어에 대한 자세한 내용은 IAM JSON 정책 참조를 참조하세요.
6. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화

되도록 정책을 재구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 정책 재구성을 참조하십시오.

7. (선택 사항) AWS Management Console에서 정책을 만들거나 편집할 때 AWS CloudFormation 템플릿에서 사용할 수 있는 JSON 또는 YAML 정책 템플릿을 생성할 수 있습니다.

   이렇게 하려면 정책 편집기에서 작업을 선택한 다음 CloudFormation템플릿 생성을 선택합니다. AWS CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management 리소스 유형 참조를 참조하세요.
8. 정책에 권한 추가를 완료했으면 다음을 선택합니다.
9. 검토 및 생성 페이지에서 생성하는 정책의 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
10. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 정책에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.
11. 정책 생성을 선택하고 새로운 정책을 저장합니다.

2단계: Amazon이 GameLift 위임할 수 있는 역할을 생성합니다.

1. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
2. 신뢰할 수 있는 엔터티 선택 페이지에서 사용자 지정 신뢰 정책 옵션을 선택합니다. 이렇게 선택하면 사용자 지정 신뢰 정책 편집기가 열립니다.
3. 기본 JSON 구문을 다음과 같이 바꾼 후 다음을 선택합니다.

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

4. 권한 추가 페이지에서 1단계에서 만든 권한 정책을 찾아 선택합니다. 다음을 선택하여 계속 진행합니다.

5. 이름, 검토 및 생성 페이지에서 생성하는 역할에 대한 역할 이름 및 설명(선택 사항)을 입력합니다. 신뢰 엔터티 및 추가된 권한을 검토합니다.
6. 역할 생성 선택하여 새 역할을 저장합니다.

권한 정책 구문

- Amazon이 서비스 역할을 GameLift 맡을 수 있는 권한

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

- 기본적으로 활성화되지 않은 AWS 리전에 액세스할 수 있는 권한

```
{
  "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"
    }
  ]
```

```
}
```

아마존을 통한 개발 지원 GameLift

GameLift Amazon은 관리형 게임 호스팅 솔루션과 함께 사용할 수 있는 SDK 세트를 제공합니다. Amazon GameLift SDK를 사용하여 Amazon GameLift 호스팅 서비스와 상호 작용해야 하는 멀티플레이어 게임 서버, 게임 클라이언트 및 게임 서비스에 필요한 기능을 추가할 수 있습니다.

Amazon GameLift SDK 버전 및 SDK 호환성에 대한 최신 정보는 을 참조하십시오. 아마존 GameLift 릴리스 노트

사용자 지정 게임 서버의 경우

Amazon 서버 SDK를 사용하여 64비트 사용자 지정 게임 서버를 생성하고 GameLift 배포합니다. 서버 SDK와 통합되고 호스팅용으로 배포된 게임 서버는 Amazon GameLift 서비스와 통신하여 게임 세션을 시작하고 관리할 수 있습니다. Server SDK의 통합에 대한 자세한 내용은 아마존에서 게임을 준비하기 GameLift 섹션을 참조하세요.

개발 운영 체제

- Windows
- Linux

지원되는 프로그래밍 언어

GameLift Amazon은 다음 언어에 대한 서버 SDK를 제공합니다. 서버 SDK를 다운로드하세요. 버전별 자세한 정보는 각 패키지에 포함된 Readme 파일을 참조하세요.

- C++ Server SDK
  - SDK 참조
  - SDK 통합
- C# Server SDK(버전은 .NET 4 및 .NET 6 지원 가능)
  - SDK 참조
  - SDK 통합
- Go
  - SDK 참조

- SDK 통합

지원되는 게임 엔진

C++, C# 또는 Go 라이브러리를 지원하는 모든 엔진에서 언어별 SDK를 사용합니다. 또한 Amazon은 다음과 같은 게임 엔진 플러그인을 GameLift 제공합니다. Amazon GameLift 플러그인 다운로드

- Unity
  - Unity용 C# Server SDK 플러그인은 Unity 패키지 관리자를 사용하여 설치할 수 있는 사전 빌드된 라이브러리가 포함된 경량 플러그인입니다. 이 플러그인은 Windows 및 Mac OS용 2020.3 LTS, 2021.3 LTS, 2022.3 LTS와 같은 Unity 버전에서 사용할 수 있습니다. NET Standard 2.1 및 .NET 4.x가 있는 Unity의 .NET Framework 및 .NET Standard 프로필을 지원합니다.
    - Amazon GameLift를 Unity 프로젝트에 통합
  - Unity 2021.3 LTS 및 2022.3 LTS용 독립형 플러그인은 Unity용으로 빌드된 C# SDK 라이브러리와 호스팅용 Amazon 리소스를 구성 및 배포하기 위한 GUI 요소가 포함된 모든 기능을 갖춘 플러그인입니다. GameLift
    - 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드
    - C#용 Amazon GameLift Server SDK 참조
- Unreal Engine
  - Unreal용 C++ Server SDK 플러그인은 C++ Unreal 소스 코드로 구성된 경량 플러그인으로, Unreal Engine 버전 4, 5, 5.1에서 사용할 라이브러리로 빌드할 수 있습니다.
    - GameLift Amazon을 언리얼 엔진 프로젝트에 통합
    - Amazon GameLift Unreal Engine Server SDK 5.x 참조
  - 언리얼 엔진 5.0, 5.1, 5.2용 스탠드얼론 플러그인은 언리얼 서버 SDK 라이브러리 및 SDK용 C++가 포함된 모든 기능을 갖춘 플러그인입니다. AWS 플러그인은 호스팅용 Amazon GameLift 리소스를 구성 및 배포하기 위한 UI 요소 및 지원 자료와 함께 언리얼 에디터에 설치됩니다.
    - 언리얼 엔진용 Amazon GameLift 플러그인과 게임 통합
    - Amazon GameLift Unreal Engine Server SDK 5.x 참조

게임 서버 운영 체제

Amazon GameLift Server SDK를 사용하여 다음 플랫폼에서 실행할 게임 서버를 구축하십시오.

- Windows Server 2016
- Amazon Linux 2023

- Amazon Linux 2(AL2)
- 윈도우 서버 2012 (윈도우 2012의 경우 아마존 GameLift FAQ 참조)
- 아마존 리눅스 (AL1) (AL1에 대한 아마존 GameLift FAQ 참조)

사용자 지정 클라이언트 서비스의 경우

Amazon API와 함께 AWS SDK를 사용하여 64비트 사용자 지정 클라이언트 서비스를 생성합니다. GameLift 이 SDK를 사용하면 클라이언트 서비스가 게임 세션을 관리하고 플레이어를 GameLift Amazon에서 호스팅되는 게임에 참여시킬 수 있습니다. 시작하려면 AWS SDK를 다운로드하십시오. Amazon에서 SDK를 사용하는 방법에 대한 자세한 내용은 Amazon GameLift GameLift API 참조를 참조하십시오.

Realtime 서버의 경우

Realtime 서버를 구성하고 배포하여 멀티플레이어 게임을 호스팅할 수 있습니다. 게임 클라이언트가 실시간 서버에 연결할 수 있도록 하려면 Amazon GameLift Realtime 클라이언트 SDK를 사용하십시오. 게임 클라이언트는 이 SDK를 사용하여 Realtime 서버 및 이 서버에 연결된 다른 게임 클라이언트와 메시지를 교환합니다. 시작하려면 Amazon GameLift 실시간 클라이언트 SDK를 다운로드하십시오. 구성 정보에 대해서는 Realtime 서버용 게임 클라이언트 통합 섹션을 참조하세요.

SDK 지원

Realtime Client SDK에는 다음 언어를 위한 소스가 들어 있습니다.

- C#(.NET)

개발 환경

다음과 같이 지원되는 개발 운영 체제와 게임 엔진에 필요하므로 원본에서 SDK를 빌드합니다.

- 운영 체제 - Windows, Linux, Android, iOS
- 게임 엔진 - Unity, C# 라이브러리를 지원하는 엔진

게임 서버 운영 체제

Realtime 서버는 다음 플랫폼을 실행하는 호스팅 리소스에 배포될 수 있습니다.

- Amazon Linux

- Amazon Linux 2

게임 호스팅 비용 관리

AWS 청구서에는 게임 호스팅 비용이 반영되어 있습니다. 결제 콘솔 https://console.aws.amazon.com/billing/에서 이번 달의 예상 요금과 이전 달의 최종 요금을 확인할 수 있습니다. AWS 비용을 관리하는 데 도움이 되는 도구와 리소스에 대한 자세한 내용은 AWS Billing 사용 설명서를 참조하세요. 이 안내서는 리소스 소비를 검토하고, 향후 사용을 설정하며, 조정 요구를 결정하는 데 도움이 됩니다.

특히 Amazon GameLift 서비스 비용을 관리하는 데 도움이 되는 다음 팁을 고려해 보십시오.

결제 알림을 생성하여 사용량 모니터링

AWS프리 티어 사용 알림을 설정하여 사용량이 Amazon GameLift 및 기타 AWS 서비스 지역의 프리 티어 한도에 근접하거나 초과할 때 알림을 받을 수 있습니다. 사용량 수준에 따라 조치를 취하도록 알림을 구성할 수 있습니다. 예를 들어 프리 티어 한도에 도달하면 예산을 0으로 자동 설정할 수 있습니다.

사용량이 사용자 지정 임계값에 도달하면 알림을 받도록 Amazon CloudWatch 결제 알림을 설정할 수도 있습니다.

자세한 내용은 AWS Billing 사용 설명서의 다음 주제를 참조하세요.

- AWS 프리 티어 사용량 추적
- 결제 알림 기본 설정

Amazon GameLift 플릿별 비용 추적

AWS비용 할당 태그를 사용하여 Amazon GameLift Amazon EC2 플릿 및 기타 EC2 리소스를 기반으로 게임 호스팅 비용을 구성하고 추적할 수 있습니다. 플릿에 개별적으로 또는 그룹별로 태그를 지정하여 할당된 태그를 기준으로 비용을 분류하는 비용 할당 보고서를 생성할 수 있습니다. 이 보고서 유형을 사용하여 플릿이 호스팅 비용에 어떻게 기여하고 있는지 식별할 수 있습니다. AWS Cost Explorer에서 보기를 필터링하는 데 태그를 사용할 수도 있습니다.

자세한 내용은 다음 주제를 참조하십시오.

- AWS Billing 사용 설명서의 AWS 비용 할당 태그 사용
- AWS Cost Management 사용 설명서의 AWS Cost Explorer를 사용한 비용 분석

미사용 플릿 용량을 0으로 설정

플릿은 게임 세션을 호스팅하는 데 사용하지 않는 경우에도 계속 비용이 발생할 수 있습니다. 불필요한 요금이 발생하지 않게 하려면 사용하지 않는 동안 플릿 규모를 축소해야 할 수 있습니다. Auto Scaling을 사용하는 경우 이 활동을 일시 중단하고 플릿 용량을 수동으로 설정합니다.

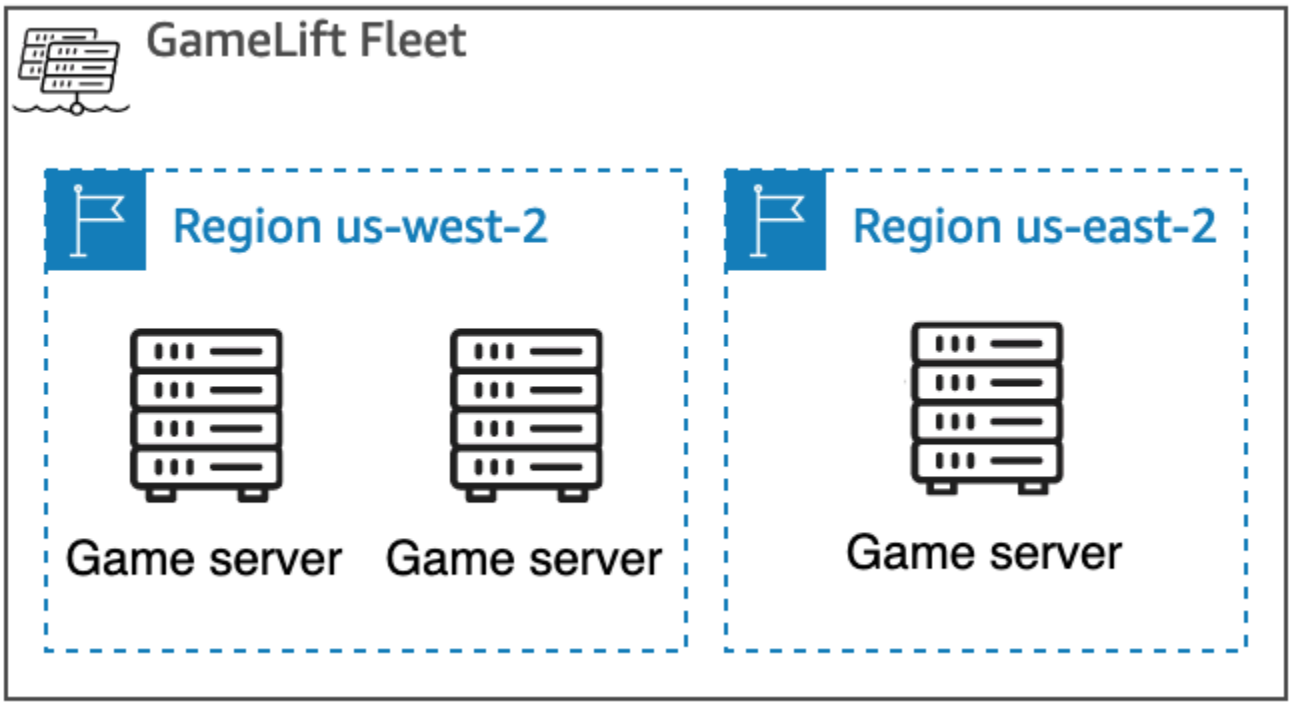
아마존 GameLift 호스팅 위치

GameLift Amazon은 여러 지역 AWS 리전 및 로컬 영역에서 사용할 수 있습니다. 전체 위치 목록은 의 Amazon GameLift 엔드포인트 및 할당량을 참조하십시오. AWS 일반 참조

아마존 GameLift 호스팅

Amazon GameLift 플릿을 생성하면 Amazon은 현재 플릿에 플릿의 리소스를 GameLift 생성합니다 AWS 리전. Amazon은 이 지역을 플릿의 홈 지역이라고 GameLift 부릅니다. 플릿을 관리하려면 홈 리전에서 플릿에 액세스합니다.

다중 위치 플릿은 플릿의 홈 리전 외에도 다른 위치에 인스턴스를 배포합니다. 여러 위치에 있는 플릿을 사용하면 각 위치의 용량을 개별적으로 관리하고 위치별로 게임 세션을 배치할 수 있습니다. 다중 위치 플릿은 GameLift Amazon이 지원하는 모든 지역 또는 로컬 영역에 원격 위치를 둘 수 있습니다. 다음 다이어그램은 두 리전의 리소스가 있는 다중 위치 플릿을 보여줍니다. 다이어그램에서 `us-west-2` 리전에는 두 개의 게임 서버가 있고 `us-east-2` 리전에는 한 개의 게임 서버가 있습니다.



기본적으로 활성화되지 않은 리전의 인스턴스와 함께 다중 위치 플릿을 사용하기로 선택한 경우 AWS 계정에서 해당 리전을 활성화해야 합니다. 또한 Amazon GameLift 관리자 정책에서 `ec2:DescribeRegions` 작업을 허용해야 합니다. 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전 관리를 참조하세요. 기본적으로 활성화되지 않는 리전의 정책 예제는 관리자 권한 예제 섹션을 참조하세요.

> **Important**
>
> 기본적으로 활성화되지 않은 리전을 사용하려면 AWS 계정에서 활성화합니다.
>
> - 2022년 2월 28일 이전에 생성한 리전 중 활성화되지 않은 리전의 플릿은 영향을 받지 않습니다.
> - 새 다중 위치 플릿을 생성하거나 기존의 다중 위치 플릿을 업데이트하려면 먼저 사용하려는 리전을 모두 활성화해야 합니다.

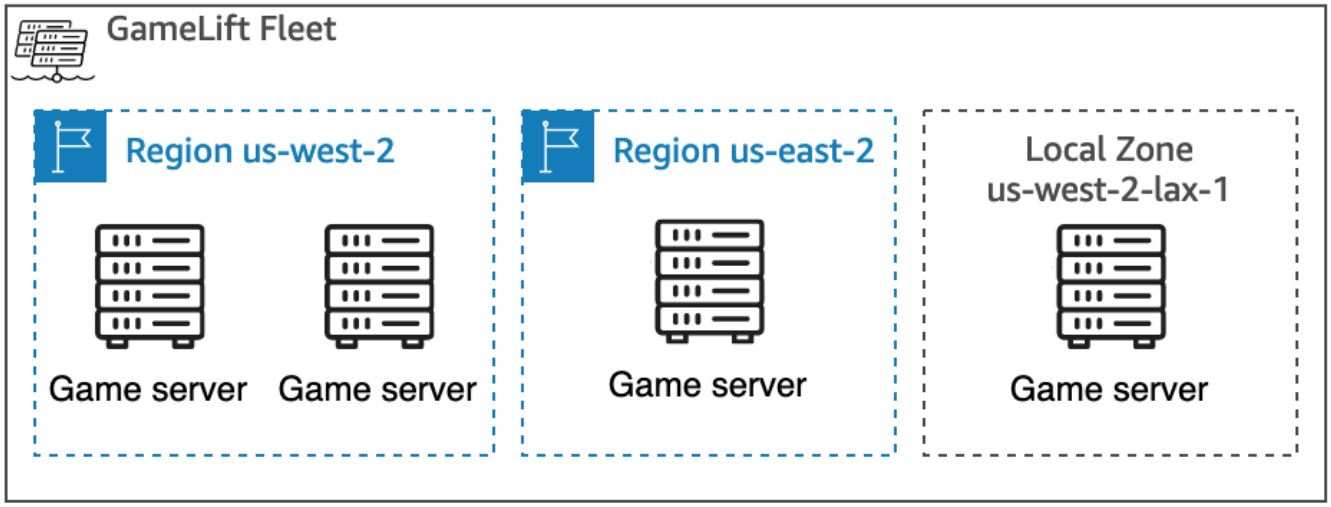
게임 세션 배치를 위해 GameLift Amazon이 지원하는 모든 위치에 게임 세션 대기열을 생성할 수 있습니다. Amazon은 대기열을 생성한 GameLift 위치에서 게임 세션을 배치합니다.

로컬 영역

로컬 영역은 사용자와 지리적으로 근접한 AWS 리전의 확장입니다. 로컬 영역은 인터넷에 대한 자체 연결을 가지고 있습니다. 로컬 영역은 AWS Direct Connect를 지원하므로 로컬 영역에 생성된 리소스를 짧은 지연 시간의 통신을 통해 로컬 사용자에게 제공할 수 있습니다. 자세한 내용은 AWS 로컬 영역을 참조하세요.

로컬 영역에 대한 코드는 물리적 위치를 나타내는 식별자가 뒤에 붙는 리전 코드입니다. 예를 들어 `us-west-2-lax-1` 로컬 영역은 로스앤젤레스에 있습니다. 사용 가능한 로컬 영역 목록은 사용 가능한 로컬 영역 섹션을 참조하세요.

Amazon은 플릿을 위해 선택한 각 위치에서 게임을 GameLift 호스팅합니다. 다음 다이어그램은 `us-west-2` 리전에 두 개의 게임 서버, `us-east-2` 리전에 한 개의 게임 서버, `us-west-2-lax-1` 로컬 영역에 한 개의 게임 서버가 있는 플릿을 보여줍니다.



사용 가능한 로컬 영역

다음 표에는 사용 가능한 로컬 영역과 물리적 위치가 나열되어 있습니다.

로컬 영역	위치(메트로 지역)	
`us-east-1-atl-1`	애틀랜타	
`us-east-1-chi-1`	시카고	
`us-east-1-dfw-1`	댈러스	

로컬 영역	위치(메트로 지역)	
us-east-1-iah-1	휴스턴	
us-east-1-mci-1	캔자스시티	
us-west-2-den-1	Denver	
us-west-2-lax-1	로스엔젤레스	
us-west-2-phx-1	피닉스	

아마존 GameLift Anywhere

GameLift AnywhereAmazon을 사용하여 자체 하드웨어로 플릿을 생성하고 GameLift Amazon을 사용하여 게임 빌드, 스크립트, 게임 서버 및 클라이언트를 관리할 수 있습니다. GameLift AnywhereAmazon은 Amazon이 GameLift 지원하는 모든 지역에서 사용할 수 있습니다. Anywhere 플릿 생성 및 게임 서버 통합 테스트에 대한 자세한 내용은 아마존 GameLift Anywhere 플릿 생성 및 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 섹션을 참조하세요.

GameLift AnywhereAmazon에서는 Amazon GameLift 통합 게임 서버를 호스팅하는 데 사용하는 하드웨어의 물리적 위치를 나타내는 사용자 지정 위치를 만들 수 있습니다.

아마존 GameLift FlexMatch

예를 들어 FlexMatch, GameLift Amazon이 지원하는 모든 위치에서 경기 생성 게임 세션을 호스팅할 수 있습니다. 실제 매치메이킹 활동은 매치메이커 리소스를 생성하기로 선택한 AWS 리전 곳에서 이루어집니다. Amazon은 매칭 요청을 매치메이커로 GameLift 라우팅하고 해당 위치에서 처리합니다. Amazon에 대한 자세한 내용은 GameLift FlexMatch Amazon이란 무엇입니까 GameLift FlexMatch? 를 참조하십시오.

AWS 리전 FlexMatch 리소스를 지원하는

중국 아마존 GameLift

Sinnet에서 운영하는 중국 (베이징) 지역 또는 NWCD에서 운영하는 중국 (닝샤) 지역의 리소스로 GameLift Amazon을 사용하는 경우 별도의 AWS (중국) 계정이 있어야 합니다. 중국 리전에서 일부 기능을 사용할 수 없다는 점을 참고하세요. 이러한 GameLift 지역에서 Amazon을 사용하는 방법에 대한 자세한 내용은 다음 리소스를 참조하십시오.

- 중국의 Amazon Web Services
- 아마존 GameLift (중국 아마존 웹 서비스 시작하기)

Amazon GameLift 시작하기

Amazon GameLift를 자체 게임에 사용하기 전에 다음 예를 시도해 보는 것이 좋습니다. 사용자 지정 게임 서버 예제는 Amazon GameLift 콘솔에서의 게임 호스팅 경험을 제공합니다. Realtime 서버 예제는 Realtime 서버를 사용하여 호스팅할 게임을 준비하는 방법을 보여줍니다.

자체 게임용 Amazon GameLift를 시작하려면 Amazon GameLift 관리형 호스팅 로드맵 섹션을 참조하세요.

사용자 지정 게임 서버 예제

이 예제는 Amazon GameLift의 라이브 사용자 지정 게임을 보여줍니다. 이 예제에서는 다음 단계를 살펴봅니다.

- 예제 게임 빌드를 생성합니다.
- 게임 서버를 실행하기 위한 플릿을 생성합니다.
- 예제 게임 클라이언트에서 게임 서버에 연결합니다.
- 플릿 및 게임 세션 지표를 검토합니다.

이러한 단계를 거친 후, 여러 게임 클라이언트를 시작하고 게임을 플레이하여 호스팅 데이터를 생성할 수 있습니다. 그런 다음, Amazon GameLift 콘솔을 탐색하여 호스팅 리소스를 보고, 지표를 추적하며, 호스팅 용량을 확장하는 방법을 경험할 수 있습니다.

시작하려면 Amazon GameLift 콘솔에 로그인합니다.

Realtime 서버 예제 게임

이 예제는 Mega Frog Race라는 완전한 멀티플레이어 게임으로, 소스 코드가 포함되어 있습니다. 이 예제에서는 게임 클라이언트를 Realtime 서버와 통합하는 방법을 보여 줍니다. 또한 이 예제 게임을 시작발점으로 사용하여 FlexMatch와 같은 다른 Amazon GameLift 기능을 시험해 볼 수도 있습니다.

실습 자습서를 읽으려면 게임 블로그용 AWS에서 Creating Servers for Multiplayer Mobile Games with Just a Few Lines of JavaScript를 참조하세요.

Mega Frog Race의 소스 코드는 GitHub 리포지토리를 참조하세요.

소스 코드에는 다음 부분이 포함됩니다.

- 게임 클라이언트 - Unity에서 생성된 C++ 게임 클라이언트에 대한 소스 코드입니다. 게임 클라이언트를 통해 게임 세션 연결 정보를 얻고, 서버에 연결하며, 다른 플레이어와 업데이트를 교환합니다.
- 백엔드 서비스 - Amazon GameLift로 직접 API 호출을 관리하는 AWS Lambda 함수의 소스 코드입니다.
- 실시간 스크립트 - 게임의 Realtime 서버 플릿을 구성하는 소스 스크립트 파일입니다. 이 스크립트에는 Realtime 서버가 Amazon GameLift와 통신하고 게임을 호스팅하는 데 필요한 최소 구성이 포함되어 있습니다.

Amazon GameLift 관리형 호스팅 로드맵

이 주제는 세션 기반 멀티플레이어 게임을 위한 다양한 Amazon GameLift 호스팅 옵션 중에서 선택할 수 있도록 도움을 줍니다. 이 섹션의 나머지 항목에서는 관리형 호스팅에 Amazon GameLift를 사용하는 방법에 대해 안내합니다.

게임 출시를 준비를 시작하기 전에 출시 설문지를 작성하여 Amazon GameLift 팀과 함께 작업을 시작합니다.

주제

호스팅 옵션 선택

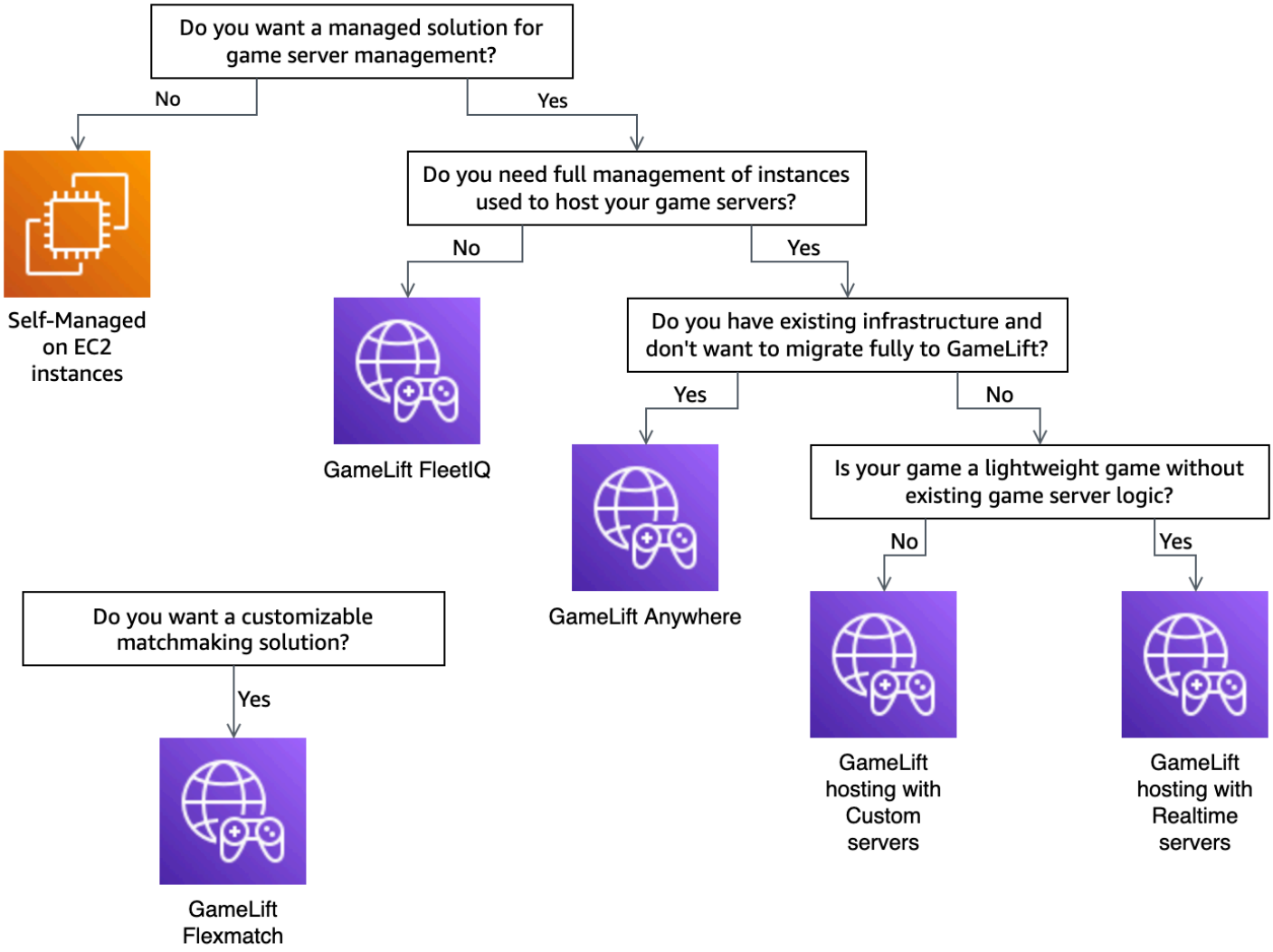
다음 흐름 차트는 사용 사례에 맞는 올바른 Amazon GameLift 솔루션으로 안내하기 위한 질문을 제공합니다.

1. 게임 서버 관리를 위한 관리형 솔루션을 원하십니까?
   - 예 - 계속해서 2단계를 진행합니다.
   - 아니요 - Amazon EC2 인스턴스의 자체 관리형 게임 서버를 고려합니다.
2. 게임 서버를 호스팅하는 인스턴스를 완전히 제어해야 합니까?
   - 예 - Amazon GameLift FleetIQ를 고려합니다.
   - 아니요 - 계속해서 3단계를 진행합니다.
3. Amazon GameLift와 함께 사용하려는 기존 인프라가 있습니까?
   - 예 - Amazon GameLift Anywhere를 고려합니다.

- 아니요 - 계속해서 4단계를 진행합니다.

4. 기존 게임 서버 로직이 없는 간소화된 게임입니까?

   - 예 - Realtime 서버를 고려합니다.
   - 아니요 - 사용자 지정 서버를 고려합니다.



다음은 흐름 차트에 언급된 일부 Amazon GameLift 호스팅 옵션에 대한 추가 정보입니다.

Amazon GameLift Anywhere

Amazon GameLift Anywhere를 사용하면 Amazon GameLift 관리 도구의 이점을 활용하여 자체 하드웨어에서 게임을 호스팅할 수 있습니다. Anywhere 플릿을 사용하여 게임 서버를 반복적으로 테스트할 수도 있습니다. 자세한 내용은 아마존 GameLift Anywhere 플릿 생성 섹션을 참조하세요.

관리형 Amazon GameLift

관리형 Amazon GameLift 호스팅에는 다음의 두 가지 옵션이 있습니다.

사용자 지정 서버 - Amazon GameLift는 게임 서버 바이너리를 실행하는 사용자 지정 서버를 호스팅합니다.

Realtime 서버 - Amazon GameLift는 경량 게임 서버를 호스팅합니다.

Amazon GameLift FleetIQ

흐름 차트에서 리프트 앤드 시프트 마이그레이션이란 게임 아키텍처를 변경할 수 없는 경우의 마이그레이션을 말합니다. Amazon GameLift FleetIQ를 사용하면 기존 배포를 변경할 필요가 없으며 플릿 관리를 위한 Amazon GameLift 도구를 제공합니다. 자세한 내용을 알아보려면 Amazon GameLift FleetIQ 개발자 가이드를 참조하세요.

Amazon GameLift Anywhere를 사용하거나 관리형 Amazon GameLift를 사용하기로 결정했다면 Amazon GameLift의 게임 준비 섹션으로 계속 진행합니다.

Amazon GameLift의 게임 준비

이 주제에서는 관리형 Amazon GameLift 호스팅과의 통합을 위해 멀티플레이어 게임을 준비하는 단계를 설명합니다. 게임을 준비하려면 게임과 Amazon GameLift 간의 통신을 활성화해야 합니다.

사용자 지정 게임 서버 준비

게임 세션을 시작/중지하고 다른 작업을 수행하려면 게임 서버가 해당 상태를 Amazon GameLift에 알릴 수 있어야 합니다. Amazon GameLift와의 통신을 활성화하려면 게임 서버 프로젝트에 코드를 추가합니다. 자세한 내용은 사용자 지정 게임 서버와 게임 통합 섹션을 참조하세요.

1. Amazon GameLift에서 호스팅하기 위한 사용자 지정 게임 서버를 준비합니다.
   - Amazon GameLift Server SDK를 가져와서 원하는 프로그래밍 언어와 게임 엔진에 맞게 빌드합니다.
   - Amazon GameLift와의 통신을 활성화하려면 게임 서버 프로젝트에 코드를 추가합니다.
2. Amazon GameLift 게임 세션에 게임 클라이언트 연결을 준비합니다.
   - AWS SDK를 백엔드 서비스 및 게임 클라이언트 프로젝트에 추가합니다. 자세한 내용은 클라이언트 서비스용 Amazon GameLift SDK 다운로드를 참조하세요.

- 게임 세션에 대한 정보를 검색하고, 새 게임 세션을 배치하며, 게임 세션에서 플레이어를 위한 공간을 예약하는 기능을 추가합니다.
- (선택 사항) 플레이어 매치메이킹에 FlexMatch를 사용합니다. 자세한 내용은 Amazon GameLift 호스팅과 FlexMatch 통합을 참조하세요.

Realtime 서버 준비

Amazon GameLift Realtime 서버는 게임에 맞게 구성할 수 있는 경량 서버 솔루션을 제공합니다. Realtime 서버는 Amazon GameLift가 게임 서버에 제공하는 것과 동일한 이점을 제공하지만, 게임 서버 사용자 지정 가능성은 낮습니다.

Amazon GameLift에서 호스팅하기 위한 Realtime 스크립트를 생성합니다.

Realtime 스크립트에는 서버 구성 및 선택적 사용자 지정 게임 로직이 포함됩니다. Realtime 서버는 게임 세션을 시작 및 중지하고, 플레이어 연결을 수락하며, Amazon GameLift와의 통신 및 게임 내 플레이어 간의 통신을 관리하도록 구축되었습니다. 게임에 사용자 지정 서버 로직을 추가할 수 있는 후크도 있습니다. Realtime 서버는 Node.js 및 JavaScript를 사용합니다. 자세한 정보는 Realtime 스크립트 생성 및 Amazon GameLift와의 통합 테스트 섹션을 참조하세요.

Amazon GameLift와의 통합 테스트

Amazon GameLift는 게임 서버를 테스트할 때 빠른 반복을 지원합니다. 이 주제에서는 사용 가능한 테스트 유형을 살펴봅니다.

사용자 지정 게임 서버

Amazon GameLift를 사용하여 사용자 환경 내 어디에서나 하드웨어를 Amazon GameLift 게임 호스팅 아키텍처에 통합합니다. Amazon GameLift Anywhere는 하드웨어를 Anywhere 플릿의 Amazon GameLift에 등록하므로, 사용자가 자체 로컬 개발 컴퓨터를 사용하여 테스트할 수 있습니다. Amazon GameLift Anywhere를 사용하여 테스트하는 방법에 대한 자세한 내용은 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 섹션을 참조하세요. Amazon GameLift Anywhere를 사용하여 온프레미스 솔루션으로 게임을 호스팅하는 방법에 대한 자세한 내용은 Amazon GameLift 컴퓨팅 리소스 선택 섹션을 참조하세요.

Realtime 서버

Realtime 서버를 사용하면 언제든지 스크립트를 업데이트할 수 있습니다. Realtime 스크립트를 업데이트하면 Amazon GameLift는 몇 분 내에 호스팅 리소스에 새 버전을 배포합니다. Amazon GameLift에서 새 스크립트를 배포한 후에는 모든 새 게임 세션에서 새 스크립트 버전을 사용합니다. Amazon

GameLift에서 새 스크립트를 배포하는 즉시 테스트를 시작할 수 있습니다. Realtime 서버에 대한 자세한 내용은 Amazon GameLift Realtime 서버와 게임 통합 섹션을 참조하세요.

Amazon GameLift 리소스 계획 및 배포

다음 팁을 사용하면 Amazon GameLift 리소스 배포를 계획하는 데 도움이 됩니다. Amazon GameLift로 게임을 호스팅할 수 있는 위치에 대한 자세한 내용은 아마존 GameLift 호스팅 위치 섹션을 참조하세요.

Amazon GameLift 리소스를 배포하려면 다음 작업을 완료합니다.

- 게임 서버를 패키징하여 Amazon GameLift 또는 하드웨어에 업로드합니다. 서버를 Amazon GameLift에 업로드할 때는 플릿의 홈 AWS 리전에만 서버를 업로드합니다. Amazon GameLift는 서버를 플릿 내 다른 위치에 자동으로 배포합니다. 자세한 내용은 Amazon GameLift에 빌드 및 스크립트 업로드 섹션을 참조하세요.
- 게임을 위한 Amazon GameLift 플릿을 설계하고 배포합니다. 사용할 컴퓨팅 리소스의 유형, 배포할 위치, 대기열 사용 여부 및 기타 옵션을 결정합니다. 자세한 내용은 Amazon GameLift 플릿 설계 가이드 섹션을 참조하세요.
- 대기열을 생성하여 새 게임 세션 배치 및 스팟 인스턴스 전략을 관리합니다. 자세한 내용은 게임 세션 대기열 설계 섹션을 참조하세요.
- 예상되는 플레이어 수요에 맞게 플릿의 호스팅 용량을 관리하려면 Auto Scaling을 사용합니다. 자세한 내용은 Amazon GameLift 호스팅 용량 확장 섹션을 참조하세요.
- 게임에 FlexMatch 매치메이킹 규칙을 사용합니다. 자세한 내용은 Amazon GameLift 호스팅과 FlexMatch 통합을 참조하세요.

Amazon GameLift 리소스 자동 배포

Amazon GameLift 리소스의 글로벌 배포를 간소화하려면 코드형 인프라(IaC)를 사용하여 리소스를 정의하는 것이 좋습니다. Amazon GameLift는 AWS CloudFormation 템플릿을 지원하므로 모든 배포별 구성에 대한 파라미터를 템플릿에 설정할 수 있습니다.

AWS CloudFormation 스택 배포를 관리하려면 AWS CodePipeline과 같은 지속적 통합 및 지속적인 전송(CI/CD) 도구와 서비스를 사용하는 것이 좋습니다. 이를 통해 게임 서버 바이너리를 빌드할 때마다 자동으로 배포하거나 승인을 받아 배포할 수 있습니다.

다음은 CI/CD 도구 또는 서비스를 사용하여 자동화할 수 있는 새 게임 서버 버전에 대한 Amazon GameLift 리소스 배포의 몇 가지 일반적인 단계입니다.

- 게임 서버 바이너리를 빌드하고 테스트합니다.
- 바이너리를 Amazon GameLift 또는 하드웨어에 업로드합니다.
- 새 빌드에 새 플릿을 배포합니다.
- 새 플릿을 배포한 후 Amazon GameLift 대기열에서 이전 버전 플릿을 제거하고 새 플릿으로 교체합니다.
- 이전 버전 플릿이 모든 게임 세션을 성공적으로 종료하면 해당 플릿의 AWS CloudFormation 스택이 삭제됩니다.

또한 AWS Cloud Development Kit (AWS CDK)를 사용하여 Amazon GameLift 리소스를 정의할 수 있습니다. AWS CDK에 대한 자세한 내용은 AWS Cloud Development Kit (AWS CDK) 개발자 가이드 섹션을 참조하세요.

게임 클라이언트 서비스 설계

플레이어를 인증하고 Amazon GameLift API와 통신하는 게임 클라이언트 서비스를 구현하는 것이 좋습니다. 사용자 지정 게임 클라이언트 서비스를 구현하여 다음을 수행할 수 있습니다.

- 플레이어에 맞게 인증을 사용자 지정할 수 있습니다.
- Amazon GameLift가 게임 세션을 매칭하고 시작하는 방식을 제어할 수 있습니다.
- 클라이언트를 신뢰하는 대신 매치메이킹을 위한 스킬 등급과 같은 플레이어 속성에 대해 플레이어 데이터베이스를 사용할 수 있습니다.

또한, 게임 클라이언트 서비스를 사용하면 게임 클라이언트가 Amazon GameLift API와 직접 상호 작용할 때 발생하는 보안 위험도 줄일 수 있습니다.

플레이어 인증

Amazon Cognito 및 플레이어 세션 ID를 사용하여 게임 클라이언트를 인증할 수 있습니다. 플레이어 자격 증명의 수명 주기 및 속성을 관리하려면 Amazon Cognito 사용자 풀을 사용합니다.

원하는 경우, 사용자 지정 자격 증명 솔루션을 구축하여 AWS에 호스팅합니다. 또한, API Gateway를 통해 사용자 지정 권한 부여 로직에 Lambda 권한 부여자를 사용할 수 있습니다.

추가 리소스:

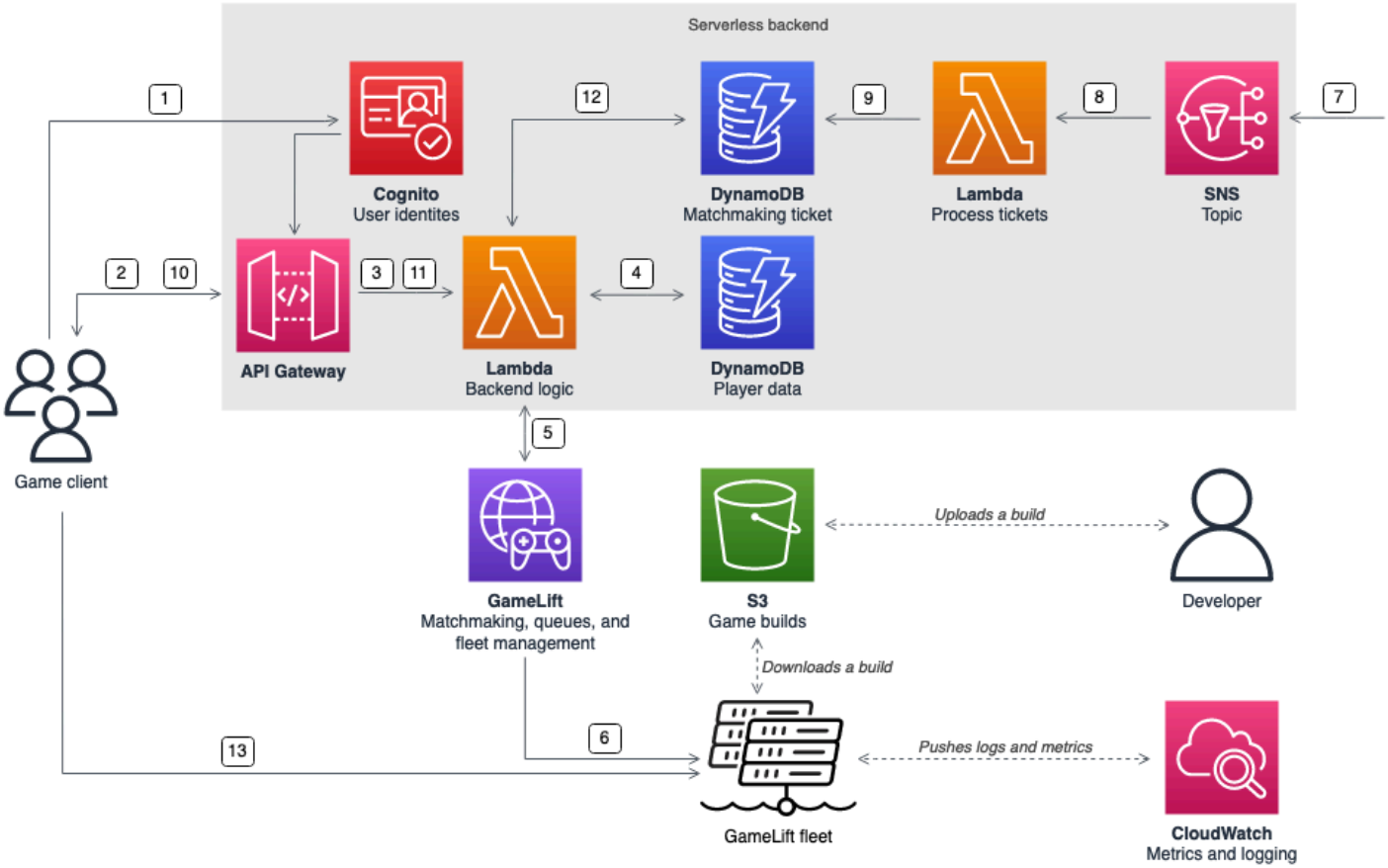
- 자격 증명 풀 사용(페더레이션 자격 증명)(Amazon Cognito 개발자 가이드)

- 사용자 풀 시작하기(Amazon Cognito 개발자 가이드)
- Amazon Cognito를 사용하여 플레이어 인증을 설정하는 방법(게임 블로그용 AWS)

서버리스 백엔드가 있는 독립형 게임 세션 서버

백엔드는 서버리스 클라이언트 서비스 아키텍처를 사용하여 Amazon GameLift API에 직접 액세스하는 대신 확장성이 뛰어난 데이터베이스에서 매치메이킹 티켓의 상태를 볼 수 있습니다.

다음 다이어그램은 Amazon GameLift 플릿에서 실행되는 게임에 플레이어를 매칭하도록 구축된 AWS 서비스 서버리스 백엔드를 보여줍니다. 다음 목록은 다이어그램에서 번호가 매겨진 각 콜아웃에 대한 설명을 제공합니다. 이 예제를 사용해 보려면 GitHub의 Multiplayer Session-based Game Hosting on AWS(AWS의 멀티플레이어 세션 기반 게임 호스팅)을 참조하세요.



1. 게임 클라이언트가 Amazon Cognito 자격 증명 풀에서 Amazon Cognito 사용자 자격 증명 풀에 요청합니다.
2. 게임 클라이언트는 Amazon API Gateway API를 통해 임시 액세스 자격 증명을 받고 게임 세션을 요청합니다.

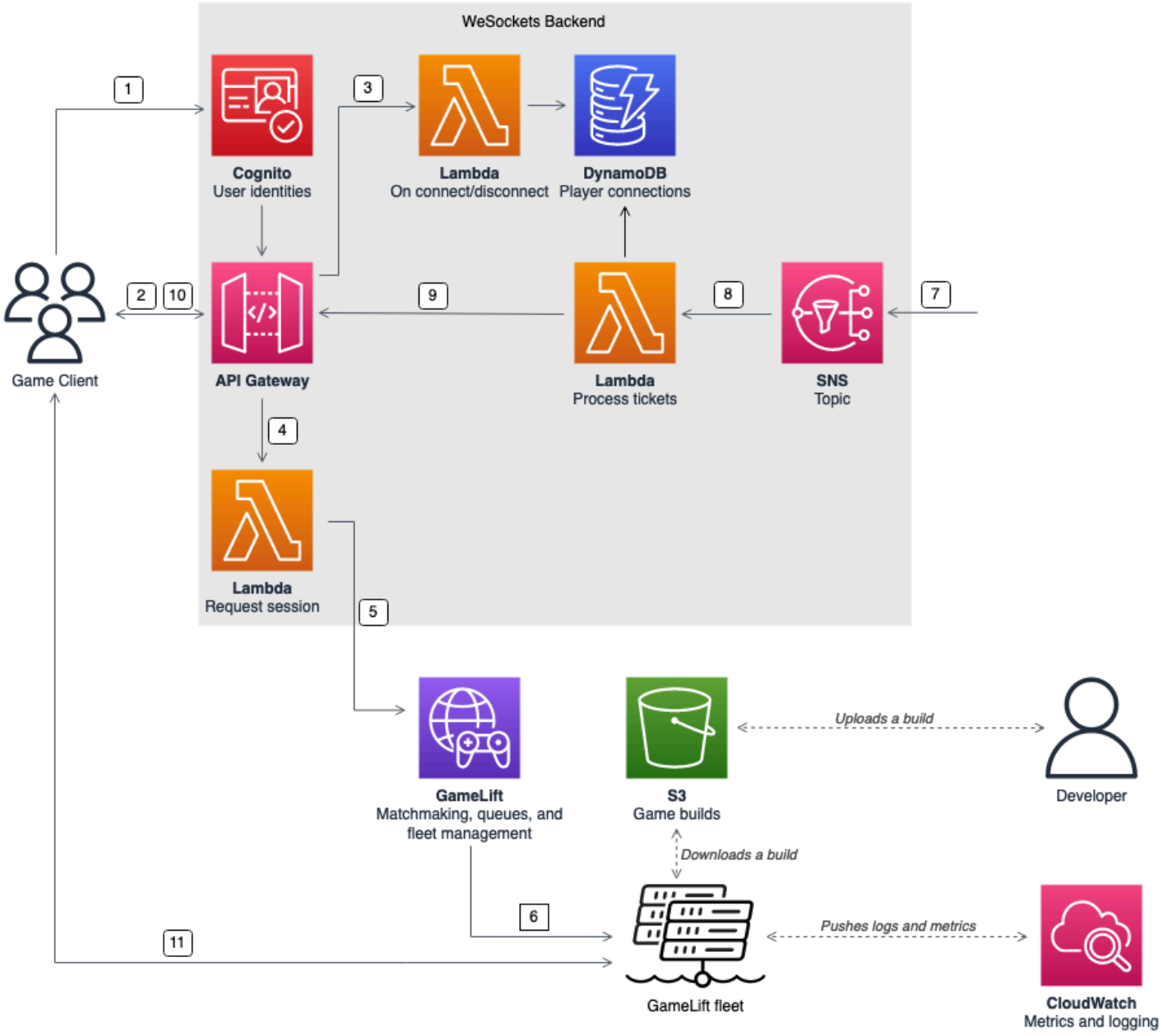
3. API Gateway는 AWS Lambda 함수를 호출합니다.
4. Lambda 함수는 Amazon DynamoDB NoSQL 테이블에서 플레이어 데이터를 요청합니다. 이 함수는 요청 컨텍스트 데이터에서 Amazon Cognito 자격 증명을 제공합니다.
5. Lambda 함수는 Amazon GameLift FlexMatch 매치메이킹을 통해 매칭을 요청합니다.
6. FlexMatch는 적절한 지연 시간으로 플레이어 그룹을 매칭한 다음 Amazon GameLift 대기열을 통해 게임 세션 배치를 요청합니다. 대기열에는 하나 이상의 AWS 리전 위치가 포함된 플릿이 있습니다.
7. Amazon GameLift가 플릿 위치 중 하나에 세션을 배치하면 Amazon GameLift는 Amazon Simple Notification Service(SNS) 주제에 이벤트 알림을 전송합니다.
8. Lambda 함수는 Amazon SNS 이벤트를 수신하고 처리합니다.
9. 매치메이킹 티켓이 `MatchmakingSucceeded` 이벤트인 경우 Lambda 함수는 게임 서버의 포트 및 IP 주소와 함께 결과를 DynamoDB 테이블에 기록합니다.
10. 게임 클라이언트는 API Gateway에 서명된 요청을 보내 특정 간격의 매치메이킹 티켓 상태를 확인합니다.
11. API Gateway는 매치메이킹 티켓 상태를 확인하는 Lambda 함수를 사용합니다.
12. Lambda 함수는 DynamoDB 테이블을 검사하여 티켓이 성공했는지 확인합니다. 성공하면 함수는 플레이어 세션 ID와 함께 게임 서버의 포트 및 IP 주소를 클라이언트에 다시 보냅니다. 티켓이 성공하지 못한 경우 함수는 매치가 아직 준비되지 않았음을 확인하는 응답을 보냅니다.
13. 게임 클라이언트는 백엔드 서비스가 제공하는 포트와 IP 주소를 사용하여 TCP 또는 UDP로 게임 서버에 연결합니다. 그런 다음 게임 클라이언트는 플레이어 세션 ID를 게임 서버로 전송하고, 게임 서버는 Amazon GameLift Server SDK를 사용하여 ID를 검증합니다.

WebSocket 기반 백엔드가 포함된 독립형 게임 세션 서버

Amazon API Gateway WebSocket 기반 아키텍처를 사용하면 WebSocket으로 매치메이킹을 요청하고 서버에서 시작한 메시지를 사용하여 매치메이킹 완료를 위한 푸시 알림을 보낼 수 있습니다. 이 아키텍처는 클라이언트와 서버 간에 양방향 통신을 통해 성능을 개선합니다.

API Gateway WebSocket API 사용에 대한 자세한 내용은 WebSocket API 작업을 참조하세요.

다음 다이어그램은 API Gateway 및 기타 AWS 서비스를 사용하여 플레이어를 Amazon GameLift 플릿에서 실행되는 게임에 연결하는 WebSocket 기반 백엔드 아키텍처를 보여줍니다. 다음 목록은 다이어그램에서 번호가 매겨진 각 콜아웃에 대한 설명을 제공합니다.



1. 게임 클라이언트가 Amazon Cognito 자격 증명 풀에서 Amazon Cognito 사용자 자격 증명 풀에 요청합니다.
2. 게임 클라이언트는 Amazon Cognito 자격 증명을 사용하여 API Gateway API에 대한 WebSocket 연결을 서명합니다.
3. API Gateway는 연결에서 AWS Lambda 함수를 호출합니다. 이 함수는 Amazon DynamoDB 테이블에 연결 정보를 저장합니다.
4. 게임 클라이언트는 WebSocket 연결에서 API Gateway API를 통해 Lambda 함수에 메시지를 보내 세션을 요청합니다.

5. Lambda 함수는 Amazon GameLift FlexMatch 매치메이킹을 통해 매칭을 요청합니다.
6. FlexMatch는 플레이어 그룹을 매칭한 다음 Amazon GameLift 대기열을 통해 게임 세션 배치를 요청합니다.
7. Amazon GameLift가 플릿 위치 중 하나에 세션을 배치하면 Amazon GameLift는 Amazon Simple Notification Service(SNS) 주제에 이벤트 알림을 전송합니다.
8. Lambda 함수는 Amazon SNS 이벤트를 수신하고 처리합니다.
9. 매치메이킹 티켓이 `MatchmakingSucceeded` 이벤트인 경우 Lambda 함수는 DynamoDB에 올바른 플레이어 연결을 요청합니다. 그러면 해당 함수는 WebSocket 연결에서 API Gateway API를 통해 게임 클라이언트에 메시지를 보냅니다. 이 아키텍처에서는 게임 클라이언트가 매치메이킹 상태를 적극적으로 폴링하지 않습니다.
10. 게임 클라이언트는 WebSocket 연결을 통해 플레이어 세션 ID와 함께 게임 서버의 포트 및 IP 주소를 수신합니다.
11. 게임 클라이언트는 백엔드 서비스가 제공하는 포트와 IP 주소를 사용하여 TCP 또는 UDP로 게임 서버에 연결합니다. 또한 게임 클라이언트는 플레이어 세션 ID를 게임 서버로 전송하고, 게임 서버는 Amazon GameLift Server SDK를 사용하여 ID를 검증합니다.

Amazon GameLift에 대한 지표 및 로깅 설정

Amazon GameLift 게임 서버 및 리소스에서 수집한 데이터를 사용하여 이상 현상을 식별할 수 있습니다. 또한 지표를 사용하면 성능을 개선하는 데 도움이 됩니다.

Amazon GameLift에서 관찰해야 할 주요 영역은 다음과 같습니다.

- Amazon GameLift 서비스 지표 - Amazon GameLift는 게임 서버, 플릿, 대기열 및 FlexMatch를 포함한 리소스에 대한 Amazon CloudWatch 지표를 제공합니다. Amazon GameLift 콘솔과 CloudWatch 콘솔에서 이러한 지표를 찾을 수 있습니다. CloudWatch에서 Amazon GameLift 지표에 대한 자세한 내용은 Amazon CloudWatch를 사용한 Amazon GameLift 모니터링 섹션을 참조하세요.
- 게임 서버 지표 - Amazon GameLift는 게임 서버 지표에 액세스할 수 없습니다. 하지만 CloudWatch 에이전트를 사용하여 게임 서버에서 직접 사용자 지정 지표를 CloudWatch로 보낼 수 있습니다. 플릿 AWS Identity and Access Management (IAM) 역할 및 AWS SDK를 사용하여 지표를 CloudWatch에 직접 전송할 수도 있습니다. 지표를 구성하는 방법에 대한 예는 GitHub의 Multiplayer Session-based Game Hosting on AWS(AWS 멀티플레이어 세션 기반 게임 호스팅) 섹션을 참조하세요. 이 리포지토리에는 C# StatSD 클라이언트용 예제 CloudWatch 에이전트 구성 및 코드가 포함되어 있습니다.

- 게임 서버 로그 - 게임 서버에서 게임 서버 로그 파일을 구성하려면 Amazon GameLift Server SDK 구성을 사용합니다. 또한 Amazon CloudWatch Logs를 실시간 로그 관리 솔루션으로 사용할 수 있으며 CloudWatch 에이전트를 사용하여 로그를 구성할 수 있습니다. 자세한 내용은 Amazon GameLift에서 서버 메시지 로깅 섹션을 참조하세요.

게임 출시 체크리스트

이 체크리스트를 사용하여 게임 배포 단계를 검증할 수 있습니다. 체크리스트에서 [중요] 로 표시된 항목은 프로덕션 출시에 매우 중요합니다.

주제

온보딩

다음 체크리스트를 사용하여 Amazon GameLift 호스팅을 위한 게임 온보딩에 항목을 추적합니다. [중요]로 표시된 항목은 프로덕션 출시에 매우 중요합니다.

- [중요] Amazon GameLift 콘솔에서 Amazon GameLift 온보딩 설문지를 작성합니다.
- [중요] 게임 클라이언트가 게임 서버와 상호 작용할 수 있도록 백엔드 서비스를 설계하고 구현합니다.
- [중요] 다른 AWS 리소스에 액세스할 수 있도록 Amazon GameLift 서버 인스턴스에 제공하는 AWS Identity and Access Management(IAM) 역할을 생성합니다.
- [중요] FlexMatch 및 대기열에 대해 다른 AWS 리전으로의 장애 조치를 설계하고 구현합니다.
- 게임의 대기열과 플릿 구조를 고려하여 대상 위치로의 플릿 롤아웃을 계획합니다.
- AWS CloudFormation 및 AWS Cloud Development Kit (AWS CDK)가 포함된 코드형 인프라(IaC)를 사용하여 배포를 자동화합니다.
- Amazon CloudWatch 및 Amazon Simple Storage Service(S3)를 사용하여 로그와 분석을 수집합니다.

테스트

Amazon GameLift 호스팅으로 게임을 개발하는 동안 다음 체크리스트를 사용하여 테스트 항목을 추적합니다. [중요]로 표시된 항목은 프로덕션 출시에 매우 중요합니다.

- [중요] 출시 설문지를 작성하고 작성한 설문지를 Amazon GameLift 출시 팀에 제출합니다. Amazon GameLift 콘솔에서 출시 설문지를 찾을 수 있습니다.
- [중요] 실제 환경을 프로덕션 요구 사항에 맞게 확장할 수 있도록 Amazon GameLift Service Quotas 및 기타 AWS 서비스 할당량 증가를 요청합니다.
- [중요] 라이브 플릿의 열린 포트가 서버에서 사용할 수 있는 포트 범위와 일치하는지 확인합니다.
- [중요] RDP 포트 3389 및 SSH 포트 22를 닫습니다.
- 게임의 DevOps 관리를 위한 계획을 개발합니다. Amazon CloudWatch Logs 또는 Amazon CloudWatch 사용자 지정 지표를 사용하는 경우 서버 플릿에서 심각하거나 중요한 문제에 대한 경보를 정의합니다. 장애를 시뮬레이션하고 런북을 테스트합니다.
- 인스턴스에서 최대 사용량으로 실행되는 서버의 수가 해당 서버 인스턴스 유형의 용량 이내인지 확인합니다.
- 처음에는 좀 더 보수적으로 조정 정책을 조정하고 필요하다고 생각하는 것보다 더 많은 유휴 용량을 제공합니다. 나중에 비용을 최적화할 수 있습니다. 유휴 용량이 20%인 대상 기반 크기 조정 정책을 사용하는 것을 고려합니다.
- FlexMatch 지연 시간 규칙을 사용하여 지리적으로 동일한 AWS 리전에 있는 플레이어를 매칭할 수 있습니다. 로드 테스트 클라이언트의 합성 지연 시간 데이터를 사용하여 부하 상태에서 어떻게 작동하는지 테스트합니다.
- 플레이어 인증 및 게임 세션 인프라를 로드 테스트하여 수요에 맞게 효과적으로 확장되는지 확인합니다.
- 며칠 동안 실행한 서버가 여전히 연결을 허용할 수 있는지 확인합니다.
- 문제나 정전 발생 시 AWS가 응답할 수 있도록 AWS Support 플랜 수준을 Business 또는 Enterprise로 높입니다.

시작

다음 체크리스트를 사용하여 Amazon GameLift에 호스팅된 게임의 출시 항목을 추적합니다. [중요]로 표시된 항목은 프로덕션 출시에 매우 중요합니다.

- [중요] 모든 라이브 플릿에 대해 플릿 보호 정책을 전체 보호로 설정하여 규모를 축소해도 활성 게임 세션이 중단되지 않도록 합니다.

- [중요] 최소한으로 최대 예상 수요를 수용할 수 있을 만큼 높은 최대 플릿 크기를 크게 설정합니다. 예상치 못한 수요에 대비하여 최대 크기를 두 배로 늘리는 것이 좋습니다.
- 개발팀 전체가 출시 이벤트에 참여하고 런칭 룸에서 게임 출시를 모니터링하도록 권장하세요.
- 플레이어 지연 시간과 플레이어 경험을 모니터링합니다.

출시 이후

다음 체크리스트를 사용하여 Amazon GameLift에 호스팅된 게임의 출시 이후 항목을 추적합니다.

- 크기 조정 규칙을 조정하여 유휴 용량을 최소화합니다.
- 지연 시간 요구 사항에 따라 FlexMatch 규칙을 수정하거나 위치를 추가합니다.
- 성능 효율성이 플릿 비용에 직접적인 영향을 미치므로 서버 실행 파일을 최적화합니다. 동일한 인프라로 더 많은 게임 세션을 실행하려면 인스턴스당 서버 프로세스 수를 늘립니다.
- 분석 데이터를 사용하여 지속적인 개발을 주도하고, 플레이어 경험과 게임 수명을 개선하며, 수익 창출을 최적화합니다.

아마존에서 게임을 준비하기 GameLift

GameLiftAmazon에서 멀티플레이어 게임을 호스팅할 수 있도록 준비하려면 게임과 Amazon 간의 통신을 설정하십시오 GameLift. 이 섹션의 항목에서는 게임을 Amazon GameLift, 사용자 지정 게임 서버, 실시간 서버와 통합하고 매치메이킹을 추가하는 방법에 대한 자세한 도움말을 제공합니다. FlexMatch

주제

- 사용자 지정 게임 서버와 게임 통합
- Amazon GameLift Realtime 서버와 게임 통합
- Unity용 Amazon GameLift 플러그인과 게임 통합
- 언리얼 엔진용 Amazon GameLift 플러그인과 게임 통합
- Amazon GameLift 인스턴스의 플릿 데이터를 가져옵니다.
- FlexMatch 매치메이킹 추가

사용자 지정 게임 서버와 게임 통합

Amazon GameLift에는 멀티플레이어 게임과 사용자 지정 게임 서버를 Amazon GameLift에서 실행하도록 준비할 때 필요한 전체 도구 세트가 갖추어져 있습니다. Amazon GameLift SDK에는 게임 클라이언트와 게임 서버가 Amazon GameLift와 통신하는 데 필요한 라이브러리가 포함되어 있습니다. SDK와 다운로드 경로에 대한 자세한 내용은 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.

이 섹션의 주제에는 Amazon GameLift에서 배포하기 전에 게임 클라이언트와 게임 서버에 필요한 Amazon GameLift 기능을 추가하는 방법에 대한 자세한 지침을 포함합니다. Amazon GameLift에서 게임을 시작하고 실행하는 완전한 로드맵은 Amazon GameLift 관리형 호스팅 로드맵 섹션을 참조하세요.

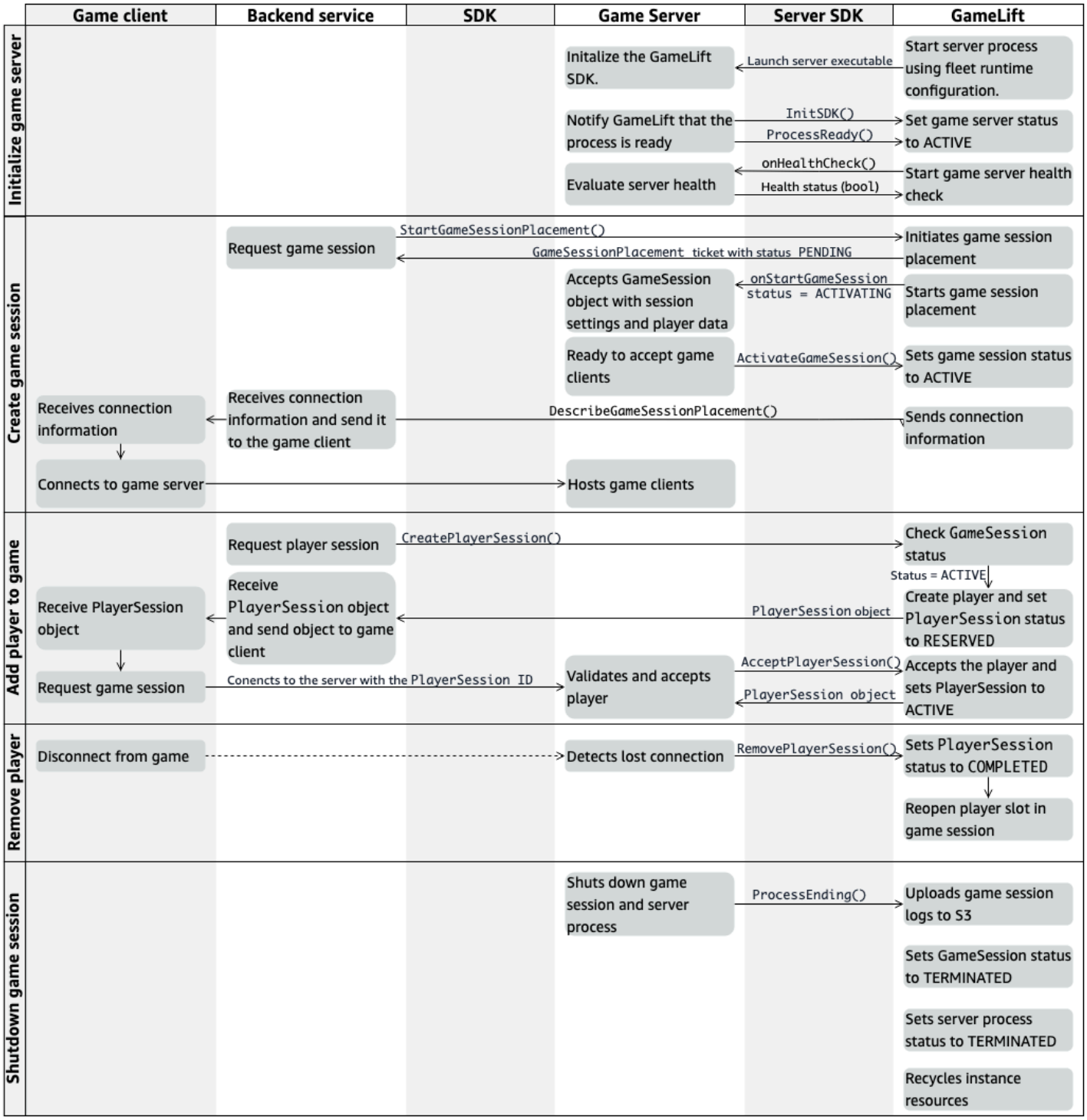
주제

- Amazon GameLift 및 게임 클라이언트 서버 상호 작용
- Amazon GameLift에 게임 서버 통합
- Amazon GameLift에 게임 클라이언트 통합
- 게임 엔진 및 Amazon GameLift
- Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트
- Amazon GameLift Local을 사용하여 통합 테스트

Amazon GameLift 및 게임 클라이언트 서버 상호 작용

이 주제에서는 게임 클라이언트, 백엔드 서비스, 게임 서버 및 Amazon GameLift 간의 상호 작용을 설명합니다.

다음 다이어그램은 게임 클라이언트, 백엔드 서비스, Amazon GameLift SDK, 관리형 EC2 게임 서버, Amazon GameLift Server SDK, Amazon GameLift 간의 상호 작용을 보여줍니다. 표시된 상호 작용에 대한 자세한 설명은 이 페이지의 다음 섹션을 참조하세요.



게임 서버 초기화

다음 단계에서는 게임 세션을 호스팅하기 위해 게임 서버를 준비할 때 발생하는 상호 작용을 설명합니다.

1. Amazon GameLift는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 실행 가능한 서버를 시작합니다.
2. 게임 서버는 다음을 호출합니다.

   a. `InitSDK()` 서버 SDK를 초기화합니다.

   b. `ProcessReady()` 게임 세션 준비, 연결 정보 및 게임 세션 로그 파일의 위치를 통신합니다.

   그런 다음 서버 프로세스는 Amazon GameLift의 콜백을 기다립니다.
3. Amazon GameLift는 게임 세션 배치가 가능하도록 서버 프로세스의 상태를 `ACTIVE`로 업데이트 합니다.
4. Amazon GameLift는 `onHealthCheck` 콜백 호출을 시작하고 서버 프로세스가 활성화되어 있는 동안 주기적으로 콜백을 계속 호출합니다. 서버 프로세스는 1분 내에 정상 또는 비정상을 보고할 수 있습니다.

게임 세션을 생성

게임 서버를 초기화한 후 플레이어를 호스팅할 게임 세션을 생성할 때 다음과 같은 상호 작용이 발생합니다.

1. 백엔드 서비스는 SDK 작업 `StartGameSessionPlacement()`를 호출합니다.
2. Amazon GameLift는 `PENDING` 상태가 포함된 `GameSessionPlacement` 새 티켓을 생성하여 백엔드 서비스에 반환합니다.
3. 백엔드 서비스는 대기열에서 배치 티켓 상태를 가져옵니다. 자세한 내용은 게임 세션 배치의 이벤트 알림 설정 섹션을 참조하세요.
4. Amazon GameLift는 적절한 플릿을 선택하고 `0` 게임 세션이 있는 플릿에서 활성 서버 프로세스를 검색하여 게임 세션 배치를 시작합니다. Amazon GameLift가 서버 프로세스를 찾으면 Amazon GameLift는 다음을 수행합니다.

   a. 게임 세션 설정과 배치 요청의 플레이어 데이터(`ACTIVATING` 상태 포함)를 사용하여 `GameSession` 객체를 생성합니다.

   b. 서버 프로세스의 `onStartGameSession` 콜백을 호출합니다. Amazon GameLift는 서버 프로세스가 게임 세션을 설정할 수 있음을 나타내는 정보를 `GameSession` 객체에 전달합니다.

   c. 서버 프로세스의 게임 세션 수를 1로 변경합니다.
5. 서버 프로세스가 `onStartGameSession` 콜백 함수를 실행합니다. 서버 프로세스가 플레이어 연결을 수락할 준비가 되면 `ActivateGameSession()`을 호출하고 플레이어 연결을 기다립니다.

6. Amazon GameLift는 서버 프로세스에 대한 연결 정보로 GameSession 객체를 업데이트합니다. (이 정보에는 ProcessReady()로 보고된 포트 설정이 포함됩니다.) 또한 Amazon GameLift는 상태를 ACTIVE로 변경합니다.
7. 백엔드 서비스는 업데이트된 티켓 상태를 감지하기 위해 DescribeGameSessionPlacement()를 호출합니다. 그런 다음 백엔드 서비스는 연결 정보를 사용하여 게임 클라이언트를 서버 프로세스에 연결하고 게임 세션에 참여합니다.

게임에 플레이어 추가

이 시퀀스는 기존 게임 세션에 플레이어를 추가하는 프로세스를 설명합니다. 플레이어 세션은 게임 세션 배치 요청의 일부로 요청할 수도 있습니다.

1. 백엔드 서비스는 게임 세션 ID를 사용하여 클라이언트 API 작업 CreatePlayerSession()을 호출합니다.
2. Amazon GameLift가 게임 세션 상태(ACTIVE여야 함)를 확인하고 게임 세션에서 열린 플레이어 슬롯을 찾습니다. 슬롯이 발견되면 Amazon GameLift는 다음을 수행합니다.
   a. 새로운 PlayerSession 객체를 만들고 상태를 RESERVED로 설정합니다.
   b. PlayerSession 객체를 사용하여 백엔드 서비스 요청에 응답합니다.
3. 백엔드 서비스는 플레이어 세션 ID로 서버 프로세스에 직접 게임 클라이언트를 연결합니다.
4. 서버가 서버 API 작업 AcceptPlayerSession()을 호출하여 플레이어 세션 ID를 확인합니다. 확인되면 Amazon GameLift가 PlayerSession 객체를 서버 프로세스에 전달합니다. 서버 프로세스가 연결을 수락하거나 거부합니다.
5. Amazon GameLift는 다음 중 하나를 수행합니다.
   a. 연결이 수락되면 Amazon GameLift는 PlayerSession 상태를 ACTIVE로 설정합니다.
   b. 백엔드 서버의 원래 CreatePlayerSession() 호출 후 60초 내에 응답이 수신되지 않으면, Amazon GameLift는 PlayerSession 상태를 TIMEDOUT으로 변경하고 게임 세션에서 플레이어 슬롯을 다시 엽니다.

플레이어 제거

새 플레이어가 참여할 공간을 만들기 위해 게임 세션에서 플레이어를 제거하면 다음과 같은 상호 작용이 발생합니다.

1. 플레이어가 게임 연결을 끊습니다.

2. 서버가 끊어진 연결을 찾아내고 서버 API 작업 `RemovePlayerSession()`을 호출합니다.
3. Amazon GameLift는 `PlayerSession` 상태를 `COMPLETED`로 변경하고 게임 세션에서 플레이어 슬롯을 다시 엽니다.

게임 세션 종료

이 상호 작용 시퀀스는 서버 프로세스가 현재 게임 세션을 종료할 때 발생합니다.

1. 서버가 게임 세션과 서버를 종료합니다.
2. 서버가 Amazon GameLift로 `ProcessEnding()`을 호출합니다.
3. Amazon GameLift는 다음을 수행합니다.
    a. 게임 세션 로그를 Amazon Simple Storage Service(S3)에 업로드합니다.
    b. `GameSession` 상태가 `TERMINATED`로 변경됩니다.
    c. 서버 프로세스 상태를 `TERMINATED`로 변경합니다.
    d. 인스턴스 리소스를 재활용합니다.

Amazon GameLift에 게임 서버 통합

사용자 지정 게임 서버를 Amazon GameLift 인스턴스에 배포하고 실행한 후에는 Amazon GameLift(및 잠재적으로 다른 리소스)와 상호 작용할 수 있어야 합니다. 이 섹션에서는 게임 서버 소프트웨어를 Amazon GameLift와 통합하는 방법에 대해 설명합니다.

> **Note**
> 이 지침에서는 AWS 계정를 만들었고 기존 게임 서버 프로젝트가 있다고 가정합니다.

이 섹션의 항목에서는 다음 통합 작업을 처리하는 방법에 대해 설명합니다.

- Amazon GameLift 및 게임 서버 간에 통신을 설정합니다.
- TLS 인증서를 생성하고 사용하여 게임 클라이언트와 게임 서버 간에 보안 연결을 설정합니다.
- 게임 서버 소프트웨어가 다른 AWS 리소스와 상호 작용할 수 있는 권한을 제공합니다.
- 게임 서버 프로세스가 실행 중인 플릿에 대한 정보를 얻을 수 있도록 허용합니다.

주제

- Amazon GameLift를 게임 서버에 추가
- 플릿에서 다른 AWS 리소스와 통신

Amazon GameLift를 게임 서버에 추가

사용자 지정 게임 서버가 Amazon GameLift와 통신해야 하는 이유는 각 게임 서버 프로세스가 Amazon GameLift에서 시작하는 이벤트에 응답할 수 있어야 하기 때문입니다. 또한 게임 서버는 Amazon GameLift에 서버 프로세스 상태 및 플레이어 연결에 대한 정보를 지속적으로 제공해야 합니다. 게임 서버, 백엔드 서비스, 게임 클라이언트, Amazon GameLift가 함께 작동하여 게임 호스팅을 관리하는 방법에 대한 자세한 내용은 Amazon GameLift 및 게임 클라이언트 서버 상호 작용 섹션을 참조하세요.

Amazon GameLift와 통합하도록 게임 서버를 준비시키려면 게임 서버 프로젝트에 Amazon GameLift Server SDK를 추가하고 이 주제에서 설명한 기능을 빌드합니다. Server SDK는 여러 언어로 제공됩니다. Amazon GameLift Server SDK에 대한 자세한 내용은 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.

Server SDK API 참조:

- C++용 Amazon GameLift Server SDK 5.x 참조
- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조
- Amazon GameLift Unreal Engine Server SDK 5.x 참조

서버 프로세스 초기화

Amazon GameLift와의 통신을 설정하고 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 보고하는 코드를 추가합니다. 이 코드는 Amazon GameLift 코드보다 먼저 실행되어야 합니다.

1. `InitSdk()`를 호출하여 Amazon GameLift API 클라이언트를 초기화합니다. Amazon GameLift Anywhere 컴퓨팅 리소스에서 서버 프로세스를 초기화하려면 다음 `ServerParameters`를 사용하여 `InitSdk()`를 호출합니다.

   - 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
   - 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
   - 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
   - Amazon GameLift Anywhere 컴퓨팅을 포함하는 GameLift 플릿의 ID입니다.
   - Amazon GameLift 작업(`GetComputeAuthToken`)을 통해 생성된 인증 토큰입니다.

> **Note**
>
> Amazon GameLift 관리형 Amazon EC2 인스턴스에서 게임 서버를 초기화하려면 기본 `InitSDK()`(C++) (C#) (Unreal) 생성자(파라미터 없음)를 사용하여 ServerParameters를 구성합니다. Amazon GameLift는 컴퓨팅 환경을 설정하고 자동으로 Amazon GameLift에 연결합니다.

2. Amazon GameLift에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 다음 정보와 함께 `ProcessReady()`(C++) (C#) (Unreal) 를 호출합니다. (서버 프로세스당 한 번만 `ProcessReady()`를 호출해야 한다는 점에 유의하세요.)

   - 서버 프로세스에서 사용하는 포트 번호입니다. 백엔드 서비스는 서버 프로세스에 연결하고 게임 세션에 참여할 수 있도록 게임 클라이언트에 포트 번호와 IP 주소를 제공합니다.
   - Amazon GameLift에서 보관하려는 파일(예: 게임 세션 로그)의 위치입니다. 서버 프로세스는 게임 세션 중에 이러한 파일을 생성합니다. 파일은 서버 프로세스가 실행 중인 인스턴스에 임시로 저장되며 인스턴스가 종료되면 손실됩니다. 목록에 있는 모든 파일은 Amazon GameLift에 업로드됩니다. Amazon GameLift 콘솔을 통하거나 Amazon GameLift API 작업 GetGameSessionLogUrl()을 호출하여 이러한 파일에 액세스할 수 있습니다.
   - Amazon GameLift가 서버 프로세스에 호출할 수 있는 콜백 함수의 이름입니다. 게임 서버는 이러한 함수를 구현해야 합니다. 자세한 내용은 (C++) (C#) (Unreal) 을 참조하세요.
     - (선택 사항) `onHealthCheck` - Amazon GameLift는 이 함수를 정기적으로 호출하여 서버에서 상태 보고서를 요청합니다.
     - `onStartGameSession` - Amazon GameLift는 클라이언트 요청 CreateGameSession()에 대한 응답으로 이 함수를 호출합니다.
     - `onProcessTerminate` - Amazon GameLift는 서버 프로세스를 강제로 중지하여 정상적으로 종료합니다.
     - (선택 사항) `onUpdateGameSession` - Amazon GameLift가 업데이트된 게임 세션 객체를 게임 서버에 전달하거나 매치 채우기 요청에 대한 상태 업데이트를 제공합니다. FlexMatch 채우기 기능은 이 콜백이 필요합니다.

   소유하거나 제어하는 AWS 리소스에 안전하게 액세스하도록 게임 서버를 설정할 수도 있습니다. 자세한 내용은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

(선택 사항) 서버 프로세스 상태 보고

코드를 게임 서버에 추가하여 콜백 함수 `onHealthCheck()`를 구현합니다. Amazon GameLift는 이 콜백 메서드를 주기적으로 호출하여 상태 지표를 수집합니다. 이 콜백 함수를 구현하려면 다음을 수행합니다.

- 서버 프로세스의 상태를 평가합니다. 예를 들어, 외부 종속성에 오류가 있는 경우 서버 프로세스의 상태를 비정상으로 보고할 수 있습니다.
- 상태 평가를 완료하고 60초 내에 콜백에 응답합니다. Amazon GameLift가 60초 내에 응답을 수신하지 못하면 자동으로 서버 프로세스를 비정상으로 간주합니다.
- 정상에 대해 true, 비정상에 대해 false 부울 값을 반환합니다.

상태 확인 콜백을 구현하지 않으면 Amazon GameLift는 서버가 응답하지 않는 한 서버 프로세스를 정상으로 간주합니다.

Amazon GameLift는 서버 프로세스 상태를 사용하여 비정상 프로세스를 종료하고 리소스를 정리합니다. 서버 프로세스가 계속 비정상으로 보고되거나 세 번 연속 상태 확인에 응답이 없으면 Amazon GameLift는 해당 프로세스를 종료하고 새 프로세스를 시작할 수 있습니다. Amazon GameLift는 플릿의 서버 프로세스 상태에 대한 지표를 수집합니다.

(선택 사항) TLS 인증서 받기

TLS 인증서 생성이 활성화된 플릿에서 서버 프로세스가 실행 중인 경우 TLS 인증서를 검색하여 게임 클라이언트와 보안 연결을 설정하고 클라이언트 서버 통신을 암호화할 수 있습니다. 인증서 복사본이 인스턴스에 저장됩니다. 파일 위치를 확인하려면 `GetComputeCertificate()`(C++) (C#) (Unreal)을 호출합니다.

게임 세션 시작

코드를 추가하여 콜백 함수 `onStartGameSession`을 구현합니다. Amazon GameLift는 이 콜백을 호출하여 서버에서 게임 세션을 시작합니다.

`onStartGameSession` 함수는 GameSession 객체를 입력 파라미터로 받아들입니다. 이 객체에는 최대 플레이어 수와 같은 주요 게임 세션 정보가 포함됩니다. 게임 데이터 및 플레이어 데이터도 포함될 수 있습니다. 함수 구현은 다음 작업을 수행해야 합니다.

- `GameSession` 속성을 기반으로 새 게임 세션을 생성하는 작업을 시작합니다. 최소한 게임 서버는 게임 클라이언트가 서버 프로세스에 연결할 때 참조하는 게임 세션 ID에 연결해야 합니다.

- 필요에 따라 게임 데이터와 플레이어 데이터를 처리합니다. 이 데이터는 GameSession 객체에 있습니다.
- 새 게임 세션이 플레이어를 받아들일 준비가 되면 Amazon GameLift에 알립니다. 서버 API 작업 ActivateGameSession()(C++) (C#) (Unreal) 을 호출합니다. 성공적인 호출에 대한 응답으로 Amazon GameLift가 게임 세션 상태를 ACTIVE로 변경합니다.

(선택 사항) 새 플레이어 확인

플레이어 세션 상태를 추적하는 경우 새 플레이어가 게임 서버에 연결될 때 해당 플레이어를 확인하는 코드를 추가합니다. Amazon GameLift는 현재 플레이어와 사용 가능한 게임 세션 슬롯을 추적합니다.

검증을 위해 게임 세션에 대한 액세스를 요청하는 게임 클라이언트는 플레이어 세션 ID를 포함해야 합니다. Amazon GameLift는 플레이어가 StartGameSessionPlacement() 또는 StartMatchmaking()을 사용하여 게임에 참여하도록 요청하면 이 ID를 자동으로 생성합니다. 그러면 플레이어 세션이 게임 세션에서 열린 슬롯을 예약합니다.

게임 서버 프로세스가 게임 클라이언트 연결 요청을 받으면 플레이어 세션 ID로 AcceptPlayerSession()(C++) (C#) (Unreal) 을 호출합니다. 이에 대한 응답으로 Amazon GameLift는 플레이어 세션 ID가 게임 세션에 예약된 열린 슬롯과 일치하는지 확인합니다. Amazon GameLift가 플레이어 세션 ID를 검증하면 서버 프로세스가 연결을 수락합니다. 그러면 플레이어가 게임 세션에 참여할 수 있습니다. Amazon GameLift가 플레이어 세션 ID를 검증하지 않으면 서버 프로세스가 연결을 거부합니다.

(선택 사항) 플레이어 세션 종료 보고

플레이어 세션 상태를 추적하는 경우 플레이어가 게임 세션을 종료할 때 Amazon GameLift에 알리는 코드를 추가합니다. 이 코드는 서버 프로세스가 끊어진 연결을 감지할 때마다 실행되어야 합니다. Amazon GameLift는 이 알림을 사용하여 게임 세션에서 현재 플레이어와 사용 가능한 슬롯을 추적합니다.

코드에서 끊긴 연결을 처리하려면 Server API 작업 RemovePlayerSession()(C++) (C#) (Unreal) 에 대한 호출을 해당 플레이어 세션 ID와 함께 추가합니다.

게임 세션 종료

코드를 서버 프로세스 종료 시퀀스에 추가하여 게임 세션이 종료될 때 Amazon GameLift에 알립니다. 호스팅 리소스를 재활용하고 새로 고치려면 게임 세션이 완료된 후 Amazon GameLift가 서버 프로세스를 종료합니다.

서버 프로세스 종료 코드를 시작할 때 서버 API 작업 `ProcessEnding()`(C++) (C#) (Unreal) 을 호출합니다. 이 호출은 서버 프로세스가 종료 중임을 Amazon GameLift에 알립니다. Amazon GameLift는 게임 세션 상태 및 서버 프로세스 상태를 `TERMINATED`로 변경합니다. `ProcessEnding()`을 호출한 후에는 프로세스를 안전하게 종료할 수 있습니다.

서버 프로세스 종료 알림에 응답

Amazon GameLift의 알림에 대한 응답으로 서버 프로세스를 종료하는 코드를 추가합니다. Amazon GameLift는 서버 프로세스가 지속적으로 비정상이라고 보고하거나 서버 프로세스가 실행 중인 인스턴스가 종료되는 경우 이 알림을 보냅니다. Amazon GameLift는 스케일 다운 이벤트의 일부로 또는 스팟 인스턴스 중단에 대한 응답으로 인스턴스를 중지할 수 있습니다.

종료 알림을 처리하려면 게임 서버 코드를 다음과 같이 변경합니다.

- 콜백 함수 `onProcessTerminate()`을 구현합니다. 이 함수는 게임 프로세스를 종료하는 코드를 호출해야 합니다. Amazon GameLift에서 이 작업을 호출하면 스팟 인스턴스 중단이 발생하여 2분 동안 알림을 보냅니다. 이 알림으로 서버 프로세스 시간을 제공하여 안전하게 플레이어 연결을 해제하고, 게임 상태 데이터를 보존하며, 기타 정리 작업을 수행할 수 있습니다.
- 게임 서버 종료 코드에서 서버 API 작업 `GetTerminationTime()`(C++) (C#) (Unreal) 을 호출합니다. Amazon GameLift가 서버 프로세스 중지 호출을 실행한 경우 `GetTerminationTime()`이 예상 종료 시간을 반환합니다.
- 게임 서버 종료 코드를 시작할 때 서버 API 작업 `ProcessEnding()`(C++) (C#) (Unreal) 을 호출합니다. 이 호출은 서버 프로세스가 종료되었음을 Amazon GameLift에 알리고 Amazon GameLift는 서버 프로세스 상태를 `TERMINATED`로 변경합니다. `ProcessEnding()`을 호출한 후에는 프로세스를 안전하게 종료할 수 있습니다.

플릿에서 다른 AWS 리소스와 통신

Amazon GameLift 플릿에 배포할 게임 서버 빌드를 생성할 때는 게임 빌드의 애플리케이션이 소유한 다른 AWS 리소스와 직접 안전하게 통신하기를 원할 수 있습니다. Amazon GameLift는 게임 호스팅 플릿을 관리하므로 Amazon GameLift에 이러한 리소스 및 서비스에 대한 제한된 액세스 권한을 부여해야 합니다.

몇 가지 예제 시나리오는 다음과 같습니다.

- Amazon CloudWatch 에이전트를 사용하여 관리형 EC2 플릿 및 Anywhere 플릿에서 지표, 로그 및 추적을 수집합니다.
- 인스턴스 로그 데이터를 Amazon CloudWatch Logs로 전송합니다.

- Amazon Simple Storage Service(S3) 버킷에 저장된 게임 파일을 가져옵니다.
- Amazon DynamoDB 데이터베이스 또는 다른 데이터 스토리지 서비스에 저장된 게임 데이터(예: 게임 모드 또는 인벤토리)를 읽고 씁니다.
- Amazon Simple Queue Service(Amazon SQS)를 사용하여 신호를 인스턴스에 직접 전송합니다.
- Amazon Elastic Compute Cloud(Amazon EC2)에서 배포되고 실행하는 사용자 지정 리소스에 액세스합니다.

Amazon GameLift는 액세스를 설정하기 위해 다음과 같은 방법을 지원합니다.

- IAM 역할로 AWS 리소스에 액세스
- VPC 피어링으로 AWS 리소스에 액세스

IAM 역할로 AWS 리소스에 액세스

IAM 역할을 사용하여 리소스에 액세스할 수 있는 사용자를 지정하고 해당 액세스에 제한을 설정합니다. 신뢰할 수 있는 당사자는 역할을 “수임”하고 리소스와 상호 작용할 수 있는 권한을 부여하는 임시 보안 자격 증명을 받을 수 있습니다. 당사자가 리소스와 관련된 API 요청을 할 때는 자격 증명을 포함해야 합니다.

IAM 역할로 액세스를 제어하도록 설정하려면 다음 작업을 수행합니다.

1. IAM 역할 생성
2. 애플리케이션을 수정하여 자격 증명 획득
3. IAM 역할을 사용하는 플릿과 연결합니다.

IAM 역할 생성

이 단계에서는 AWS 리소스에 대한 액세스를 제어할 수 있는 권한 세트와 Amazon GameLift에 역할 권한을 사용할 수 있는 권한을 부여하는 신뢰 정책을 포함하는 IAM 역할을 생성합니다.

IAM 역할을 생성하는 방법에 대한 지침은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요. 권한 정책을 생성할 때 애플리케이션이 처리해야 하는 특정 서비스, 리소스 및 작업을 선택합니다. 권한 범위를 최대한 제한하는 것이 가장 좋습니다.

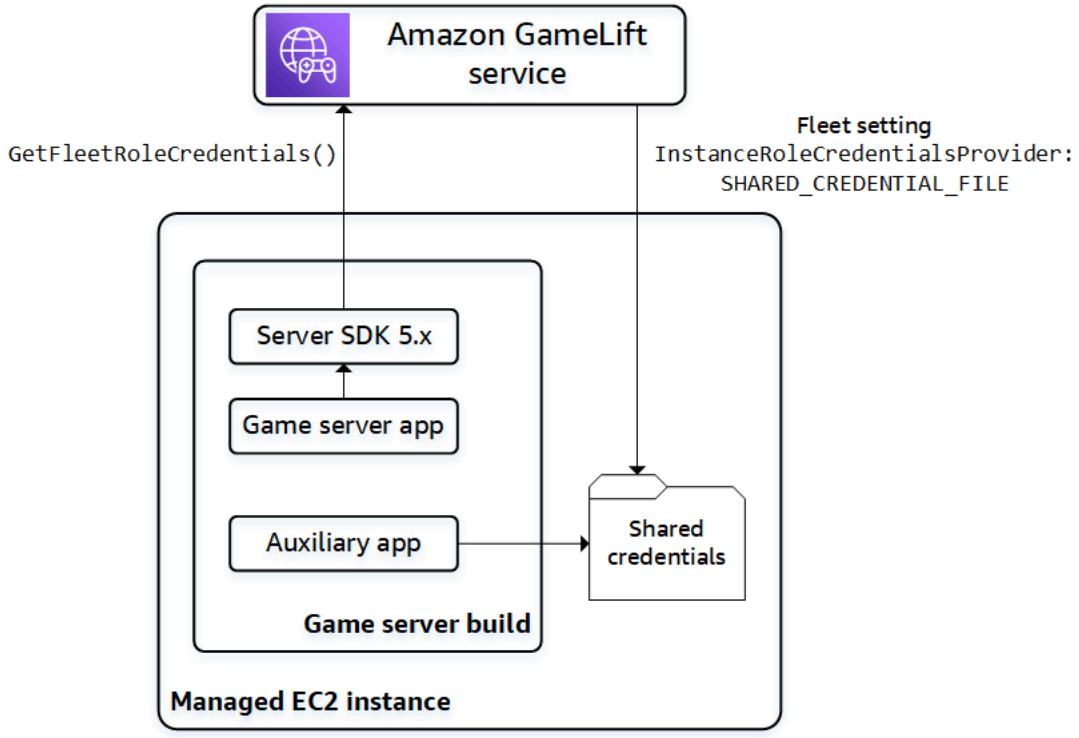
역할을 생성한 후 Amazon 리소스 이름(ARN)을 기록해 둡니다. 플릿 생성 중에는 역할 ARN이 필요합니다.

애플리케이션을 수정하여 자격 증명 획득

이 단계에서는 IAM 역할에 대한 보안 자격 증명을 획득하고 AWS 리소스와 상호 작용할 때 이를 사용하도록 애플리케이션을 구성합니다. 다음 표를 참조하여 (1) 애플리케이션 유형 및 (2) 게임에서 Amazon GameLift와 통신하는 데 사용하는 Server SDK 버전을 기반으로 애플리케이션을 수정하는 방법을 결정합니다.

	게임 서버 애플리케이션	기타 애플리케이션
Server SDK 버전 5.x 사용	게임 서버 코드에서 Server SDK 메서드 `GetFleetRoleCredentials()` 를 호출합니다.	코드를 애플리케이션에 추가하여 플릿 인스턴스의 공유 파일에서 자격 증명을 가져옵니다.
Server SDK 버전 4 또는 이전 버전 사용	역할 ARN을 사용하여 AWS Security Token Service(AWS STS) `AssumeRole` 을 호출합니다.	역할 ARN을 사용하여 AWS Security Token Service(AWS STS) `AssumeRole` 을 호출합니다.

Server SDK 5.x와 통합된 게임의 경우 이 다이어그램은 배포된 게임 빌드의 애플리케이션이 IAM 역할에 대한 자격 증명을 획득하는 방법을 보여줍니다.



`GetFleetRoleCredentials()` 호출(Server SDK 5.x)

Amazon GameLift Server SDK 5.x와 이미 통합되어 있어야 하는 게임 서버 코드에서 `GetFleetRoleCredentials`(C++) (C#) (Unreal)를 호출하여 임시 자격 증명 세트를 검색합니다. 자격 증명이 만료되면 `GetFleetRoleCredentials`를 다시 호출하여 자격 증명을 새로 고칠 수 있습니다.

공유 자격 증명 사용(Server SDK 5.x)

Server SDK 5.x를 사용하는 게임 서버 빌드와 함께 배포되는 비서버 애플리케이션의 경우 공유 파일에 저장된 자격 증명을 가져와 사용하는 코드를 추가합니다. Amazon GameLift는 각 플릿 인스턴스에 대한 자격 증명 프로필을 생성합니다. 자격 증명은 인스턴스의 모든 애플리케이션에서 사용할 수 있습니다. Amazon GameLift는 계속해서 임시 자격 증명을 새로 고칩니다.

플릿 생성 시 공유 자격 증명 파일을 생성하도록 플릿을 구성해야 합니다.

공유 자격 증명 파일을 사용해야 하는 각 애플리케이션에서 다음과 같이 파일 위치와 프로필 이름을 지정합니다.

Windows:

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

Linux:

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

예: Amazon GameLift 플릿 인스턴스에 대한 지표를 수집하도록 CloudWatch 에이전트를 설정합니다.

Amazon CloudWatch 에이전트를 사용하여 Amazon GameLift 플릿에서 지표, 로그 및 추적을 수집하려는 경우, 이 방법을 사용하여 에이전트가 사용자 계정으로 데이터를 내보내도록 승인합니다. 이 시나리오에서는 다음 단계를 수행합니다.

1. CloudWatch 에이전트 `config.json` 파일을 검색하거나 작성합니다.
2. 위에서 설명한 대로 에이전트가 자격 증명 파일 이름과 프로필 이름을 식별할 수 있도록 `common-config.toml` 파일을 업데이트합니다.
3. 게임 서버 빌드 설치 스크립트를 설정하여 CloudWatch 에이전트를 설치하고 시작합니다.

`AssumeRole()` 사용(Server SDK 4)

코드를 애플리케이션에 추가하여 IAM 역할을 맡고 AWS 리소스와 상호 작용하는 데 필요한 자격 증명을 얻습니다. Sever SDK 4 또는 이전 버전을 사용하는 Amazon GameLift 플릿 인스턴스에서 실행되는 모든 애플리케이션이 IAM 역할을 수임할 수 있습니다.

애플리케이션 코드에서 AWS 리소스에 액세스하기 전에 애플리케이션은 AWS Security Token Service(AWS STS) `AssumeRole` API 작업을 호출하고 역할 ARN을 지정해야 합니다. 이 작업은 AWS 리소스에 액세스할 수 있는 애플리케이션을 승인하는 임시 자격 증명 세트를 반환합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용을 참조하세요.

IAM 역할을 사용하는 플릿과 연결합니다.

IAM 역할을 생성하고 액세스 자격 증명을 가져와 사용하도록 게임 서버 빌드의 애플리케이션을 업데이트한 후, 플릿을 배포할 수 있습니다. 새 플릿을 구성할 때 다음 파라미터를 설정합니다.

- InstanceRoleArn - 이 파라미터를 IAM 역할의 ARN으로 설정합니다.
- InstanceRoleCredentialsProvider - Amazon GameLift에서 각 플릿 인스턴스에 대한 공유 자격 증명 파일을 생성하도록 요청하려면 이 파라미터를 `SHARED_CREDENTIAL_FILE`로 설정합니다.

플릿을 생성할 때 이러한 값을 설정해야 합니다. 이는 나중에 업데이트할 수 없습니다.

VPC 피어링으로 AWS 리소스에 액세스

Amazon Virtual Private Cloud(VPC) 피어링 기능을 사용하여 Amazon GameLift 인스턴스에서 실행되는 애플리케이션과 다른 AWS 리소스 간에 통신할 수 있습니다. VPC는 사용자가 정의하고 AWS 계정를 통해 관리하는 리소스 집합을 포함하는 가상 프라이빗 네트워크입니다. 각 Amazon GameLift 플릿에는 자체 VPC가 있습니다. VPC 피어링을 사용하여 플릿용 VPC와 다른 AWS 리소스용 VPC 사이의 직접 네트워크 연결을 설정할 수 있습니다.

Amazon GameLift는 게임 서버에 대한 VPC 피어링 연결 설정 프로세스를 간소화합니다. 피어링 요청을 처리하고, 라우팅 테이블을 업데이트하고, 필요에 따라 연결을 구성합니다. 게임 서버용 VPC 피어링을 설정하는 방법에 대한 자세한 지침은 Amazon GameLift용 VPC 피어링 섹션을 참조하세요.

Amazon GameLift에 게임 클라이언트 통합

이 섹션의 항목에서는 백엔드 서비스에 추가할 수 있는 관리형 Amazon GameLift 기능을 설명합니다. 백엔드 서비스는 다음 작업을 처리합니다.

- Amazon GameLift에서 활성 게임 세션과 관련된 정보를 요청합니다.

- 기존 게임 세션에 플레이어를 참가시킵니다.
- 새 게임 세션을 생성하고 해당 세션에 플레이어를 참가시킵니다.
- 기존 게임 세션에 대한 메타데이터를 변경합니다.

게임 클라이언트가 Amazon GameLift 및 Amazon GameLift에서 실행되는 게임 서버와 상호 작용하는 방식에 대한 자세한 내용은 Amazon GameLift 및 게임 클라이언트 서버 상호 작용 섹션을 참조하세요.

필수 조건

- AWS 계정.
- Amazon GameLift에 업로드된 게임 서버 빌드입니다.
- 게임 호스팅을 위한 플릿.

주제

- GameLift Amazon을 게임 클라이언트에 추가
- 플레이어 ID 생성

GameLift Amazon을 게임 클라이언트에 추가

GameLift Amazon을 게임 세션 정보가 필요한 게임 구성 요소에 통합하고, 새 게임 세션을 만들고, 게임에 플레이어를 추가합니다. 게임 아키텍처에 따라 이 기능은 플레이어 인증, 매치메이킹 또는 게임 세션 배치와 같은 작업을 처리하는 백엔드 서비스에 있습니다.

> **Note**
>
> Amazon GameLift 호스팅 게임의 매치메이킹을 설정하는 방법에 대한 자세한 내용은 Amazon GameLift FlexMatch 개발자 안내서를 참조하십시오.

백엔드 GameLift 서비스에 Amazon 설정

Amazon GameLift 클라이언트를 초기화하고 키 설정을 저장하는 코드를 추가합니다. 이 코드는 Amazon에 종속된 코드보다 먼저 실행되어야 GameLift 합니다.

1. 클라이언트 구성을 설정합니다. 기본 클라이언트 구성을 사용하거나 사용자 지정 클라이언트 구성 객체를 만듭니다. 자세한 내용은 AWS::Client::ClientConfiguration(C++) 또는 AmazonGameLiftConfig(C#) 을 참조하십시오.

클라이언트 구성은 GameLift Amazon에 문의할 때 사용할 대상 지역 및 엔드포인트를 지정합니다. 리전은 사용하도록 배포된 리소스(플릿, 대기열, 매치메이커) 세트를 식별합니다. 기본 클라이언트 구성은 위치를 미국 동부(버지니아 북부) 리전으로 설정합니다. 다른 리전을 사용하려면 사용자 지정 구성을 생성합니다.

2. Amazon GameLift 클라이언트를 초기화합니다. 기본 클라이언트 구성 또는 사용자 지정 클라이언트 구성과 함께 AwsGameLift::: GameLiftClient AmazonGameLiftClient() (C++) 또는 () (C#) 를 사용하십시오.
3. 각 플레이어에 대한 고유 식별자를 생성하는 메커니즘을 추가합니다. 자세한 정보는 플레이어 ID 생성을 참조하세요.
4. 다음 정보를 수집하고 저장합니다.

   - 대상 플릿 — 많은 Amazon GameLift API 요청은 플릿을 지정해야 합니다. 이렇게 하려면 대상 플릿을 가리키는 플릿 ID 또는 별칭 ID를 사용합니다. 백엔드 서비스를 업데이트하지 않고도 한 플릿에서 다른 플릿으로 플레이어를 전환할 수 있도록 플릿 별칭을 사용하는 것이 가장 좋습니다.
   - 대상 대기열 - 다중 플릿 대기열을 사용하여 새 게임 세션을 배치하는 게임의 경우 사용할 대기열 이름을 지정합니다.
   - AWS credentials — Amazon에 거는 모든 호출은 게임을 AWS 계정 호스팅하는 사람의 자격 증명을 GameLift 제공해야 합니다. 게임의 프로그래밍 방식 액세스 설정에 설명된 대로 플레이어 사용자를 생성하여 이러한 자격 증명을 획득합니다. 플레이어 사용자의 액세스를 관리하는 방법에 따라 다음을 수행합니다.
     - 역할을 사용하여 플레이어 사용자 권한을 관리하는 경우 Amazon GameLift API를 호출하기 전에 역할을 수임하는 코드를 추가하세요. 역할 수임 요청은 임시 보안 자격 증명 집합을 반환합니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할 (AWS API) 로 전환을 참조하십시오.
     - 장기 보안 자격 증명이 있는 경우 저장된 자격 증명을 찾아 사용하도록 코드를 구성합니다.AWS SDK 및 도구 참조 가이드에서 장기 보안 인증 정보를 사용한 인증을 참조하세요. 자격 증명 저장에 대한 자세한 내용은 (C++) 및 (.NET) 의AWS API 참조를 참조하십시오.
     - 임시 보안 자격 증명이 있는 경우 IAM 사용 설명서의 AWS SDK에서 임시 보안 자격 증명 사용에 설명된 대로 AWS Security Token Service (AWS STS)를 사용하여 자격 증명을 정기적으로 새로 고치는 코드를 추가합니다. 이전 자격 증명이 만료되기 전에 코드는 새 자격 증명을 요청해야 합니다.

게임 세션 가져오기

코드를 추가하여 사용 가능한 게임 세션을 검색하고 게임 세션 설정 및 메타데이터를 관리합니다.

활성 게임 세션 검색

특정 게임 세션, 모든 활성 세션 또는 검색 기준을 충족하는 세션에 대한 정보를 가져오는 SearchGameSessions데 사용합니다. 이 호출은 각 활성 게임 세션에 대해 검색 요청과 일치하는 GameSession객체를 반환합니다.

검색 기준을 사용하여 플레이어가 참여할 활성 게임 세션의 필터링된 목록을 가져옵니다. 예를 들어 다음과 같이 세션을 필터링할 수 있습니다.

- 가득 찬 게임 세션 제외: `CurrentPlayerSessionCount = MaximumPlayerSessionCount`
- 세션이 실행된 시간을 기준으로 게임 세션 선택: `CreationTime` 평가
- 사용자 지정 게임 속성을 기반으로 게임 세션 찾기: `gameSessionProperties.gameMode = "brawl"`

게임 세션 관리

다음 작업 중 하나를 사용하여 게임 세션 정보를 검색 또는 업데이트합니다.

- DescribeGameSessionDetails() — 게임 세션 정보와 함께 게임 세션의 보호 상태를 가져옵니다.
- UpdateGameSession() — 필요에 따라 게임 세션의 메타데이터와 설정을 변경합니다.
- GetGameSessionLogUrl— 저장된 게임 세션 로그에 액세스할 수 있습니다.

게임 세션 만들기

코드를 추가하여 배포된 플릿에서 새 게임 세션을 시작하고 플레이어가 사용할 수 있게 허용합니다. 게임을 여러 AWS 리전 지역에 배포할지 아니면 단일 지역에 배포하는지에 따라 두 가지 게임 세션 생성 옵션이 있습니다.

다중 위치 대기열에서 게임 세션 만들기

대기열에 새 게임 세션을 요청하는 StartGameSessionPlacement데 사용합니다. 이 작업을 사용하려면 대기열을 만듭니다. 이는 Amazon이 새 게임 세션을 GameLift 배치하는 위치를 결정합니다. 대기열 및 사용 방법에 대한 자세한 내용은 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.

게임 세션 배치를 생성할 때는 사용할 대기열의 이름, 게임 세션 이름, 최대 동시 플레이어 수 및 선택적 게임 속성 세트를 지정합니다. 필요하다면 자동으로 게임 세션에 참가시킬 플레이어 목록을 제공할 수도 있습니다. 관련 지역의 플레이어 지연 시간 데이터를 포함하면 GameLift Amazon은 이 정보를 사용하여 플레이어에게 이상적인 게임플레이 경험을 제공하는 플릿에 새 게임 세션을 배치합니다.

게임 세션 배치는 비동기식 프로세스입니다. 요청한 후에는 이 요청을 성공하거나 시간 초과되도록 할 수 있습니다. 를 사용하여 언제든지 요청을 취소할 수 StopGameSessionPlacement있습니다. 배치 요청 상태를 확인하려면 전화하십시오 DescribeGameSessionPlacement.

특정 플릿에서 게임 세션 생성

지정된 플릿에 새 세션을 생성하는 CreateGameSession데 사용합니다. 이 동기식 작업을 해당 플릿에 새 게임 세션을 호스팅하는 데 사용할 수 있는 리소스가 있는지 여부에 따라 성공 또는 실패가 결정됩니다. Amazon에서 새 게임 세션을 GameLift 생성하고 GameSession객체를 반환한 후 플레이어를 해당 세션에 참여시킬 수 있습니다.

이 작업을 사용하는 경우 플릿 ID 또는 별칭 ID, 세션 이름, 게임의 최대 동시 플레이어 수를 제공합니다. 선택적으로 게임 속성 세트를 포함할 수 있습니다. 게임 속성은 키-값 페어의 배열에 정의됩니다.

Amazon GameLift 리소스 보호 기능을 사용하여 한 명의 플레이어가 생성할 수 있는 게임 세션 수를 제한하는 경우 게임 세션 생성자의 플레이어 ID를 제공하십시오.

게임 세션에 플레이어 참여

코드를 추가하여 활성 게임 세션에서 플레이어 슬롯을 예약하고 게임 클라이언트를 게임 세션에 연결합니다.

1. 게임 세션에서 플레이어 슬롯을 예약합니다.

   플레이어 슬롯을 예약하려면 게임 세션에 대해 새 플레이어 세션을 만듭니다. 플레이어 세션에 대한 자세한 내용은 플레이어의 게임 연결 방법 섹션을 참조하세요.

   새 플레이어 세션을 만드는 방법은 두 가지가 있습니다.

   - 새 게임 세션에서 한 명 이상의 플레이어를 위한 슬롯을 예약하는 StartGameSessionPlacement데 사용합니다.
   - 게임 세션 ID를 사용하거나 게임 세션 ID가 CreatePlayerSessions있는 한 명 이상의 플레이어를 위해 플레이어 슬롯을 예약하십시오. CreatePlayerSession

   Amazon은 GameLift 먼저 게임 세션이 새 플레이어를 받아들이고 있으며 플레이어 슬롯이 사용 가능한지 확인합니다. 성공하면 Amazon은 플레이어를 위한 슬롯을 GameLift 예약하고, 새 플레이어 세션을 생성하고, PlayerSession객체를 반환합니다. 이 객체에는 게임 클라이언트가 게임 세션에 연결하는 데 필요한 DNS 이름, IP 주소 및 포트를 포함합니다.

플레이어 세션 요청은 각 플레이어의 고유한 ID를 포함해야 합니다. 자세한 정보는 플레이어 ID 생성을 참조하세요.

플레이어 세션이 사용자 지정 플레이어 데이터 집합을 포함할 수 있습니다. 이 데이터는 새로 생성된 플레이어 세션 객체에 저장되며, 이 객체는 DescribePlayerSessions() 를 호출하여 검색할 수 있습니다. GameLift 또한 Amazon은 플레이어가 게임 세션에 직접 연결할 때 이 객체를 게임 서버로 전달합니다. 여러 플레이어 세션을 요청하는 경우 각 플레이어에 대해 요청의 플레이어 ID와 매핑된 플레이어 데이터 문자열을 제공합니다.

2. 게임 세션에 연결

   게임 클라이언트에 코드를 추가하여 게임 세션의 연결 정보를 포함한 `PlayerSession` 객체를 검색합니다. 이 정보를 사용하여 서버에 직접 연결을 설정합니다.

   - 지정된 포트와 서버 프로세스에 할당된 DNS 이름 또는 IP 주소를 사용하여 연결할 수 있습니다.
   - 플릿에 TLS 인증서 생성이 활성화된 경우 DNS 이름과 포트를 사용하여 연결합니다.
   - 게임 서버가 들어오는 플레이어 연결을 검증하는 경우 플레이어 세션 ID를 참조하세요.

   연결 후 게임 클라이언트와 서버 프로세스는 Amazon의 개입 없이 직접 GameLift 통신합니다. 서버는 Amazon과 통신을 GameLift 유지하여 플레이어 연결 상태, 상태 등을 보고합니다. 게임 서버가 들어오는 플레이어를 검증하면 플레이어 세션 ID가 게임 세션의 예약된 슬롯과 일치하는지 확인하고 플레이어 연결을 수락하거나 거부합니다. 플레이어 연결이 끊기면 서버 프로세스는 끊어진 연결을 보고합니다.

게임 세션 속성 사용

게임 클라이언트는 게임 속성을 사용하여 게임 세션에 데이터를 전달할 수 있습니다. 게임 속성은 게임 서버가 추가하고, 읽고, 나열하고, 변경할 수 있는 키-값 쌍입니다. 새 게임 세션을 만들 때 또는 나중에 게임 세션이 활성 상태일 때 게임 속성을 전달할 수 있습니다. 게임 세션에는 최대 16개의 게임 속성이 포함될 수 있습니다. 게임 속성은 삭제할 수 없습니다.

예를 들어, 게임의 난이도는`Novice`,, `EasyIntermediate`, `Expert` 등입니다. 플레이어가 `Easy` 선택하고 게임을 시작합니다. 게임 클라이언트는 이전 섹션에 설명된 `CreateGameSession` 대로 `StartGameSessionPlacement` 또는 하나를 GameLift 사용하여 Amazon에 새 게임 세션을 요청합니다. 요청에서 클라이언트는 다음을 `{"Key": "Difficulty", "Value":"Easy"}` 전달합니다.

요청에 따라 Amazon은 지정된 게임 속성이 포함된 `GameSession` 객체를 GameLift 생성합니다. GameLift 그러면 Amazon은 사용 가능한 게임 서버에 새 게임 세션을 시작하라고 지시하고

GameSession 객체를 전달합니다. 게임 서버는 o를 사용하여 게임 세션을 시작합니다Difficulty. Easy

자세히 알아보기

- GameProperty 데이터 유형
- SearchGameSessions() 예제
- UpdateGameSession() GameProperties 매개변수

플레이어 ID 생성

Amazon GameLift는 플레이어 세션을 사용하여 게임 세션에 연결된 플레이어를 나타냅니다. Amazon GameLift는 플레이어가 Amazon GameLift와 통합된 게임 클라이언트를 사용하여 게임 세션에 연결할 때마다 플레이어 세션을 생성합니다. 플레이어가 게임에서 나가면 플레이어 세션이 종료됩니다. Amazon GameLift는 플레이어 세션을 재사용하지 않습니다.

다음 코드 샘플은 고유한 플레이어 ID를 무작위로 생성합니다.

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

플레이어 세션에 대한 자세한 내용은 게임 및 플레이어 세션에서 데이터 보기 섹션을 참조하세요.

게임 엔진 및 Amazon GameLift

C++ 또는 C# 라이브러리를 지원하는 가장 많이 사용되는 게임 엔진(O3DE, Unreal Engine 및 Unity 등)과 함께 관리형 Amazon GameLift 서비스를 사용할 수 있습니다. 게임에 필요한 버전을 빌드합니다. 빌드 지침 및 최소 요구 사항에 대해서는 각 버전의 README 파일을 참조하세요. 사용 가능한 Amazon GameLift SDK, 지원되는 개발 플랫폼, 운영 체제에 대한 자세한 내용은 게임 서버의 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.

이 주제에서 제공되는 엔진별 정보 외에도 다음 주제에 나오는 Amazon GameLift를 게임 서버, 클라이언트 및 서비스에 통합하는 작업과 관련된 추가 도움말을 참조하세요.

- Amazon GameLift 관리형 호스팅 로드맵 – Amazon GameLift를 게임에 성공적으로 통합하고 호스팅 리소스를 설정하기 위한 6단계 워크플로우.

- Amazon GameLift를 게임 서버에 추가 – Amazon GameLift를 게임 서버에 통합하는 방법에 대한 자세한 지침.
- GameLift Amazon을 게임 클라이언트에 추가 – 게임 세션을 생성하고 플레이어를 게임에 참가시키는 등 게임과의 통합에 대한 자세한 지침.

O3DE

게임 서버

C++용 Amazon GameLift Server SDK를 사용하여 Amazon GameLift에서 호스팅할 게임 서버를 준비합니다. 필요한 기능을 게임 서버에 통합하는 방법에 대한 도움을 얻으려면 Amazon GameLift를 게임 서버에 추가를 참조하세요.

게임 클라이언트와 서비스

게임 클라이언트 및/또는 게임 서비스가 사용할 수 있는 게임 세션 찾기 또는 새 게임 세션 생성 등 Amazon GameLift 서비스와 상호 작용하고 게임에 플레이어를 추가할 수 있습니다. 핵심 클라이언트 기능은 C++용 AWS SDK.에서 제공됩니다. Amazon GameLift를 O3DE 게임 프로젝트에 통합하려면 Amazon GameLift를 O3DE 게임 클라이언트 및 서버에 추가 및 GameLift Amazon을 게임 클라이언트에 추가 섹션을 참조하세요.

Unreal Engine

게임 서버

Unreal Engine용 Amazon GameLift Server SDK를 프로젝트에 추가하고 필요한 서버 기능을 구현하여 Amazon GameLift에서 호스팅할 게임 서버를 준비합니다. Unreal Engine 플러그인 설정 및 Amazon GameLift 코드 추가에 대한 도움말은 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

게임 클라이언트와 서비스

게임 클라이언트 및/또는 게임 서비스가 사용할 수 있는 게임 세션 찾기 또는 새 게임 세션 생성 등 Amazon GameLift 서비스와 상호 작용하고 게임에 플레이어를 추가할 수 있습니다. 핵심 클라이언트 기능은 C++용 AWS SDK.에서 제공됩니다. Amazon GameLift를 Unreal Engine 게임 프로젝트에 통합하려면 GameLift Amazon을 게임 클라이언트에 추가 섹션을 참조하세요.

Unity

게임 서버

C#용 Amazon GameLift Server SDK를 프로젝트에 추가하고 필요한 서버 기능을 구현하여 Amazon GameLift에서 호스팅할 게임 서버를 준비합니다. Unity를 통한 설정 및 Amazon GameLift 코드 추가에 대한 도움말은 Amazon GameLift를 Unity 프로젝트에 통합 섹션을 참조하세요.

게임 클라이언트와 서비스

게임 클라이언트 및/또는 게임 서비스가 사용할 수 있는 게임 세션 찾기 또는 새 게임 세션 생성 등 Amazon GameLift 서비스와 상호 작용하고 게임에 플레이어를 추가할 수 있습니다. 핵심 클라이언트 기능은 AWS SDK for .NET에서 제공됩니다. Amazon GameLift를 Unity 게임 프로젝트에 통합하려면 GameLift Amazon을 게임 클라이언트에 추가 섹션을 참조하세요.

기타 엔진

게임 서버 및 클라이언트에 사용할 수 있는 전체 Amazon GameLift SDK 목록은 the section called “개발 지원” 섹션을 참조하세요.

Amazon GameLift를 O3DE 게임 클라이언트 및 서버에 추가

O3DE, 오픈 소스, 크로스 플랫폼, 실시간 3D 엔진을 사용하여 게임 및 시뮬레이션을 포함한 고성능 대화형 환경을 만들 수 있습니다. O3DE 렌더러와 도구는 모듈식 프레임워크로 구성되어 있으며, 선호하는 개발 도구를 사용하여 수정하고 확장할 수 있습니다.

모듈식 프레임워크는 표준 인터페이스 및 애셋이 포함된 라이브러리가 포함된 Gem을 사용합니다. 자체 Gem을 선택하여 요구 사항에 따라 추가할 기능을 선택합니다.

Amazon GameLift Gem은 다음의 기능을 제공합니다.

Amazon GameLift 통합

> O3DE 네트워킹 계층을 확장하고 멀티플레이어 젬이 Amazon GameLift 전용 서버 솔루션과 함께 작동할 수 있도록 하는 프레임워크입니다. Gem은 Amazon GameLift Server SDK 및 AWS SDK 클라이언트(Amazon GameLift 서비스 자체라고 함)와의 통합을 제공합니다.

빌드 및 패키지 관리

> 리소스를 설정하고 업데이트하기 위한 전용 서버 빌드 및 AWS Cloud Development Kit (AWS CDK)(AWS CDK) 애플리케이션을 패키징하고 선택적으로 업로드하기 위한 지침입니다.

Amazon GameLift Gem 설정

이 섹션의 절차에 따라 O3DE에서 Amazon GameLift Gem을 설정합니다.

필수 조건

- Amazon GameLift를 위한 AWS 계정을 설정합니다. 자세한 내용은 설정 AWS 계정 섹션을 참조하세요.
- O3DE용 AWS 자격 증명을 설정합니다. 자세한 내용은 AWS 자격 증명 구성을 참조하세요.
- AWS CLI 및 AWS CDK를 설정합니다. 자세한 정보는 AWS Command Line Interface 및 AWS Cloud Development Kit (AWS CDK) 섹션을 참조하세요.

Amazon GameLift Gem 및 해당 종속성을 켭니다.

1. 프로젝트 관리자를 엽니다.
2. 프로젝트 아래의 메뉴를 열고 프로젝트 설정 편집...을 선택합니다.
3. 젬 구성을 선택합니다.
4. Amazon GameLift Gem 및 다음과 같은 종속 Gem을 켭니다.
   - AWS 코어 젬 - O3DE에서 AWS 서비스를 사용할 프레임워크를 제공합니다.
   - 멀티플레이어 젬 - 네트워킹 프레임워크를 확장하여 멀티플레이어 기능을 제공합니다.

Amazon GameLift Gem 정적 라이브러리 포함

1. 프로젝트 서버 대상에 대해 `BUILD_DEPENDENCIES`로 `Gem::AWSGameLift.Server.Static`을 포함합니다.

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

2. 프로젝트 서버 시스템 구성 요소에 필요한 `AWSGameLiftService`를 설정합니다.

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

3. (선택 사항) C++로 Amazon GameLift 서비스를 요청하려면 클라이언트 대상의 `BUILD_DEPENDENCIES`에 `Gem::AWSGameLift.Client.Static`을 포함합니다.

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

게임 및 전용 서버 통합

세션 관리 통합을 통해 게임 및 전용 게임 서버 내의 게임 세션을 관리합니다. FlexMatch를 지원하려면 FlexMatch 통합을 참조하세요.

GameLift Amazon을 언리얼 엔진 프로젝트에 통합

이 주제에서는 언리얼 엔진용 Amazon GameLift C++ 서버 SDK 플러그인을 설정하고 이를 게임 프로젝트에 통합하는 방법을 설명합니다.

추가 리소스:

- Unreal 다운로드 사이트용 Server SDK 플러그인
- Amazon GameLift Unreal Engine Server SDK 5.x 참조
- the section called “개발 지원”

필수 조건

진행하기 전에 다음과 같은 사전 조건을 검토해야 합니다.

필수 조건

- Unreal Engine을 실행할 수 있는 컴퓨터 Unreal Engine 요구 사항에 대한 자세한 내용은 Unreal Engine의 하드웨어 및 소프트웨어 사양 문서를 참조하세요.
- Microsoft Visual Studio 2019 이상 최신 버전
- CMake 버전 3.1 이상
- Python 버전 3.6 이상
- PATH에서 사용할 수 있는 Git 클라이언트
- Epic Games 계정 공식 Unreal Engine 웹사이트에서 계정을 등록합니다.
- 언리얼 GitHub 엔진 계정과 연결된 계정입니다. 자세한 내용은 언리얼 엔진 웹 사이트의 언리얼 엔진 소스 코드 액세스를 참조하십시오. GitHub

> **Note**
>
> Amazon은 GameLift 현재 다음 버전의 언리얼 엔진을 지원합니다.
>
> - 4.22
> - 4.23
> - 4.24
> - 4.25
> - 4.26
> - 4.27
> - 5.1.0
> - 5.1.1
> - 5.2
> - 5.3

소스에서 Unreal Engine 빌드

Epic 시작 관리자를 통해 다운로드한 표준 버전의 Unreal Engine Editor에서는 Unreal 클라이언트 애플리케이션 빌드만 허용합니다. Unreal 서버 애플리케이션을 빌드하려면 Unreal Engine Github 리포지토리를 사용하여 소스에서 Unreal Engine을 다운로드하고 빌드해야 합니다. 자세한 내용은 Unreal Engine 설명서 웹 사이트의 소스에서 Unreal Engine 빌드 자습서를 참조하세요.

> **Note**
>
> 아직 연결하지 않았다면, 언리얼 엔진 소스 코드 액세스의 GitHub 안내를 따라 계정을 에픽게임즈 GitHub 계정에 연결하세요.

Unreal Engine 소스를 개발 환경에 복제하려면

1. Unreal Engine 소스를 선택한 브랜치의 개발 환경에 복제합니다.

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

2. 게임 개발에 사용 중인 버전의 태그를 확인합니다. 예를 들어, 다음 예제는 Unreal Engine 버전 5.1.1을 확인합니다.

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

3. 로컬 리포지토리의 루트 폴더로 이동합니다. 루트 폴더에 있는 경우 다음 `Setup.bat` 파일을 실행합니다.
4. 루트 폴더에 있는 동안에도 `GenerateProjectFiles.bat` 파일을 실행합니다.
5. 이전 단계의 파일을 실행한 후, `UE5.sln` Unreal Engine 솔루션 파일이 생성됩니다. Visual Studio를 열고 Visual Studio 편집기에서 `UE5.sln` 파일을 엽니다.
6. Visual Studio에서 보기 메뉴를 열고 Solution Explorer 옵션을 선택합니다. 그러면 Unreal 프로젝트 노드의 컨텍스트 메뉴가 열립니다. Solution Explorer 창에서 `UE5.sln` 파일(UE5으로 나열 가능)을 마우스 오른쪽 버튼으로 클릭한 다음 빌드를 선택하여 Development Editor Win64 대상으로 Unreal 프로젝트를 빌드합니다.

   > **Note**
   >
   > 빌드를 완료하는 데 1시간이 넘게 걸릴 수 있습니다.

빌드가 완료되면 Unreal Development Editor를 열고 프로젝트를 만들거나 가져올 준비가 된 것입니다.

플러그인에 대한 Unreal 프로젝트 구성

다음 단계에 따라 게임 GameLift 서버 프로젝트에 사용할 언리얼 엔진용 Amazon 서버 SDK 플러그인을 준비하세요.

플러그인에 대한 프로젝트를 구성하려면

1. Visual Studio를 연 상태에서 Solution Explorer 창으로 이동한 다음 UE5 파일을 선택하여 Unreal 프로젝트의 컨텍스트 메뉴를 엽니다. 컨텍스트 메뉴에서 시작 프로젝트로 설정 옵션을 선택합니다.
2. Visual Studio 창 상단에서 디버깅 시작(녹색 화살표)을 선택합니다.

   이 작업을 수행하면 소스로 빌드된 새 Unreal Editor 인스턴스가 시작됩니다. Unreal Editor 사용에 대한 자세한 내용은 Unreal Engine 설명서 웹 사이트의 Unreal Editor 인터페이스를 참조하세요.
3. Unreal Editor에는 Unreal 프로젝트와 게임 프로젝트가 포함된 다른 Visual Studio 창이 열리므로, 열린 Visual Studio 창을 닫습니다.
4. Unreal Editor에서 다음 중 하나를 수행합니다.
   - GameLiftAmazon과 통합하려는 기존 언리얼 프로젝트를 선택합니다.
   - 새 프로젝트를 생성합니다 언리얼용 Amazon GameLift 플러그인을 실험해 보려면 언리얼 엔진의 3인칭 템플릿을 사용해 보세요. 이 템플릿에 대한 자세한 내용은 Unreal Engine 설명서 웹 사이트의 3인칭 템플릿을 참조하세요.

     또는 다음 설정으로 새 프로젝트를 구성합니다.

     - C++
     - 스타터 콘텐츠 포함
     - 데스크톱
     - 프로젝트 이름 이 주제의 예제에서는 프로젝트 이름을 `GameLiftUnrealApp`이라고 지정했습니다.
5. Visual Studio의 Solution Explorer에서 Unreal 프로젝트가 있는 위치로 이동합니다. Unreal `Source` 폴더에서 *`Your-application-name`*`.Target.cs`로 이름이 지정된 파일을 찾습니다.

   예를 들면 `GameLiftUnrealApp.Target.cs`입니다.
6. 이 파일을 복사하여 사본 이름을 *`Your-application-name`*`Server.Target.cs`로 지정합니다.

7. 새 파일을 열고 다음과 같이 변경합니다.

   - `class` 및 `constructor`를 변경하여 파일 이름과 일치시킵니다.
   - `Type`을 `TargetType.Game`에서 `TargetType.Server`로 변경합니다.
   - 최종 파일은 다음 예제와 같습니다.

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

이제 프로젝트가 Amazon GameLift 서버 SDK 플러그인을 허용하도록 구성되었습니다.

다음 작업은 Unreal용 C++ Server SDK 라이브러리를 빌드하여 프로젝트로 가져올 수 있도록 하는 것입니다.

Unreal용 C++ Server SDK 라이브러리를 빌드하려면

1. 언리얼용 Amazon GameLift C++ 서버 SDK 플러그인을 다운로드하세요.

   > **Note**
   >
   > SDK를 기본 다운로드 디렉터리에 넣으면 경로가 260자 제한을 초과하여 빌드에 실패할 수 있습니다. 예: `C:\Users\Administrator\Downloads\GameLift-SDK-Release-06_15_2023\GameLift-Cpp-ServerSDK-5.0.4`
   > 예를 들어 `C:\GameLift-Cpp-ServerSDK-5.0.4`의 경우 SDK를 다른 디렉터리로 이동하는 것이 좋습니다.

2. OpenSSL을 다운로드하고 설치합니다. OpenSSL을 다운로드하는 방법에 대한 자세한 내용은 Github OpenSSL 빌드 및 설치 설명서를 참조하세요.

   자세한 내용은 Windows 플랫폼에 대한 정보 설명서를 참조하세요.

> **Note**
>
> Amazon GameLift 서버 SDK를 빌드하는 데 사용하는 OpenSSL 버전은 언리얼에서 게임 서버를 패키징하는 데 사용하는 OpenSSL 버전과 일치해야 합니다. 언리얼 설치 디렉토리에서 버전 정보를 찾을 수 있습니다. `...Engine\Source\ThirdParty\OpenSSL`

3. 라이브러리를 다운로드한 후 Unreal Engine용 C++ Server SDK 라이브러리를 빌드합니다.

   다운로드한 SDK의 `GameLift-Cpp-ServerSDK-<version>` 디렉터리에서 `-DBUILD_FOR_UNREAL=1` 파라미터를 사용하여 컴파일하고 Server SDK를 빌드합니다. 다음 예제에서는 `cmake`를 사용하여 컴파일하는 방법을 보여줍니다.

   터미널에서 다음 명령을 실행합니다.

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

   Windows 빌드는 `out\gamelift-server-sdk\Release` 폴더에 다음과 같은 바이너리 파일을 생성합니다.

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

   두 라이브러리 파일을 Amazon GameLift Unreal Engine 플러그인 패키지의 `ThirdParty\GameLiftServerSDK\Win64` 폴더에 복사합니다.

다음 절차를 사용하여 Amazon GameLift 플러그인을 예제 프로젝트로 가져올 수 있습니다.

Amazon GameLift 플러그인 가져오기

1. 이전 절차에서 플러그인에서 추출한 `GameLiftServerSDK` 폴더를 찾습니다.
2. 게임 프로젝트 루트 `Plugins` 폴더에서 찾으세요. (폴더가 없는 경우 해당 폴더에서 생성하세요.)
3. `GameLiftServerSDK`폴더를 에 복사합니다`Plugins`.

   이렇게 하면 언리얼 프로젝트에서 플러그인을 볼 수 있게 됩니다.

4. Amazon GameLift 서버 SDK 플러그인을 게임 `.uproject` 파일에 추가합니다.

   이 예제에서는 앱은 `GameLiftUnrealApp`이라고 하므로 파일은 `GameLiftUnrealApp.uproject`입니다.

5. `.uproject` 파일을 편집하여 게임 프로젝트에 플러그인을 추가합니다.

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

6. 게임이 플러그인에 종속되는지 확인하십시오. ModuleRules `.Build.cs`파일을 열고 Amazon GameLiftServer SDK 종속성을 추가합니다. 이 파일은 *`Your-application-name`*`/Source//`*`Your-application-name/`*에 있습니다.

   예를 들어 자습서 파일 경로는 `../GameLiftUnrealApp/Source/GameLiftUnrealApp/GameLiftUnrealApp.Build.cs`입니다.

7. `PublicDependencyModuleNames`의 목록 끝에 `"GameLiftServerSDK"`를 추가합니다.

```
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;
    }
  }
```

이제 플러그인이 애플리케이션에서 작동할 것입니다. Amazon GameLift 기능을 게임에 통합하려면 다음 섹션을 계속 진행하십시오.

언리얼 GameLift 프로젝트에 Amazon 서버 코드 추가

언리얼 엔진 환경을 구성하고 설정했으며, 이제 게임 서버를 GameLift Amazon과 통합할 수 있습니다. 이 주제에 제시된 코드는 Amazon GameLift 서비스를 호출하는 데 필요합니다. 또한 Amazon

GameLift 서비스의 요청에 응답하는 콜백 함수 세트를 구현합니다. 각 함수와 코드가 수행하는 작업에 대한 자세한 내용은 서버 프로세스 초기화를 참조하세요. 이 코드에 사용된 SDK 작업 및 데이터 유형에 대한 자세한 내용은 Unreal Engine용 Amazon GameLift Server SDK 참조 섹션을 참조하세요.

Amazon에서 게임 서버를 GameLift 초기화하려면 다음 절차를 사용하십시오.

> **Note**
>
> 다음 섹션에 제공된 Amazon GameLift 전용 코드는 `WITH_GAMELIFT` 전처리기 플래그 사용에 따라 달라집니다. 이 플래그는 다음 두 조건이 모두 충족되는 경우에만 유효합니다.
>
> - `Target.Type == TargetRules.TargetType.Server`
> - 플러그인이 Amazon GameLift 서버 SDK 바이너리를 찾았습니다.
>
> 이렇게 하면 언리얼 서버 빌드만 GameLift Amazon의 백엔드 API를 호출할 수 있습니다. 또한 게임에서 생성할 수 있는 모든 다양한 Unreal 대상에 대해 제대로 실행되는 코드를 작성할 수 있습니다.

게임 서버를 Amazon과 통합 GameLift

1. Visual Studio에서 애플리케이션용 `.sln` 파일을 엽니다. 이 예제의 경우 `GameLiftUnrealApp.sln` 파일은 루트 폴더에서 찾을 수 있습니다.
2. 솔루션이 열린 상태에서 애플리케이션의 *`Your-application-name`*`GameMode.h` 파일을 찾습니다. 예: `GameLiftUnrealAppGameMode.h`.
3. 헤더 파일을 다음 예제 코드에 맞게 변경합니다. 반드시 GameLiftUnrealApp ""를 자신의 애플리케이션 이름으로 바꾸십시오.

```
#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. 관련 소스 파일인 *Your-application-name*GameMode.cpp 파일을 엽니다. 예제: GameLiftUnrealAppGameMode.cpp. 다음 예제 코드에 맞게 코드를 변경합니다. 반드시 GameLiftUnrealApp ""를 자신의 애플리케이션 이름으로 바꾸십시오.

   이 샘플은 게임 서버에 Amazon 추가에 설명된 대로 GameLift Amazon과의 통합에 필요한 모든 요소를 추가하는 방법을 보여줍니다. GameLift 여기에는 다음이 포함됩니다.

   - Amazon GameLift API 클라이언트를 초기화하는 중입니다.
   - OnStartGameSessionOnProcessTerminate, 및 onHealthCheck 등 Amazon GameLift 서비스의 요청에 응답하는 콜백 함수를 구현합니다.
   - 지정된 포트로 ProcessReady () 를 호출하여 게임 세션을 GameLiftservice 호스팅할 준비가 되면 Amazon에 알립니다.

```
#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. Development Editor 및 개발 서버 대상 유형 모두를 위한 게임 프로젝트를 빌드합니다.

   > **Note**
   >
   > 솔루션을 다시 빌드할 필요는 없습니다. 대신 앱 이름과 일치하는 Games 폴더 아래에 프로젝트만 빌드합니다. 그렇지 않으면 Visual Studio에서 전체 UE5 프로젝트를 다시 빌드하므로 최대 한 시간이 걸릴 수 있습니다.

6. 두 빌드가 모두 완료되면 Visual Studio를 닫고 프로젝트 `.uproject` 파일을 열어 Unreal Editor에서 빌드를 엽니다.
7. Unreal Editor에서 게임의 서버 빌드를 패키징합니다. 대상을 선택하려면 플랫폼, Windows로 이동하여 ***Your-application-name Server#*** 선택합니다.
8. 서버 애플리케이션 빌드 프로세스를 시작하려면 플랫폼, Windows로 이동하고 패키지 프로젝트를 선택합니다. 빌드가 완료되면 실행 파일이 있어야 합니다. 이 예제의 경우 파일 이름은 `GameLiftUnrealAppServer.exe`입니다.
9. Unreal Editor에서 서버 애플리케이션을 빌드하면 두 개의 실행 파일이 생성됩니다. 하나는 게임 빌드 폴더의 루트에 있으며 실제 서버 실행 파일의 래퍼 역할을 합니다.

   서버 빌드로 Amazon GameLift 플릿을 생성할 때는 실제 서버 실행 파일을 런타임 구성 시작 경로로 전달하는 것이 좋습니다. 예를 들어 게임 빌드 폴더의 루트에는 `GameLiftFPS.exe` 파일이 있고 다른 파일은 `\GameLiftFPS\Binaries\Win64\GameLiftFPSServer.exe`위치에 있을 수 있습니다. 플릿을 생성할 때는 런타임 구성의 시작 경로로 `C:\GameLiftFPS\Binaries\Win64\GameLiftFPSServer.exe`를 사용하는 것이 좋습니다.
10. 게임 서버가 게임 클라이언트와 통신할 수 있도록 Amazon GameLift 플릿에서 필요한 UDP 포트를 열어야 합니다. 기본적으로 Unreal Engine은 포트 7777을 사용합니다. 자세한 내용은 UpdateFleetPortSettingsAmazon GameLift 서비스 API 참조 안내서를 참조하십시오.
11. 게임 빌드용 `install.bat` 파일을 생성합니다. 이 설치 스크립트는 게임 빌드가 Amazon GameLift 플릿에 배포될 때마다 실행됩니다. 다음은 예제 파일 `install.bat`입니다.

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

    일부 언리얼 엔진 버전의 경우 대신 `install.bat` 다음과 같아야 합니다.

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

> **Note**
>
> `<>PrereqSetup_x64.exe` 파일로의 파일 경로는 `Engine\Extras\Redist\en-us`입니다.

12. 이제 게임 빌드를 패키징하여 Amazon에 업로드할 수 GameLift 있습니다.

    게임 빌드와 함께 패키징하는 OpenSSL 버전은 게임 서버를 빌드할 때 게임 엔진이 사용한 버전과 일치해야 합니다. 게임 서버 빌드와 함께 올바른 OpenSSL 버전을 패키징해야 합니다. Windows OS의 경우 OpenSSL 형식은 `.dll`입니다.

    > **Note**
    >
    > 게임 서버 빌드에 OpenSSL DLL을 패키징하세요. 게임 서버를 빌드할 때 사용한 것과 동일한 버전의 OpenSSL을 패키징해야 합니다.
    >
    > - `libssl-1_1-x64.dll`
    >
    >   `libcrypto-1_1-x64.dll`
    >
    > zip 파일의 루트에 게임 서버 실행 파일과 함께 종속 항목을 패키징합니다. 예를 들어 `openssl-lib` dll은 `.exe` 파일과 같은 디렉터리에 있어야 합니다.

다음 단계

언리얼 엔진 환경을 구성하고 설정했으니 이제 GameLift Amazon을 게임에 통합할 수 있습니다.

GameLift Amazon을 게임에 추가하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon GameLift를 게임 서버에 추가
- Unreal Engine용 Amazon GameLift Server SDK 참조

게임 테스트에 대한 지침은 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트, 섹션을 참조하세요.

Amazon GameLift를 Unity 프로젝트에 통합

이 주제에서는 Amazon GameLift Unity용 C#Server SDK 플러그인을 설정하고 이를 게임 프로젝트에 통합하는 방법에 대해 설명합니다.

추가 리소스:

- Amazon GameLift Server SDK 다운로드 사이트
- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조
- the section called “개발 지원”

필수 조건

Amazon GameLift Unity용 C# Server SDK 플러그인을 사용하려면 다음 구성 요소가 필요합니다.

- 플러그인이 지원하는 개발 환경 및 Unity 편집기 버전(아마존을 통한 개발 지원 GameLift 참조). Unity 버전에 대한 자세한 내용은 Unity 문서의 Unity용 시스템 요구 사항을 참조하세요.
- Unity 패키지용 Amazon GameLift Server SDK 플러그인. 이 패키지에는 C#용 Server SDK 5+가 포함되어 있습니다. 패키지는 다음 사이트에서 다운로드할 수 있습니다. Amazon GameLift 시작하기.
- 타사의 범위를 지정하는 레지스트리 UnityNuGet입니다. 이 도구는 타사 DLL을 관리합니다. 자세한 내용은 UnityNuGet GitHub 리포지토리를 참조하세요.

UnityNuGet 설정

게임 프로젝트에 UnityNuGet을 설정하지 않은 경우 다음 단계에 따라 Unity 패키지 관리자를 사용하여 도구를 설치합니다. 또는 NuGet CLI를 사용하여 DLL을 수동으로 다운로드할 수 있습니다. 자세한 내용은 Amazon GameLift Unity용 C# Server SDK `README`를 참조하세요.

UnityNuGet을 게임 프로젝트에 통합하려면

1. Unity 편집기에서 프로젝트를 열고 기본 메뉴로 이동하여 편집, 프로젝트 설정을 선택합니다. 옵션에서 패키지 관리자 섹션을 선택하고 범위 지정 레지스트리 그룹을 엽니다.
2. + 버튼을 선택하고 UnityNuGet 범위 지정 레지스트리에 다음 값을 입력합니다.

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

3. Unity 2021 버전 사용자의 경우

UnityNuGet을 설정한 후 Unity 콘솔에 `Assembly Version Validation` 오류가 표시되는지 확인합니다. 이러한 오류는 NuGet 패키지의 이름이 엄격한 어셈블리에 대한 바인딩 리디렉션이 Unity 프로젝트 내 경로로 제대로 확인되지 않는 경우 발생합니다. 이 문제를 해결하려면 Unity의 어셈블리 버전 검증을 구성하세요.

a. Unity 편집기에서 기본 메뉴로 이동하여 편집, 프로젝트 설정을 선택하고 플레이어 섹션을 엽니다.

b. 어셈블리 버전 검증 옵션을 선택 취소합니다.

플러그인 설치

다음 절차를 사용하여 Amazon GameLift Unity용 C# Server SDK 플러그인을 설치하고 log4net 로깅을 구성합니다.

플러그인을 설치하려면

1. Unity 편집기에서 프로젝트를 열고 기본 메뉴로 이동하여 창, 패키지 관리자를 선택합니다.
2. + 버튼을 선택하여 새 패키지를 추가합니다. tarball에서 패키지 추가 옵션을 선택합니다.
3. 디스크의 패키지 선택에서 Amazon GameLift Unity용 C# 서버 SDK 플러그인 다운로드 파일을 찾아 Amazon GameLiftServer버 SDK `.tgz` 파일을 선택합니다. 열기를 선택하여 플러그인을 설치합니다.

Amazon GameLift Server SDK는 log4net 프레임워크를 사용하여 로그 메시지를 출력합니다. 기본적으로 서버 빌드의 터미널에 메시지를 출력하도록 구성되어 있지만 Unity에는 파일 로깅 지원을 추가하기 위한 구성이 필요합니다. Amazon GameLift Server SDK 패키지 내에서 제공된 샘플을 가져와서 프로젝트에 이러한 지원을 추가할 수 있습니다. 다음 절차를 사용하여 샘플을 추가하고 log4net을 구성합니다.

파일 출력을 위해 log4net을 구성하려면

1. Unity 편집기에서 프로젝트를 열고 기본 메뉴로 이동하여 창, 패키지 관리자를 선택합니다.
2. 드롭다운 메뉴에서 패키지: 프로젝트에서를 선택한 다음 패키지 목록에서 Amazon GameLift Server SDK를 선택합니다. 그러면 패키지 세부 정보가 열립니다.
3. 패키지 세부 정보에서 샘플 그룹 옵션을 선택하고 가져오기를 누릅니다.
4. `log4net.config` 파일과 함께 제공되는 `LoggingConfiguration.cs` 스크립트는 구성을 자동으로 실행하며, 구성은 이제 `Assets/Samples` 프로젝트 폴더에 설정됩니다.

> **Note**
>
> `log4net.config` 파일을 프로젝트의 다른 폴더로 이동해야 하는 경우 `LoggingConfiguration.cs` 스크립트에 있는 구성 파일의 파일 경로도 새 경로로 업데이트해야 합니다. 자세한 내용은 log4net 구성에 대한 log4net 설명서를 참조하세요.

자세한 지침과 테스트 지침에 대한 자세한 내용은 플러그인 다운로드에 있는 `README`를 참조하세요.

테스트를 위한 Amazon GameLift Anywhere 플릿 설정

개발 워크스테이션을 Amazon GameLift Anywhere 호스팅 플릿으로 설정하여 Amazon GameLift 통합을 반복적으로 테스트할 수 있습니다. 이 설정을 통해 워크스테이션에서 게임 서버 프로세스를 시작하고, Amazon GameLift에 플레이어 가입 또는 매치메이킹 요청을 보내 게임 세션을 시작하고, 클라이언트를 새 게임 세션에 연결할 수 있습니다. 자체 워크스테이션을 호스팅 서버로 설정하면 Amazon GameLift와의 게임 통합의 모든 측면을 모니터링할 수 있습니다.

워크스테이션 설정에 대한 지침은 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 섹션을 참조하여 다음 단계를 완료합니다.

1. 워크스테이션에 대한 사용자 지정 위치를 만듭니다.
2. 새 사용자 지정 위치를 사용하여 Amazon GameLift Anywhere 플릿을 생성합니다. 요청이 성공하면 플릿 ID가 반환됩니다. 나중에 필요하므로 이 값은 기록해 둡니다.
3. 워크스테이션을 새 Anywhere 플릿의 컴퓨팅으로 등록합니다. 고유한 컴퓨팅 이름을 제공하고 워크스테이션의 IP 주소를 지정합니다. 성공하면 이 요청은 WebSocket URL의 형태로 서비스 SDK 엔드포인트를 반환합니다. 나중에 필요하므로 이 값은 기록해 둡니다.
4. 워크스테이션 컴퓨팅을 위한 인증 토큰을 생성합니다. 이 단수명 인증에는 토큰과 만료 날짜가 포함됩니다. 게임 서버는 이를 사용하여 Amazon GameLift 서비스와의 통신을 인증합니다. 워크스테이션 컴퓨팅에 인증을 저장하여 실행 중인 게임 서버 프로세스가 액세스할 수 있도록 합니다.

Amazon GameLift 서버 코드를 Unity 프로젝트에 추가

게임 서버는 Amazon GameLift 서비스와 통신하여 지침을 받고 진행 상태를 보고합니다. 이를 수행하려면 Amazon GameLift Server SDK를 사용하는 게임 서버 코드를 추가합니다.

제공된 코드 예제는 기본적인 필수 통합 요소를 보여줍니다. `MonoBehavior`를 사용하여 Amazon GameLift를 사용한 간단한 게임 서버 초기화를 설명합니다. 이 예제에서는 테스트를 위해 게임 서버가

Amazon GameLift Anywhere 플릿에서 실행된다고 가정합니다. 여기에는 다음과 같은 코드가 포함됩니다.

- Amazon GameLift API 클라이언트를 초기화합니다. 이 샘플은 Anywhere 플릿 및 컴퓨팅용 서버 파라미터가 있는 `InitSDK()` 버전을 사용합니다. 이전 주제 테스트를 위한 Amazon GameLift Anywhere 플릿 설정에서 정의한 대로 WebSocket URL, 플릿 ID, 컴퓨팅 이름(호스트 ID) 및 인증 토큰을 사용합니다.
- `OnStartGameSession`, `OnProcessTerminate` 및 `onHealthCheck`를 포함한 Amazon GameLift 서비스의 요청에 응답하는 콜백 함수를 구현합니다.
- 지정된 포트와 함께 ProcessReady()를 호출하여 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift 서비스에 알립니다.

이 주제에 제시된 코드는 Amazon GameLift 서비스와의 통신을 설정합니다. 또한 다음의 요청에 응답하는 콜백 함수 집합을 구현합니다. 각 함수와 코드가 수행하는 작업에 대한 자세한 내용은 서버 프로세스 초기화를 참조하세요. 이 코드에 사용된 SDK 작업 및 데이터 유형에 대한 자세한 내용은 C#용 Amazon GameLift Server SDK 참조 섹션을 참조하세요.

이 샘플은 Amazon GameLift를 게임 서버에 추가에 설명된 대로 모든 필수 요소를 추가하는 방법을 보여줍니다. 여기에는 다음이 포함됩니다.

Amazon GameLift 기능 추가에 대한 자세한 내용은 다음 주제를 참조하세요.

- Amazon GameLift를 게임 서버에 추가
- C#용 Amazon GameLift Server SDK 참조

```
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);
        }
    }
}
```

추가 리소스

다음 리소스를 사용하여 게임 서버를 테스트하고 기능을 확장합니다.

- 개발 머신을 Amazon GameLift Anywhere 플릿으로 설정하고 로컬 테스트에 사용합니다. 사용자 지정 서버 통합 테스트를 참조하세요.
- 게임 서버를 구축하고 Amazon GameLift에 빌드를 업로드합니다. Amazon GameLift에 사용자 지정 서버 빌드 업로드를 참조하세요.
- Amazon GameLift 관리형 EC2 플릿에 게임 서버 빌드를 배포합니다. 새 Amazon GameLift 플릿 생성을 참조하세요.

Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트

Amazon GameLift Anywhere 플릿을 사용하여 Amazon GameLift와의 게임 통합을 반복적으로 구축하고 테스트할 수 있습니다. Amazon GameLift 서비스에 연결하여 자체 하드웨어를 Anywhere 플릿으로 설정한 다음 여기에 게임 서버를 설치하고 실행합니다. 테스트 앱을 사용하여 게임 세션 시작/중지, 플레이어 연결 추적, 매치메이킹 채우기 처리와 같은 시나리오를 실행할 수 있습니다. Anywhere 플릿을 사용하면 필요에 따라 게임 서버 빌드를 업데이트하고 호스팅 활동을 완벽하게 파악할 수 있습니다.

Amazon GameLift Server SDK 버전 5 이상과 통합된 게임을 사용하여 Amazon GameLift Anywhere 플릿을 사용할 수 있습니다.

주제

- 초기 개발
- 게임 서버의 반복

초기 개발

게임을 개발하고 Amazon GameLift Server SDK와 통합하고 있습니다. 통합을 테스트하려면 게임 서버 빌드의 각 새로운 반복을 Amazon GameLift에 업로드하고 플릿을 생성할 수 있습니다. 또는 개발용 랩톱과 함께 Anywhere 플릿을 사용하면 반복 개발 및 테스트를 더 효율적으로 수행할 수 있습니다.

Amazon GameLift 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Anywhere 플릿을 생성하고 랩톱에서 게임 세션을 시작하려면 다음 절차를 수행합니다.

Console

1. Amazon GameLift 콘솔을 엽니다.

2. 탐색 창의 호스팅에서 위치를 선택합니다.
3. 위치 생성을 선택합니다.
4. 위치 생성 대화 상자에서 다음 작업을 수행하세요.

   a. 위치 이름을 입력합니다. 그러면 Amazon GameLift가 Anywhere 플릿으로 게임을 실행하는 데 사용하는 컴퓨팅 리소스의 위치가 레이블로 표시됩니다. 사용자 지정 위치 이름은 custom-으로 시작해야 합니다.

   b. 생성을 선택합니다.
5. Anywhere 플릿을 생성하려면 다음 작업을 수행합니다.

   a. 탐색 창의 호스팅에서 플릿을 선택합니다.

   b. 플릿 페이지에서 플릿 생성을 선택합니다.

   c. 컴퓨팅 유형 선택 단계에서 Anywhere를 선택하고 다음을 선택합니다.

   d. 플릿 세부 정보 정의 단계에서 새 플릿을 정의합니다. 자세한 내용은 새 Amazon GameLift 플릿 생성 섹션을 참조하세요.

   e. 위치 선택 단계에서 생성한 사용자 지정 위치를 선택합니다.

   f. 나머지 플릿 생성 단계를 완료하여 Anywhere 플릿을 생성합니다.
6. 생성한 플릿에 컴퓨팅 리소스로 랩톱을 등록합니다. `register-compute` 명령(또는 `RegisterCompute` API 작업)을 사용합니다. 이전 단계에서 생성한 `fleet-id`를 포함하고 `compute-name` 및 랩톱의 `ip-address`를 추가합니다.

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

출력 예:

```
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. 게임 서버의 디버그 세션을 시작합니다.

   a. 생성한 플릿에서 랩톱의 인증 토큰을 가져옵니다. get-compute-auth-token 명령(또는 GetComputeAuthToken API 작업)을 사용합니다.

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

   출력 예:

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

   b. 게임 서버 실행 파일의 디버그 인스턴스를 실행합니다. 디버그 인스턴스를 실행하려면 게임 서버가 InitSDK()를 호출해야 합니다. 프로세스가 게임 세션을 호스팅할 준비가 되면 게임 서버가 ProcessReady()를 호출합니다.

8. 게임 세션을 생성하여 Amazon GameLift Anywhere와의 첫 통합을 테스트합니다. create-game-session 명령(또는 CreateGameSession API 작업)을 사용합니다. 플릿의 사용자 지정 위치를 지정합니다.

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

   출력 예:

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

```
}
```

Amazon GameLift는 등록된 서버 프로세스로 onStartGameSession() 메시지를 보냅니다. 메시지에는 게임 속성, 게임 세션 데이터, 매치메이커 데이터 및 게임 세션에 대한 추가 정보와 함께 이전 단계의 GameSession 객체가 포함됩니다.

9. 서버 프로세스가 onStartGameSession() 메시지에 ActivateGameSession()으로 응답하도록 게임 서버에 로직을 추가합니다. 이 작업을 통해 서버에서 게임 세션 생성 메시지를 수신하고 수락했음을 알리는 확인 메시지를 Amazon GameLift로 보냅니다. 자세한 내용은 Amazon GameLift Server SDK 참조 섹션을 참조하세요.

이제 게임 서버에서 사용자가 테스트하여 반복에 사용할 수 있는 게임 세션을 실행하고 있습니다. 게임 서버에서 반복하는 방법을 알아보려면 다음 섹션을 계속 진행합니다.

AWS CLI

1. create-location 명령(또는 CreateLocation API 작업)을 사용하여 사용자 지정 위치를 생성합니다. 사용자 지정 위치는 Amazon GameLift가 Anywhere 플릿으로 게임을 실행하는 데 사용하는 하드웨어의 위치를 레이블로 표시합니다.

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

출력 예:

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

2. create-fleet 명령(또는 CreateFleet API 작업)을 사용하여 사용자 지정 위치가 있는 Anywhere 플릿을 생성합니다. Amazon GameLift는 홈 리전과 사용자가 제공하는 사용자 지정 위치에 플릿을 생성합니다.

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

출력 예:

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

3. 생성한 플릿에 컴퓨팅 리소스로 랩톱을 등록합니다. register-compute 명령(또는 RegisterCompute API 작업)을 사용합니다. 이전 단계에서 생성한 fleet-id를 포함하고 compute-name 및 랩톱의 퍼블릭 ip-address를 추가합니다.

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

출력 예:

```
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. 게임 서버의 디버그 세션을 시작합니다.

   a. 생성한 플릿에서 랩톱의 인증 토큰을 가져옵니다. get-compute-auth-token 명령(또는 GetComputeAuthToken API 작업)을 사용합니다.

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

출력 예:

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

b. 게임 서버 실행 파일의 디버그 인스턴스를 실행합니다. 디버그 인스턴스를 실행하려면 게임 서버가 `InitSDK()`를 호출해야 합니다. 프로세스가 게임 세션을 호스팅할 준비가 되면 게임 서버가 `ProcessReady()`를 호출합니다.

5. 게임 세션을 생성하여 Amazon GameLift Anywhere와의 첫 통합을 테스트합니다. `create-game-session` 명령(또는 `CreateGameSession` API 작업)을 사용합니다.

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

출력 예:

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

Amazon GameLift는 등록된 서버 프로세스로 `onStartGameSession()` 메시지를 보냅니다. 메시지에는 게임 속성, 게임 세션 데이터, 매치메이커 데이터 및 게임 세션에 대한 추가 정보와 함께 이전 단계의 `GameSession` 객체가 포함됩니다.

6. 서버 프로세스가 `onStartGameSession()` 메시지에 `ActivateGameSession()`으로 응답하도록 게임 서버에 로직을 추가합니다. 이 작업을 통해 서버에서 게임 세션 생성 메시지를 수신하고 수락했음을 알리는 확인 메시지를 Amazon GameLift로 보냅니다. 자세한 내용은 Amazon GameLift Server SDK 참조 섹션을 참조하세요.

이제 게임 서버에서 사용자가 테스트하여 반복에 사용할 수 있는 게임 세션을 실행하고 있습니다. 게임 서버에서 반복하는 방법을 알아보려면 다음 섹션을 계속 진행합니다.

게임 서버의 반복

이 사용 사례에서는 게임 서버를 설정하고 테스트한 후 버그를 발견한 시나리오에 대해 생각해 봅니다. Amazon GameLift Anywhere를 사용하면 코드를 반복하여 작업할 수 있으므로 Amazon EC2 플릿을 사용하는 번거로운 설정을 피할 수 있습니다.

1. 가능하면 기존 GameSession을 정리합니다. 게임 서버가 충돌하거나 ProcessEnding()을 호출하지 않는 경우, Amazon GameLift는 게임 서버의 상태 확인 전송을 중단한 후 GameSession을 정리합니다.
2. 게임 서버의 코드를 변경하고 컴파일한 후 다음 테스트를 준비합니다.
3. 이전 Anywhere 플릿은 활성 상태이고 랩톱은 플릿의 컴퓨팅 리소스로 등록되어 있습니다. 다시 테스트를 시작하려면 새 디버그 인스턴스를 생성합니다.

   a. 생성한 플릿에서 랩톱의 인증 토큰을 검색합니다. get-compute-auth-token 명령(또는 GetComputeAuthToken API 작업)을 사용합니다.

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

   출력 예:

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

   b. 게임 서버 실행 파일의 디버그 인스턴스를 실행합니다. 디버그 인스턴스를 실행하려면 게임 서버가 InitSDK()를 호출해야 합니다. 프로세스가 게임 세션을 호스팅할 준비가 되면 게임 서버가 ProcessReady()를 호출합니다.
4. 이제 플릿에서 사용 가능한 서버 프로세스를 사용할 수 있습니다. 게임 세션을 만들고 다음 테스트를 수행합니다. create-game-session 명령(또는 CreateGameSession API 작업)을 사용합니다.

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

Amazon GameLift는 등록된 서버 프로세스로 onStartGameSession() 메시지를 보냅니다. 메시지에는 게임 속성, 게임 세션 데이터, 매치메이커 데이터 및 게임 세션에 대한 추가 정보와 함께 이전 단계의 GameSession 객체가 포함됩니다.

5. 서버 프로세스가 onStartGameSession() 메시지에 ActivateGameSession()으로 응답하도록 게임 서버에 로직을 추가합니다. 이 작업을 통해 서버에서 게임 세션 생성 메시지를 수신하고 수락했음을 알리는 확인 메시지를 Amazon GameLift로 보냅니다. 자세한 내용은 Amazon GameLift Server SDK 참조 섹션을 참조하세요.

게임 서버 테스트를 마친 후에도 플릿 및 게임 서버 관리에 Amazon GameLift를 계속 사용할 수 있습니다. 자세한 내용은 아마존 GameLift Anywhere 플릿 생성 섹션을 참조하세요.

Amazon GameLift Local을 사용하여 통합 테스트

> **Note**
>
> Amazon GameLift Server SDK 버전 4.x 또는 이전 버전을 사용하는 경우 이 테스트 절차를 사용합니다. Server SDK 패키지에는 호환 가능한 버전의 Amazon GameLift Local이 포함되어 있습니다. Server SDK 버전 5.x를 사용하는 경우 Amazon GameLift Anywhere 플릿을 사용한 로컬 테스트에 대해서는 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 섹션을 참조하세요.

Amazon GameLift Local을 사용하여 로컬 디바이스에서 제한된 버전의 관리형 Amazon GameLift 서비스를 실행하고 이를 기준으로 게임 통합을 테스트합니다. 이 도구는 게임 통합에서 반복적 개발을 수행할 때 유용합니다. 대안(각각의 새 빌드를 Amazon GameLift에 업로드하고 게임을 호스팅하도록 플릿을 구성)은 매번 30분 이상 걸릴 수 있습니다.

Amazon GameLift Local을 사용하여 다음을 확인할 수 있습니다.

- 게임 서버가 Server SDK와 올바르게 통합되며 Amazon GameLift 서비스와 적절히 통신하여 새 게임 세션을 시작하고 새 플레이어를 수락하며 상태를 보고합니다.

- 게임 클라이언트가 Amazon GameLift용 AWS SDK와 올바르게 통합되며 기존 게임 세션에 대한 정보를 검색하고 새 게임 세션을 시작하며 플레이어를 게임에 참가시키고 게임 세션에 연결할 수 있습니다.

Amazon GameLift Local은 독립형 버전의 관리형 Amazon GameLift 서비스를 시작하는 명령줄 도구입니다. Amazon GameLift Local은 서버 프로세스 초기화, 상태 확인, API 호출 및 응답의 실행 중인 이벤트 로그도 제공합니다. Amazon GameLift Local은 Amazon GameLift에 대한 AWS SDK 작업의 하위 집합을 인식합니다. AWS CLI 또는 게임 클라이언트에서 호출할 수 있습니다. 모든 API 작업은 Amazon GameLift 웹 서비스에서와 마찬가지로 로컬에서 수행됩니다.

각 서버 프로세스는 단일 게임 세션만 호스팅해야 합니다. 게임 세션은 Amazon GameLift Local에 연결하는 데 사용하는 실행 파일입니다. 게임 세션이 완료되면 `GameLiftServerSDK::ProcessEndning`을 호출한 다음 프로세스를 종료해야 합니다. Amazon GameLift Local을 사용하여 로컬에서 테스트하는 경우 여러 서버 프로세스를 시작할 수 있습니다. 각 프로세스는 Amazon GameLift Local에 연결됩니다. 그런 다음 각 서버 프로세스마다 하나의 게임 세션을 생성할 수 있습니다. 게임 세션이 종료되면 게임 서버 프로세스가 종료되어야 합니다. 그런 다음 다른 서버 프로세스를 수동으로 시작해야 합니다.

Amazon GameLift Local은 다음 API를 지원합니다.

- CreateGameSession
- CreatePlayerSession
- CreatePlayerSessions
- DescribeGameSessions
- DescribePlayerSessions

Amazon GameLift Local 설정

Amazon GameLift Local은 Server SDK 번들과 함께 `.jar` 실행 파일로 제공됩니다. 이는 Windows 또는 Linux에서 실행할 수 있으며 Amazon GameLift 지원 언어와 함께 사용할 수 있습니다.

Local을 실행하기 전에 다음이 설치되어 있어야 합니다.

- Amazon GameLift Server SDK 버전 3.1.5~4.x의 빌드입니다.
- Java 8

게임 서버 테스트

게임 서버만 테스트하려면 AWS CLI를 사용하여 Amazon GameLift Local 서비스에 대한 게임 클라이언트 호출을 시뮬레이션하면 됩니다. 이는 게임 서버가 다음과 같은 점에서 예상대로 작동하고 있는지 확인합니다.

- 게임 서버가 제대로 시작하고 Amazon GameLift Server SDK를 초기화합니다.
- 시작 프로세스의 일환으로 게임 서버가 게임 세션을 호스팅할 준비가 되었음을 Amazon GameLift에 알립니다.
- 실행 도중 게임 서버가 Amazon GameLift에 1분마다 상태를 전송합니다.
- 게임 서버가 새 게임 세션 시작 요청에 응답합니다.

1. Amazon GameLift Local을 시작합니다.

   명령 프롬프트 창을 열고 *GameLiftLocal.jar* 파일이 포함된 디렉터리로 이동하여 이 파일을 실행합니다. 기본적으로 Local은 포트 8080에서 게임 클라이언트의 요청을 수신 대기합니다. 다른 포트 번호를 지정하려면 다음 예제에 나온 것처럼 `-p` 파라미터를 사용합니다:

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

   Local이 시작되면 2개의 로컬 서버가 시작되었다는 로그가 표시됩니다. 이때 한 서버는 게임 서버를 수신 대기하고 다른 한 서버는 게임 클라이언트 또는 AWS CLI를 수신 대기하는 상태입니다. 로그는 게임 구성 요소와의 통신 등, 2개의 로컬 서버에서 실행되는 활동을 계속 보고합니다.

2. 게임 서버를 시작합니다.

   Amazon GameLift 통합 게임 서버를 로컬에서 시작합니다. 게임 서버의 엔드포인트를 변경할 필요는 없습니다.

   Local 명령 프롬프트 창에서 게임 서버가 Amazon GameLift Local 서비스에 연결되었다는 로그 메시지가 표시됩니다. 이는 게임 서버가 Amazon GameLift Server SDK를 성공적으로 초기화했음을 의미합니다(`InitSDK()` 사용). 게임 서버는 표시된 로그 경로를 사용하여 `ProcessReady()`를 호출했고, 게임 세션을 호스팅할 준비가 되었습니다. 게임 서버가 실행되는 동안 Amazon GameLift는 게임 서버의 각 상태 보고서를 로깅합니다. 다음 로그 메시지 예는 성공적으로 통합된 게임 서버를 보여 줍니다.

   ```
   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
```

예상 오류 및 경고 메시지는 다음과 같습니다.

- 오류: “ProcessReady가 pID: *<process ID>*인 프로세스를 찾지 못했습니다! InitSDK()가 호출됐습니까?”
- 경고: “pID(*<process ID>*)로 처리되는 프로세스에 대한 Process 상태가 이미 존재합니다! ProcessReady(...)가 두 번 이상 호출됩니까?”

3. AWS CLI를 시작합니다.

   게임 서버가 `ProcessReady()`를 성공적으로 호출하면 클라이언트 호출을 시작할 수 있습니다. 다른 명령 프롬프트 창을 열고 AWS CLI 도구를 시작합니다. AWS CLI는 기본적으로 Amazon GameLift 웹 서비스 엔드포인트를 사용합니다. 다음 요청 예제에 나온 것처럼 `--endpoint-url` 파라미터를 사용하여 모든 요청에서 Local 엔드포인트로 이를 재정의해야 합니다.

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

   AWS CLI 명령 프롬프트 창에서 `AWS gamelift` 명령을 호출하면 AWS CLI 명령 레퍼런스에 설명된 것과 같은 응답이 발생합니다.

4. 게임 세션을 생성합니다.

   AWS CLI를 사용하여 CreateGameSession() 요청을 제출합니다. 요청은 필요한 구문에 따라야 합니다. Local의 경우, `FleetId` 파라미터는 어떤 유효한 문자열(`^fleet-\S+`)로도 설정이 가능합니다.

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

Local 명령 프롬프트 창에서 Amazon GameLift Local이 게임 서버에 onStartGameSession 콜백을 전송했다는 로그 메시지가 표시됩니다. 게임 세션이 성공적으로 생성된 경우, 게임 서버는 ActivateGameSession을 호출하여 응답합니다.

```
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"
```

AWS CLI 창에서 Amazon GameLift는 게임 세션 ID를 포함한 게임 세션 객체를 사용하여 응답합니다. 새 게임 세션 상태가 Activating임에 유의하십시오. 게임 서버가 ActivateGameSession을 호출하면 상태가 Active로 변경됩니다. 변경된 상태를 보려면 AWS CLI를 사용하여 DescribeGameSessions()를 호출합니다.

```
{
    "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
    }
}
```

게임 서버 및 클라이언트 테스트

플레이어와 게임 연결 등 전체 게임 통합을 확인하기 위해 게임 서버와 클라이언트를 모두 로컬에서 실행할 수 있습니다. 이렇게 하면 게임 클라이언트에서 Amazon GameLift Local로의 프로그래밍 방식 호출을 테스트할 수 있습니다. 다음과 같은 작업을 확인할 수 있습니다.

- 게임 클라이언트가 게임 세션 생성, 기존 게임 세션에 대한 정보 검색, 플레이어 세션 생성 등을 위해 Amazon GameLift Local 서비스에 성공적으로 AWS SDK 요청을 수행합니다.
- 플레이어가 게임 세션에 참가하려 할 때 게임 서버가 올바로 플레이어를 확인합니다. 확인된 플레이어의 경우, 게임 서버는 플레이어 데이터(구현된 경우)를 검색할 수 있습니다.
- 플레이어가 게임에서 나갈 때 게임 서버는 끊긴 연결을 보고합니다.
- 게임 서버가 게임 세션 종료를 보고합니다.

1. Amazon GameLift Local을 시작합니다.

   명령 프롬프트 창을 열고 *GameLiftLocal.jar* 파일이 포함된 디렉터리로 이동하여 이 파일을 실행합니다. 기본적으로 Local은 포트 8080에서 게임 클라이언트의 요청을 수신 대기합니다. 다른 포트 번호를 지정하려면 다음 예제에 나온 것처럼 `-p` 파라미터를 사용합니다.

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

   Local이 시작되면 2개의 로컬 서버가 시작되었다는 로그가 표시됩니다. 이때 한 서버는 게임 서버를 수신 대기하고 다른 한 서버는 게임 클라이언트 또는 AWS CLI를 수신 대기하는 상태입니다.

2. 게임 서버를 시작합니다.

   Amazon GameLift 통합 게임 서버를 로컬에서 시작합니다. 메시지 로그에 대한 자세한 내용은 게임 서버 테스트 섹션을 참조하세요.

3. Local에 대해 게임 클라이언트를 구성하고 시작합니다.

Amazon GameLift Local 서비스와 함께 게임 클라이언트를 사용하려면 백엔드 GameLift 서비스에 Amazon 설정에 설명된 것처럼 게임 클라이언트의 설정을 다음과 같이 변경해야 합니다.

- `http://localhost:9080` 등 Local 엔드포인트를 가리키도록 `ClientConfiguration` 객체를 변경합니다.
- 대상 플릿 ID 값을 설정합니다. Local의 경우, 실제 플릿 ID는 필요하지 않습니다. 대상 플릿을 `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`와 같은 유효한 문자열(`^fleet-\S+`)로 설정하십시오.
- AWS 자격 증명을 설정합니다. Local의 경우, 실제 AWS 자격 증명이 필요 없습니다. 액세스 키와 비밀 키를 어떤 문자열로도 설정할 수 있습니다.

게임 클라이언트를 시작한 경우에는 Local 명령 프롬프트 창에서 게임 클라이언트가 `GameLiftClient`를 초기화했고 Amazon GameLift 서비스와 성공적으로 통신하고 있다는 로그 메시지가 표시되어야 합니다.

4. Amazon GameLift 서비스로 게임 클라이언트 호출을 테스트합니다.

   게임 클라이언트가 다음 API 호출 중 하나 또는 전부를 성공적으로 하고 있는지 확인합니다.

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

   Local 명령 프롬프트 창에서 `CreateGameSession()`에 대한 호출만 로그 메시지를 발생시킵니다. 로그 메시지는 Amazon GameLift Local이 게임 서버에게 게임 세션을 시작하라고 하며(`onStartGameSession` 콜백) 게임 서버의 호출이 있을 때 성공적인 `ActivateGameSession`을 받으면 표시됩니다. AWS CLI 창에서, 모든 API 호출은 설명된 것과 같은 응답 또는 오류 메시지를 발생시킵니다.

5. 게임 서버가 새 플레이어 연결을 검증하고 있는지 확인합니다.

   게임 세션과 플레이어 세션을 생성한 후 게임 세션과의 직접적인 연결을 설정합니다.

   Local 명령 프롬프트 창에서, 로그 메시지에는 게임 서버가 새 플레이어 연결을 검증하기 위해 `AcceptPlayerSession()` 요청을 전송했음이 표시되어야 합니다. AWS CLI를 사용하여

`DescribePlayerSessions()`를 호출하는 경우, 플레이어 세션 상태가 Reserved에서 Active로 변경되어야 합니다.

6. 게임 서버가 게임 및 플레이어 상태를 Amazon GameLift 서비스에 보고하고 있는지 확인합니다.

   Amazon GameLift가 플레이어 수요를 관리하고 지표를 올바르게 보고하려면 게임 서버가 다양한 상태를 Amazon GameLift에 다시 보고해야 합니다. Local이 다음 작업과 관련된 이벤트를 로깅하고 있는지 확인합니다. AWS CLI를 사용하여 상태 변경을 추적할 수도 있습니다.

   - 플레이어가 게임 세션에서 연결 해제 - 게임 서버가 `RemovePlayerSession()`을 호출한다는 Amazon GameLift Local 로그 메시지가 표시되어야 합니다. AWS CLI가 `DescribePlayerSessions()`를 호출하면 `Active`에서 `Completed`로 상태 변경이 이루어져야 합니다. 또한 `DescribeGameSessions()`를 호출하여 게임 세션의 현재 플레이어 수가 한 명 줄었는지 확인할 수 있습니다.
   - 게임 세션 종료 - 게임 서버가 `TerminateGameSession()`을 호출한다는 Amazon GameLift Local 로그 메시지가 표시되어야 합니다.

     > **Note**
     >
     > 이전 지침은 게임 세션을 종료할 때 `TerminateGameSession()`을 호출하는 것이었습니다. 이 메서드는 Amazon GameLift Server SDK v4.0.1에서 더 이상 사용되지 않습니다. 게임 세션 종료 섹션을 참조하세요.

   - 서버 프로세스가 종료됨 - 게임 서버가 `ProcessEnding()`을 호출한다는 Amazon GameLift Local 로그 메시지가 표시되어야 합니다. AWS CLI가 `DescribeGameSessions()`를 호출하면 `Active`에서 `Terminated`(또는 `Terminating`)로 상태 변경이 이루어져야 합니다.

Local을 사용한 변형

Amazon GameLift Local을 사용할 때는 다음 사항을 명심해야 합니다.

- Local은 Amazon GameLift 웹 서비스와 달리 서버 상태를 추적하거나 `onProcessTerminate` 콜백을 시작하지 않습니다. Local은 단순히 게임 서버의 상태 보고서 로깅을 중지합니다.
- AWS SDK에 대한 호출의 경우, 플릿 ID는 검증되지 않으며 파라미터 요구 사항(`^fleet-\S+`)을 충족하는 어떤 문자열 값도 될 수 있습니다.
- Local을 사용하여 생성된 게임 세션 ID는 다른 구조를 갖습니다. 여기서 보듯 여기에는 문자열 `local`이 포함됩니다.

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

Amazon GameLift Realtime 서버와 게임 통합

이 주제에서는 Realtime 서버 솔루션을 사용하는 관리형 Amazon GameLift에 대한 개요를 제공합니다. 이 개요에서는 이 솔루션이 게임에 적합한 시기와 Realtime 서버가 멀티플레이어 게임을 지원하는 방법에 대해 설명합니다.

게임을 시작하고 실행하는 완전한 로드맵에 대해서는 Amazon GameLift 관리형 호스팅 로드맵 섹션을 참조하세요.

> **Tip**
>
> Amazon GameLift 게임 서버 호스팅을 사용해 보려면 Amazon GameLift 시작하기 섹션을 참조하세요.

Realtime 서버란 무엇입니까?

Realtime 서버는 멀티플레이어 게임에 사용할 수 있도록 Amazon GameLift에서 제공되는 가볍고 바로 사용 가능한 게임 서버입니다. Realtime 서버는 사용자 지정 게임 서버의 개발, 테스트 및 배포 프로세스를 제거합니다. 이 솔루션을 사용하면 게임을 완료하는 데 필요한 시간과 노력을 최소화할 수 있습니다.

주요 기능

- 게임 클라이언트 및 서버 상호 작용을 위한 전체 네트워크 스택
- 핵심 게임 서버 기능
- 사용자 지정 가능한 서버 로직
- Realtime 구성 및 서버 로직에 대한 실시간 업데이트
- FlexMatch 매치메이킹
- 호스팅 리소스의 유연한 제어

플릿을 만들고 구성 스크립트를 제공하여 Realtime 서버를 설정합니다. Realtime 서버의 생성 및 게임 클라이언트를 준비하는 방법에 대해 자세한 내용은 Realtime 서버 준비 섹션을 참조하세요.

Realtime 서버가 게임 세션을 관리하는 방법

로직을 Realtime 스크립트로 빌드하여 게임 세션 관리에 대한 사용자 지정 로직을 추가할 수 있습니다. 코드를 작성하여 서버별 객체에 액세스하거나, 콜백을 사용하여 이벤트 중심 로직을 추가하고 또는 비이벤트 시나리오를 기반으로 로직을 추가할 수 있습니다.

Realtime 클라이언트와 서버가 상호 작용하는 방식

게임 세션 중에 게임 클라이언트는 백엔드 서비스를 통해 Realtime 서버에 메시지를 전송하여 상호 작용합니다. 그런 다음 백엔드 서비스는 게임 클라이언트 간에 메시지를 전달하여 활동, 게임 상태, 관련 게임 데이터를 교환합니다.

추가로, 게임 로직을 Realtime 스크립트에 추가하여 클라이언트와 서버가 상호 작용하는 방식을 사용자 지정할 수 있습니다. 사용자 지정 게임 로직을 사용하면 Realtime 서버는 이벤트 중심 응답을 시작하는 콜백을 구현할 수 있습니다.

통신 프로토콜

Realtime 서버와 연결된 게임 클라이언트는 안정적인 전송을 위한 TCP 연결과 빠른 전송을 위한 UDP 채널이라는 두 개의 채널을 통해 통신합니다. 메시지를 생성할 때 게임 클라이언트는 메시지의 특성에 따라 어떤 프로토콜을 사용할지를 선택합니다. 메시지 전송은 기본적으로 UDP로 설정됩니다. UDP 채널을 사용할 수 없는 경우 Amazon GameLift는 대체 수단으로 TCP를 사용하여 메시지를 보냅니다.

메시지 콘텐츠

메시지 콘텐츠는 필요한 작동 코드(opCode)와 페이로드(선택 사항)의 두 가지 요소로 구성됩니다. 메시지의 opCode는 특정 플레이어 활동 또는 게임 이벤트를 식별하고, 페이로드는 작업 코드에 관련된 추가 데이터를 제공합니다. 이 두 요소는 모두 개발자가 정의합니다. 게임 클라이언트는 수신하는 메시지의 opCodes를 기반으로 작동합니다.

플레이어 그룹

Realtime 서버는 플레이어 그룹을 관리하기 위한 기능을 제공합니다. 기본적으로 Amazon GameLift는 게임에 연결된 모든 플레이어를 “모든 플레이어” 그룹에 배치됩니다. 또한, 개발자는 게임을 위한 기타 그룹을 설정할 수 있으며, 플레이어는 동시에 여러 그룹의 멤버일 수 있습니다. 그룹 멤버는 메시지를 전송하고 게임 데이터를 그룹의 모든 플레이어와 공유할 수 있습니다. 그룹에 가능한 한 가지 사용은 플레이어 팀을 설정하고 팀 통신을 관리하는 것입니다.

TLS 인증서가 있는 Realtime 서버

Realtime 서버를 사용하면 서버 인증 및 데이터 패킷 암호화가 서비스에 기본 제공됩니다. TLS 인증서 생성을 켜면 이러한 보안 기능이 활성화됩니다. 게임 클라이언트가 Realtime 서버와 연결하려고 하면 서버는 TLS 인증서를 사용하여 자동으로 응답하며 클라이언트가 이러한 작업을 확인합니다. Amazon GameLift는 TCP(WebSockets) 통신에 대해 TLS를 사용하고 UDP 트래픽에 대해 DTLS를 사용하여 암호화를 처리합니다.

Realtime 서버 사용자 지정

Realtime 서버는 비저장 릴레이 서버의 역할을 합니다. Realtime 서버는 게임에 연결된 게임 클라이언트 간에 메시지와 게임 데이터의 패킷을 릴레이합니다. 하지만 Realtime 서버는 메시지를 평가, 데이터를 처리 또는 게임플레이 로직을 수행하지 않습니다. 이 방식으로 사용되는 각 게임 클라이언트는 게임 상태 보기를 유지하고 릴레이 서버를 통해 다른 플레이어에게 업데이트를 제공합니다. 각 게임 클라이언트는 이러한 업데이트를 통합하고 자체 게임 상태를 조정합니다.

Realtime 스크립트 기능에 추가하여 서버를 사용자 지정할 수 있습니다. 예를 들어, 게임 로직을 사용하면 서버 권한이 있는 게임 상태 보기와 함께 상태 저장 게임을 빌드할 수 있습니다.

Amazon GameLift는 Realtime 스크립트에 대한 서버 측 콜백 세트를 정의합니다. 이러한 콜백을 구현하여 이벤트 중심 기능을 서버에 추가합니다. 예를 들어, 다음을 수행할 수 있습니다.

- 게임 클라이언트가 서버에 연결하려고 할 때 플레이어를 인증합니다.
- 요청 시 플레이어가 그룹에 참가할 수 있는지 여부를 확인합니다.
- 특정 플레이어에서 또는 대상 플레이어로 메시지를 언제 전송할지를 평가하거나, 응답에서 추가 처리를 결정합니다.
- 플레이어가 그룹에서 나가거나 서버에서 연결이 제거되면 모든 플레이어에게 알립니다.
- 게임 세션 개체 또는 메시지 객체의 콘텐츠를 평가하고 데이터를 확인합니다.

Realtime 서버 배포 및 업데이트

Realtime 서버의 주요 장점은 언제든지 스크립트를 업데이트할 수 있는 기능입니다. 스크립트를 업데이트하면 Amazon GameLift는 몇 분 내에 모든 호스팅 리소스에 새 버전을 배포합니다. Amazon GameLift에서 새 스크립트를 배포한 후에는 그 시점 이후에 생성된 모든 새 게임 세션은 새 스크립트 버전을 사용합니다. (기존 게임 세션에서는 원본 버전을 계속 사용합니다.)

게임에 Realtime 서버 통합 시작하기

- Realtime 서버용 게임 클라이언트 통합
- Realtime 스크립트 생성

Realtime 서버용 게임 클라이언트 통합

이 주제에서는 Amazon GameLift 호스팅 게임 세션에 참가하고 참여할 수 있도록 게임 클라이언트를 준비하는 방법을 설명합니다.

게임 클라이언트를 준비하는 데 필요한 두 개의 작업 세트가 있습니다.

- 기존 게임에 대한 정보를 획득하고 매치메이킹을 요청하며 새로운 게임 세션을 시작하고 플레이어를 위한 게임 세션을 예약하도록 게임 클라이언트를 설정합니다.
- 게임 클라이언트를 활성화하여 Realtime 서버에 호스팅된 게임 세션에 참가하고 메시지를 교환합니다.

게임 세션 및 플레이어 세션 찾기 또는 생성

게임 세션을 찾거나 시작하고 FlexMatch 매치메이킹을 요청하며 게임에서 플레이어 세션을 생성하여 플레이어를 위한 공간을 예약하도록 게임 클라이언트를 설정합니다. 백엔드 서비스를 생성하고, 게임 클라이언트 실행에 의해 트리거될 때 이것을 사용하여 Amazon GameLift 서비스로 직접 요청하는 것이 가장 좋습니다. 그러면 백엔드 서비스가 관련 응답을 게임 클라이언트에 다시 릴레이합니다.

1. AWS SDK를 게임 클라이언트에 추가하고, Amazon GameLift 클라이언트를 초기화하며, 플릿 및 대기열에서 호스팅 리소스를 사용하도록 해당 클라이언트를 구성합니다. AWS SDK는 여러 언어로 제공됩니다(Amazon GameLift SDK 사용자 지정 클라이언트 서비스의 경우 참조).
2. GameLift 기능을 백엔드 서비스에 추가합니다. 자세한 지침은 GameLift Amazon을 게임 클라이언트에 추가 및 FlexMatch 매치메이킹 추가를 참조하세요. 게임 세션 배치를 사용하여 새 게임 세션을 생성하는 것이 모범 사례입니다. 이 방법을 사용하면 새 게임 세션을 신속하고 지능적으로 배치하는 GameLift의 능력을 최대로 활용하는 것은 물론 플레이어 지연 시간 데이터를 사용하여 게임 지연 시간을 최소화할 수 있습니다. 적어도 백엔드 서비스에서 새 게임 세션을 요청하고 그에 따라 게임 세션 데이터를 처리할 수 있어야 합니다. 그리고 기존 게임 세션에 관한 정보를 검색하여 가져오는 기능을 추가하고, 플레이어 세션을 요청하여 기존 게임 세션의 플레이어 슬롯을 효과적으로 유지할 수 있어야 합니다.
3. 연결 정보를 다시 게임 클라이언트에 전달합니다. 백엔드 서비스는 Amazon GameLift 서비스에 대한 요청의 응답에서 게임 세션 및 플레이어 세션 객체를 수신합니다. 이러한 객체에는 게임 클라이언트가 Realtime Server에서 실행 중인 게임 세션에 연결하는 데 필요한 정보, 특히 연결 세부 정보(IP 주소 및 포트) 및 플레이어 세션 ID가 포함됩니다.

Realtime 서버의 게임에 연결

게임 클라이언트를 활성화하여 Realtime 서버에 호스팅 게임 세션과 직접 연결하고 서버 및 다른 플레이어와 메시지를 교환합니다.

1. Realtime Client SDK를 가져오고 빌드하며 게임 클라이언트 프로젝트에 추가합니다. SDK 요구 사항에 대한 자세한 내용과 클라이언트 라이브러리를 빌드하는 방법에 대한 지침은 README 파일을 참조하십시오.
2. 사용할 클라이언트/서버 연결 유형을 지정하는 클라이언트 구성을 사용하여 Client()를 호출합니다.

   > **Note**
   >
   > TLS 인증서를 사용하여 보안 플릿에서 실행되는 Realtime 서버에 연결하는 경우 보안 연결 유형을 지정해야 합니다.

3. 다음 기능을 게임 클라이언트에 추가합니다. 자세한 내용은 Realtime Servers Client API(C#) 참조 섹션을 참조하세요.
   - 게임 연결 및 연결 해제
     - Connect()
     - Disconnect()
   - 대상 수신자에게 메시지 전송
     - SendMessage()
   - 메시지 수신 및 처리
     - OnDataReceived()
   - 그룹 참가 및 플레이어 그룹에서 나가기
     - JoinGroup()
     - RequestGroupMembership()
     - LeaveGroup()
4. 필요에 따라 클라이언트 콜백에 대한 이벤트 핸들러를 설정합니다. Realtime Servers Client API(C#) 참조: 비동기 콜백 섹션을 참조하세요.

TLS 인증서 생성이 활성화된 Realtime 플릿으로 작업하는 경우 서버는 TLS 인증서를 사용하여 자동으로 인증됩니다. TCP 및 UDP 트래픽은 전송 중에 암호화되어 전송 계층 보안을 제공합니다. TCP 트래픽은 TLS 1.2를 사용하여 암호화되고, UDP 트래픽은 DTLS 1.2를 사용하여 암호화됩니다.

게임 클라이언트 예제

Basic Realtime Client(C#)

이 예제에서는 Realtime Client SDK(C#)와 기본 게임 클라이언트 통합을 보여 줍니다. 그림과 같이, 이 예제에서는 Realtime 클라이언트 객체를 초기화하고 이벤트 핸들러를 설정한 다음 클라이언트 측 콜백을 구현하고, Realtime 서버에 연결하고 메시지를 전송한 다음 연결을 해제합니다.

```
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);
        }
    }
}
```

Realtime 스크립트 생성

게임에 Realtime 서버를 사용하려면 Realtime 서버 플릿을 구성하고 사용자 지정(선택 사항)하는 스크립트(JavaScript 코드 형식)를 제공해야 합니다. 이 주제에서는 Realtime 스크립트를 작성하는 주요 단계를 설명합니다. 스크립트가 준비되면 Amazon GameLift 서비스에 업로드한 후 이를 사용하여 플릿을 생성합니다(Amazon GameLift에 Realtime 서버 스크립트 업로드 참조).

Realtime 서버에 스크립트를 사용할 준비를 하려면 다음 기능을 Realtime 스크립트에 추가합니다.

게임 세션 수명 주기 관리(필수)

적어도 Realtime 스크립트에는 Realtime 서버가 게임 세션 시작을 준비하게 하는 `Init()` 기능이 포함되어야 합니다. 그리고 게임 세션을 종료하여 새 게임 세션이 플릿에서 계속 시작되도록 보장하는 방법도 제공할 것을 적극 권장합니다.

`Init()` 콜백 기능은 호출되면 Realtime 세션 객체로 전달되는데, 그 안에는 Realtime 서버용 인터페이스가 포함됩니다. 이 인터페이스에 관한 자세한 내용은 Realtime 서버 인터페이스 섹션을 참조하세요.

게임 세션을 명확하게 종료하려면 스크립트에서 Realtime 서버의 `session.processEnding` 기능도 호출해야 합니다. 이를 위해서는 세션 종료 시점을 결정하는 메커니즘이 필요합니다. 스크립트 예제 코드는 플레이어 연결을 확인하고 지정된 시간 동안 세션에 연결된 플레이어가 없을 경우 게임 세션을 종료하는 간단한 메커니즘을 설명합니다.

Realtime 서버 및 기본적인 구성(서버 프로세스 초기화 및 종료)은 기본적으로 상태 비저장 릴레이 서버로 작동합니다. Realtime 서버는 게임에 연결된 게임 클라이언트 간에 메시지와 게임 데이터를 중계하지만, 데이터를 처리하거나 로직을 수행하기 위한 독립적인 작업을 수행하지는 않습니다. 게임에 필요한 경우 게임 이벤트나 다른 메커니즘에 의해 트리거되는 게임 로직을 선택적으로 추가할 수 있습니다.

서버 측 게임 로직 추가(선택 사항)

Realtime 스크립트에 게임 로직을 추가할 수도 있습니다. 예를 들어, 다음 중 하나 또는 전부를 수행할 수 있습니다. 스크립트 예제 코드는 설명을 제공합니다. Amazon GameLift Realtime 서버 스크립트 참조 섹션을 참조하세요.

- 이벤트 중심 로직을 추가합니다. 콜백 기능을 구현하여 클라이언트 서버 이벤트에 응답합니다. 전체 콜백 목록은 Realtime 서버용 스크립트 콜백 섹션을 참조하세요.
- 서버로 메시지를 전송하여 로직을 트리거합니다. 게임 클라이언트에서 서버로 전송되는 메시지의 특정 작업 코드 세트를 생성하고, 수신을 처리하는 기능을 추가합니다. 콜백 `onMessage`를 사용하고, `gameMessage` 인터페이스를 이용해 메시지 내용을 분석합니다(gameMessage.opcode 참조).
- 게임 로직을 활성화하여 다른 AWS 리소스에 액세스할 수 있습니다. 자세한 내용은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.
- 게임 로직이 실행 중인 인스턴스의 플릿 정보에 액세스할 수 있도록 허용합니다. 자세한 내용은 Amazon GameLift 인스턴스의 플릿 데이터를 가져옵니다. 섹션을 참조하세요.

Realtime 서버 스크립트 예제

이 예제는 Realtime 서버 및 사용자 지정 로직을 배포하는 데 필요한 기본 스크립트에 대해 설명합니다. 여기에는 필수 `Init()` 기능이 포함되어 있으며, 타이머 메커니즘을 사용하여 플레이어 연결이 없는 시간에는 게임 세션을 종료합니다. 그리고 콜백 구현을 포함해 사용자 지정 로직용 후크도 포함되어 있습니다.

```
// 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
};
```

Unity용 Amazon GameLift 플러그인과 게임 통합

이 섹션의 항목에서는 Unity용 Amazon GameLift 플러그인과 이를 사용하여 Amazon에서 호스팅할 멀티플레이어 게임 프로젝트를 준비하는 방법을 설명합니다 GameLift. 플러그인의 안내 워크플로를 사용하여 Unity 개발 환경에서 전적으로 작업하여 Amazon 호스팅의 기본 요구 사항을 완료하세요 GameLift.

GameLift Amazon은 게임 개발자가 세션 기반 멀티플레이어 게임 전용 게임 서버를 관리하고 확장할 수 있는 완전 관리형 서비스입니다. Amazon GameLift 호스팅에 대한 자세한 내용은 을 참조하십시오Amazon GameLift 작동 방식.

- 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드, 버전 2.0.0은 서버 SDK 5.x와 함께 작동하며 Amazon을 지원합니다. GameLift Anywhere
- 서버 SDK 4.x용 유니티용 아마존 GameLift 플러그인 가이드버전 1.0.0은 서버 SDK 4.x 또는 이전 버전에서 작동합니다. 이 버전은 통합 테스트에 Amazon GameLift Local을 사용합니다.

서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드

GameLift Amazon은 Amazon과 호환되도록 멀티플레이어 게임 서버를 준비하는 도구를 제공합니다 GameLift. Unity용 Amazon GameLift 플러그인을 사용하면 GameLift Amazon을 Unity 게임 프로젝트에 쉽게 통합하고, Amazon과의 통합을 테스트하고 GameLift Anywhere, 클라우드 호스팅용 Amazon GameLift 리소스를 배포할 수 있습니다.

이 플러그인은 AWS CloudFormation 템플릿을 사용하여 일반적인 게임 시나리오에 맞는 호스팅 솔루션을 배포합니다. 이러한 솔루션을 제공된 대로 사용하거나 게임에 필요한 대로 사용자 지정하세요.

주제

플러그인 정보

Unity용 플러그인은 Unity 멀티플레이어 게임을 Amazon과 통합하고 호스팅하기 위한 간소화된 시작 환경을 제공합니다. GameLift 플러그인 기능과 사전 빌드된 컴포넌트를 활용하여 게임을 빠르게 시작하고 실행할 수 있습니다.

플러그인은 Unity 에디터에 툴과 기능을 추가합니다. 안내 워크플로를 사용하여 GameLift Amazon을 게임 프로젝트에 통합하고 로컬에서 테스트한 다음 게임 서버를 Amazon GameLift 클라우드 호스팅에 배포하십시오.

플러그인의 사전 구축된 호스팅 솔루션을 사용하여 게임을 배포합니다. 로컬 워크스테이션을 호스트로 사용하여 Amazon GameLift Anywhere 플릿을 설정하십시오. 클라우드 호스팅의 경우 플레이어 지연 시간, 게임 세션 가용성 및 비용의 균형을 서로 다른 방식으로 맞추는 두 가지 일반적인 배포 시나리오 중에서 선택하십시오. 한 가지 시나리오에는 간단한 FlexMatch 매치메이커와 규칙 세트가 포함됩니다. 이러한 시나리오를 사용하여 프로덕션에 바로 사용할 수 있는 기본 호스팅 솔루션을 마련한 다음 필요에 따라 최적화하고 사용자 지정하세요.

플러그인에는 다음과 같은 구성 요소가 포함됩니다.

- Unity 에디터용 플러그인 모듈. 플러그인이 설치되면 새 기본 메뉴 항목을 통해 Amazon GameLift 기능에 액세스할 수 있습니다.
- 클라이언트 측 기능을 갖춘 Amazon GameLift 서비스 API용 C# 라이브러리.
- 아마존 GameLift 서버 SDK용 C# 라이브러리 (버전 5.x).
- 에셋과 씬을 포함한 샘플 게임 콘텐츠를 통해 빌드할 수 있는 멀티플레이어 게임이 GameLift 없더라도 Amazon을 사용해 볼 수 있습니다.

- 호스팅용 클라우드에 게임 서버를 배포할 때 플러그인이 사용하는 솔루션 구성 (AWS CloudFormation템플릿으로 제공).

플러그인 워크플로

다음 단계에서는 Unity용 Amazon GameLift 플러그인과 게임 프로젝트를 통합하고 배포하는 일반적인 접근 방식을 설명합니다. 이 단계는 Unity 에디터와 게임 코드에서 작업하여 완료합니다.

1. 계정에 연결되고 Amazon을 사용할 권한이 있는 유효한 AWS 계정 사용자의 액세스 자격 증명을 제공하는 사용자 프로필을 GameLift 생성하십시오.
2. 게임 프로젝트에 서버 코드를 추가하여 실행 중인 게임 서버와 with Amazon GameLift 서비스 간의 통신을 설정합니다.
3. 게임 클라이언트가 GameLift Amazon에 요청을 보내 게임 세션을 시작하거나 참여하고 게임 서버에 연결할 수 있도록 게임 프로젝트에 클라이언트 코드를 추가합니다.
4. Anywhere 워크플로를 사용하여 로컬 워크스테이션을 게임 서버의 Anywhere 호스트로 설정합니다. 게임 서버와 클라이언트를 로컬에서 시작하고, 게임 세션에 연결하고, 통합을 테스트하십시오.
5. EC2 호스팅 워크플로를 사용하여 통합 게임 서버를 업로드하고 클라우드 호스팅 솔루션을 배포하십시오. 게임 서버가 준비되면 게임 클라이언트를 로컬에서 시작하고 게임 세션에 연결하여 로그인한 다음 게임을 플레이하십시오.

플러그인에서 작업할 때 AWS 리소스를 생성하고 사용하게 됩니다. 이러한 작업을 수행하면 사용 중인 AWS 계정에 요금이 부과될 수 있습니다. 처음 사용하는 AWS 경우 액션이 AWS프리 티어에 포함될 수 있습니다.

Unity용 플러그인 설치

이 섹션에서는 Unity 프로젝트에 플러그인을 추가하는 방법을 설명합니다. 플러그인을 설치한 후 Unity 에디터에서 프로젝트를 열면 플러그인 기능을 사용할 수 있습니다.

시작하기 전에

Unity용 Amazon GameLift 플러그인을 사용하는 데 필요한 사항은 다음과 같습니다.

- 윈도우 2022 LTS용 유니티 또는 macOS용 유니티
- 유니티용 아마존 GameLift 플러그인 다운로드. [다운로드 사이트] 다운로드에는 두 개의 패키지가 포함되어 있습니다.
  - 유니티용 아마존 GameLift 스탠드얼론 플러그인

  - GameLift Unity용 아마존 C# 서버 SDK
- Microsoft Visual Studio 2019 이상 최신 버전.
- C# 게임 코드가 포함된 멀티플레이어 게임 프로젝트.
- 타사의 범위가 지정된 레지스트리. UnityNuGet 이 도구는 타사 DLL을 관리합니다. 자세한 내용은 UnityNuGetGithub 리포지토리를 참조하십시오.

플러그인을 게임 프로젝트에 추가

Unity 에디터와 게임 프로젝트 파일에서 작업하면서 다음 작업을 완료하세요.

1단계: 게임 UnityNuGet 프로젝트에 추가

게임 프로젝트를 UnityNuGet 설정하지 않은 경우 다음 단계에 따라 Unity 패키지 관리자를 사용하여 툴을 설치하십시오. 또는 NuGet CLI를 사용하여 DLL을 수동으로 다운로드할 수 있습니다. 자세한 내용은 Unity용 Amazon GameLift C# 서버 SDK를 참조하십시오. README

1. Unity 에디터에서 프로젝트를 열고 메인 메뉴로 이동하여 편집, 프로젝트 설정을 선택합니다. 옵션에서 패키지 관리자 섹션을 선택하고 범위 지정 레지스트리 그룹을 엽니다.
2. + 버튼을 선택하고 UnityNuGet 범위가 지정된 레지스트리에 다음 값을 입력합니다.

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

Unity 2021 버전 사용자의 경우

설정 UnityNuGet 후 Unity 콘솔에 `Assembly Version Validation` 오류가 표시되는지 확인합니다. 이러한 오류는 NuGet 패키지의 이름이 강한 어셈블리에 대한 바인딩 리디렉션이 Unity 프로젝트 내 경로로 제대로 확인되지 않는 경우 발생합니다. 이 문제를 해결하려면 Unity의 어셈블리 버전 검증을 구성하세요.

1. Unity 에디터에서 메인 메뉴로 이동하여 편집, 프로젝트 설정을 선택하고 플레이어 섹션을 엽니다.
2. 어셈블리 버전 검증 옵션을 선택 취소합니다.

2단계: 플러그인 및 C# 서버 SDK 패키지 추가

1. 두 패키지가 모두 포함된 Unity용 Amazon GameLift 플러그인의 압축을 풉니다.

2. Unity 에디터에서 프로젝트를 열고 메인 메뉴로 이동하여 창, 패키지 관리자를 선택합니다.
3. + 버튼을 선택하여 새 패키지를 추가합니다. tarball에서 패키지 추가 옵션을 선택합니다.
4. 디스크의 패키지 선택에서 Unity 다운로드 파일용 Amazon GameLift C# Server SDK 플러그인을 찾아 파일을 선택합니다. `com.amazonaws.gameliftserver.sdk-<version>.tgz` 열기를 선택하여 플러그인을 설치합니다.
5. 디스크의 패키지 선택에서 Unity 다운로드 파일용 Amazon GameLift 독립형 플러그인을 찾아 파일을 선택합니다. `com.amazonaws.gamelift-<version>.tgz` 열기를 선택하여 플러그인을 설치합니다.
6. 독립형 플러그인이 프로젝트에 추가되었는지 확인하십시오. Unity 에디터 창으로 돌아가십시오. 메인 메뉴에서 새 Amazon GameLift 메뉴 버튼을 확인하십시오.

3단계: 샘플 게임 가져오기 (선택 사항)

Unity용 플러그인에는 게임 프로젝트에 추가할 수 있는 장면을 비롯한 샘플 게임 에셋 세트가 포함되어 있습니다. 샘플 게임을 가져오면 Amazon에서 간단한 멀티플레이어 게임을 빠르게 테스트, 빌드 및 배포할 수 있습니다. GameLift 샘플 게임은 이미 Amazon GameLift SDK와 완전히 통합되어 있으므로 통합 작업을 건너뛰고 나머지 워크플로 작업을 완료할 수 있습니다.

샘플 게임을 사용하면 단 몇 분 만에 로컬에서 호스팅되는 Amazon GameLift Anywhere 플릿을 설정하고 가입할 수 있습니다. 게임을 Amazon에 GameLift 배포하고 1시간 이내에 라이브 클라우드 호스팅 게임에 참여할 수 있습니다.

샘플 게임을 가져오려면:

1. Unity 에디터에서 게임 프로젝트를 열고 Amazon GameLift 메뉴로 이동하여 샘플 게임, 샘플 게임 가져오기를 선택합니다.
2. 파일을 가져온 후 Amazon GameLift 메뉴로 다시 이동하여 샘플 게임, 설정 초기화를 선택합니다. 이 단계에서는 게임 클라이언트와 서버를 빌드하기 위한 프로젝트를 구성합니다.

설치가 완료되면 게임 프로젝트에 두 개의 새로운 씬이 추가된 것을 볼 수 있습니다. 에셋을 비롯한 몇 GameLiftClientSettings가지 추가 프로젝트 에셋도 확인할 수 있습니다.

샘플의 UI와 게임플레이에 대한 자세한 내용은 샘플 게임 readme를 참조하십시오.

AWS 사용자 프로필 설정

플러그인을 설치한 후 프로필을 설정하고 유효한 AWS 계정 사용자에 연결합니다. 프로파일을 여러 개 유지할 수 있지만 한 번에 한 프로필만 활성화할 수 있습니다. 플러그인에서 작업할 때마다 사용할 프로필을 선택합니다.

프로필을 여러 개 유지하면 서로 다른 호스팅 시나리오 간에 전환할 수 있습니다. 예를 들어 AWS 자격 증명은 같지만 다른 AWS 리전인 프로필을 설정할 수 있습니다. 또는 다른 AWS 계정이나 다른 사용자/권한 집합으로 프로필을 설정할 수도 있습니다.

> **Note**
>
> 워크스테이션에 AWS CLI를 설치했고 프로필이 이미 구성되어 있는 경우 Amazon GameLift 플러그인이 이를 감지하여 기존 프로필로 나열합니다. 플러그인은 `[default]` 이름이 지정된 프로필을 자동으로 선택합니다. 기존의 데이터 세트를 사용하거나 새 프로필을 생성할 수 있습니다.

프로필을 설정하려면 AWS

1. Unity 에디터 메인 메뉴에서 GameLiftAmazon을 선택하고 AWS계정 프로필 설정을 선택합니다. 이 작업을 수행하면 플러그인 창이 열립니다. AWS사용자 프로필 페이지를 엽니다.
2. 플러그인이 기존 프로필을 감지하면 프로필을 만들라는 메시지가 표시되지 않습니다. 기존 프로필을 선택하거나 다른 프로필 추가를 선택하여 새 프로필을 생성합니다.
3. 플러그인이 기존 프로필을 감지하지 못하면 프로필을 만들라는 메시지가 표시됩니다. 새 계정이나 기존 AWS 계정을 사용하여 새 프로필을 만들 수 있습니다.

   > **Note**
   >
   > AWS Management Console을 사용하여 새 AWS 계정을 생성하고 적절한 권한 설정으로 사용자를 생성하거나 업데이트해야 합니다.

   프로필을 설정할 때는 다음 정보가 필요합니다.

   - AWS 계정. 새 AWS 계정을 생성해야 하는 경우 메시지에 따라 계정을 생성합니다. 자세한 내용은 AWS 계정 생성을 참조하세요.

- Amazon GameLift 및 기타 필수 AWS 서비스를 사용할 권한이 있는 AWS 계정 사용자 Amazon GameLift 권한을 가진 AWS Identity and Access Management (IAM) 사용자를 설정하는 방법에 설정 AWS 계정 대한 지침은 을 참조하십시오.
- AWS 사용자의 자격 증명. 또한 이 사용자에게는 장기 자격 증명을 통한 프로그래밍 액세스 권한이 필요합니다. 이러한 자격 증명은 AWS 액세스 키와 AWS 보안 키로 구성되어 있습니다. 자세한 내용은 액세스 키 가져오기를 참조하세요.
- AWS 리전. 이는 호스팅용 AWS 리소스를 생성하려는 지리적 위치입니다. 개발 중에는 실제 위치와 가까운 지역을 사용하여 지연 시간을 최소화하는 것이 좋습니다. 지원되는 AWS 리전 목록을 참조하세요.

4. 프로필을 선택하거나 생성한 후에는 프로필의 부트스트랩 상태를 확인하고 필요에 따라 조치를 취하십시오. Amazon GameLift 플러그인 기능을 사용하려면 모든 프로필을 부트스트랩해야 합니다.

프로필을 부트스트랩하려면

부트스트래핑은 선택한 사용자 프로필과 함께 사용할 Amazon S3 버킷을 지정합니다. Amazon S3는 데이터 및 객체 스토리지를 위한 핵심 AWS 서비스입니다. 프로젝트 구성, 빌드 아티팩트 및 기타 종속성을 저장하는 데 사용되는 버킷입니다. 버킷은 다른 프로필 간에 공유되지 않습니다.

> **Note**
>
> 부트스트래핑은 새로운 AWS 리소스를 생성하므로 비용이 발생할 수 있습니다.

1. 플러그인 창의 AWS사용자 프로필에서 프로필을 볼 때 사용하려는 프로필을 선택합니다. 프로필이 아직 부트스트랩되지 않은 경우 경고 메시지가 표시됩니다.
2. 프로필 부트스트랩 섹션의 드롭다운 목록에서 프로필을 선택하고 부트스트랩 상태를 확인합니다. 버킷이 없는 것으로 상태가 표시되면 프로필 부트스트랩 버튼을 선택합니다. 버킷 이름을 새 버킷 이름으로 설정하거나, 액세스 권한이 있는 기존 버킷을 입력하거나, 자동 생성된 이름을 유지할 수 있습니다.
3. 부트스트랩 상태가 “활성”으로 변경될 때까지 기다립니다. 몇 분 정도 소요될 수 있습니다. 상태가 “Active”이면 프로필을 사용하여 플러그인 기능을 사용할 수 있습니다.

Amazon에서 게임을 로컬 테스트용으로 설정하세요. GameLift Anywhere

이 워크플로우에서는 Amazon GameLift 기능을 위한 클라이언트 및 서버 게임 코드를 추가하고 플러그인을 사용하여 로컬 워크스테이션을 테스트 게임 서버 호스트로 지정합니다. 통합 작업을 완료하면 플러그인을 사용하여 게임 클라이언트 및 서버 구성 요소를 빌드합니다.

Amazon GameLift Anywhere 워크플로를 시작하려면:

- Unity 에디터 메인 메뉴에서 GameLiftAmazon을 선택하고 어디서든 호스팅하기를 선택합니다. 그러면 @ Anywhere 플릿으로 게임을 설정할 수 있는 플러그인 페이지가 열립니다. 이 페이지는 게임 구성 요소를 통합, 빌드 및 출시하기 위한 5단계 프로세스를 제공합니다.

프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

1. 사용 가능한 프로필 드롭다운 목록에서 프로필을 선택합니다. 아직 프로필이 없거나 새 프로필을 만들려면 Amazon GameLift 메뉴로 이동하여 AWS계정 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 “활성”이 아닌 경우, 프로필 부트스트랩을 선택하고 상태가 “활성”으로 변경될 때까지 기다립니다.

게임을 Amazon과 통합하세요 GameLift

> **Note**
>
> 샘플 게임을 가져온 경우 이 단계를 건너뛰어도 됩니다. 샘플 게임 에셋에는 이미 필요한 서버 및 클라이언트 코드가 있습니다.

워크플로의 이 단계에서는 게임 프로젝트의 클라이언트 및 서버 코드를 업데이트합니다.

- * 게임 서버는 Amazon GameLift 서비스와 통신하여 게임 세션을 시작하라는 메시지를 수신하고, 게임 세션 연결 정보를 제공하고, 상태를 보고할 수 있어야 합니다.
- 게임 클라이언트는 게임 세션에 대한 정보를 얻고, 게임 세션에 참여하거나 게임을 시작하고, 연결 정보를 얻을 수 있어야 게임에 참가할 수 있습니다.

서버 코드 통합

커스텀 씬이 포함된 자체 게임 프로젝트를 사용하는 경우 제공된 샘플 코드를 사용하여 게임 프로젝트에 필요한 서버 코드를 추가하세요.

1. 게임 프로젝트 파일에서 `Assets/Scripts/Server` 폴더를 엽니다. 폴더가 없으면 새로 만드세요.
2. aws/ GitHub amazon-gamelift-plugin-unity 저장소로 가서 경로를 여세요. `Samples~/SampleGame/Assets/Scripts/Server`
3. GameLiftServer.cs. 파일을 찾아 게임 프로젝트의 Server 폴더에 복사합니다. 서버 실행 파일을 빌드할 때는 이 파일을 빌드 타겟으로 사용하십시오.

샘플 코드에는 Amazon GameLift C# 서버 SDK (버전 5) 를 사용하는 다음과 같은 최소 필수 요소가 포함되어 있습니다.

- Amazon GameLift API 클라이언트를 초기화합니다. Amazon GameLift Anywhere 플릿에는 서버 파라미터를 사용한 `InitSDK()` 호출이 필요합니다. 이러한 설정은 플러그인에서 사용할 수 있도록 자동으로 설정됩니다.
- , 및 등 `OnStartGameSession` Amazon GameLift 서비스의 요청에 응답하는 데 필요한 콜백 함수를 구현합니다. `OnProcessTerminate` `onHealthCheck`
- 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 지정된 `ProcessReady()` 포트로 호출하여 Amazon GameLift 서비스에 알립니다.

샘플 서버 코드를 사용자 정의하려면 다음 리소스를 참조하십시오.

- Amazon GameLift를 게임 서버에 추가
- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조

클라이언트 코드 통합

커스텀 씬이 포함된 자체 게임 프로젝트를 사용하는 경우 기본 기능을 게임 클라이언트에 통합해야 합니다. 플레이어가 로그인하고 게임 세션에 참여할 수 있도록 UI 요소도 추가해야 합니다. Amazon GameLift 서비스 API (AWSSDK) 를 사용하여 게임 세션 정보를 가져오거나, 새 게임 세션을 생성하거나, 기존 게임 세션에 참여할 수 있습니다.

Anywhere 플릿으로 로컬 테스트용 클라이언트를 구축할 때 Amazon GameLift 서비스에 직접 호출을 추가할 수 있습니다. 클라우드 호스팅용 게임을 개발하거나 프로덕션 호스팅에 Anywhere fleets를 사

용하려는 경우 게임 클라이언트와 Amazon 서비스 간의 모든 통신을 처리할 클라이언트 측 백엔드 서비스를 생성해야 합니다. GameLift

GameLift Amazon을 클라이언트 코드에 통합하려면 다음 리소스를 가이드로 사용하십시오.

- 클라이언트를 amazon-gamelift-plugin-unity aws/ GitHub 리포지토리의 GameLiftCoreApi 클래스와 통합하십시오. 이 클래스는 플레이어 인증 및 게임 세션 정보 검색을 위한 컨트롤을 제공합니다.
- aws/, GitHub 리포지토리에서 사용할 수 있는 샘플 게임 통합을 확인하세요. amazon-gamelift-plugin-unity `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- Unity 게임 GameLift 클라이언트에 Amazon 추가의 지침을 따르십시오.

Anywhere 플릿에 연결하는 게임 클라이언트의 경우 게임 클라이언트에 다음 정보가 필요합니다. 플러그인은 플러그인에서 생성한 리소스를 사용하도록 게임 프로젝트를 자동으로 업데이트합니다.

- FleetId - 애니웨어 플릿의 고유 식별자.
- FleetLocation - 애니웨어 플릿의 사용자 지정 위치
- AwsRegion - 애니웨어 플릿이 호스팅되는 AWS 지역. 사용자 프로필에서 설정한 지역입니다.
- ProfileName - AWS SDK에 대한 액세스를 허용하는 로컬 컴퓨터의 AWS 자격 증명 프로필. GameLift 게임 클라이언트는 이러한 자격 증명을 사용하여 Amazon GameLift 서비스에 대한 요청을 인증합니다.

> **Note**
>
> 자격 증명 프로필은 플러그인에 의해 생성되고 로컬 컴퓨터에 저장됩니다. 따라서 로컬 시스템(또는 프로필이 동일한 시스템)에서 클라이언트를 실행해야 합니다.

애니웨어 플릿에 연결

이 단계에서는 사용할 Anywhere 플릿을 지정합니다. Anywhere 플릿은 게임 서버 호스팅을 위해 어디에서나 배치할 수 있는 컴퓨팅 리소스 모음을 정의합니다.

- 현재 사용 중인 AWS 계정에 기존 Anywhere 플릿이 있는 경우 플릿 이름 드롭다운 필드를 열고 플릿을 선택합니다. 이 드롭다운에는 현재 활성 사용자 프로필의 AWS 리전 내 Anywhere 플릿만 표시합니다.
- 기존 플릿이 없거나 새 플릿을 생성하려는 경우, Create new Anywhere 플릿을 선택하고 플릿 이름을 입력합니다.

프로젝트에 대해 Anywhere 플릿을 선택하면 Amazon은 플릿 상태가 활성 상태인지 GameLift 확인하고 플릿 ID를 표시합니다. Unity 에디터의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

컴퓨팅 등록

이 단계에서는 로컬 워크스테이션을 새로운 Anywhere 플릿의 컴퓨팅 리소스로 등록합니다.

1. 로컬 시스템의 컴퓨팅 이름을 입력합니다. 플릿에 두 개 이상의 컴퓨팅을 추가하는 경우 이름은 고유해야 합니다.
2. 컴퓨팅 등록을 선택합니다. Unreal 편집기의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

플러그인은 로컬 워크스테이션을 로컬 호스트 (127.0.0.1) 로 설정된 IP 주소로 등록합니다. 이 설정은 게임 클라이언트와 서버를 같은 컴퓨터에서 실행한다고 가정합니다.

이 조치에 대한 응답으로 Amazon은 컴퓨팅에 연결할 수 GameLift 있는지 확인하고 새로 등록된 컴퓨팅에 대한 정보를 반환합니다.

게임 실행

이 단계에서는 게임 컴포넌트를 빌드하고 실행하여 게임을 플레이합니다. 다음 단계를 완료합니다.

1. 게임 클라이언트를 구성합니다. 이 단계에서는 게임 프로젝트의 `GameLiftClientSettings` 에셋을 업데이트하라는 메시지를 플러그인에 표시합니다. 플러그인은 이 에셋을 사용하여 게임 클라이언트가 Amazon GameLift 서비스에 연결하는 데 필요한 특정 정보를 저장합니다.

    a. 샘플 게임을 임포트하여 초기화하지 않았다면 새 `GameLiftClientSettings` 에셋을 생성하십시오. Unity 에디터 메인 메뉴에서 에셋, 생성 GameLift, 클라이언트 설정을 선택합니다. 프로젝트에 복사본을 여러 개 생성하면 플러그인이 이를 자동으로 감지하여 플러그인이 업데이트할 에셋을 알려줍니다. GameLiftClientSettings

    b. 게임 실행에서 [클라이언트 구성: 어디서나 설정 적용] 을 선택합니다. 이렇게 하면 방금 설정한 Anywhere 플릿을 사용하도록 게임 클라이언트 설정이 업데이트됩니다.
2. 게임 클라이언트를 빌드하고 실행합니다.

    a. 표준 Unity 빌드 프로세스를 사용하여 클라이언트 실행 파일을 빌드합니다. 파일, 빌드 설정에서 플랫폼을 Windows, Mac, Linux로 전환합니다. 샘플 게임을 가져와서 설정을 초기화한 경우 빌드 목록과 빌드 타겟이 자동으로 업데이트됩니다.

    b. 새로 빌드한 게임 클라이언트 실행 파일의 인스턴스를 하나 이상 실행합니다.

3. Anywhere 플릿에서 게임 서버를 시작하세요. 서버: 에디터에서 서버 시작을 선택합니다. 이 태스크는 Unity 에디터가 열려 있는 한 클라이언트가 연결할 수 있는 라이브 서버를 시작합니다.
4. 게임 세션을 시작하거나 참여하세요. 게임 클라이언트 인스턴스에서 UI를 사용하여 각 클라이언트를 게임 세션에 참여시키십시오. 이 작업을 수행하는 방법은 클라이언트에 기능을 추가한 방식에 따라 달라집니다.

샘플 게임 클라이언트를 사용하는 경우 다음과 같은 특징이 있습니다.

- 플레이어 로그인 컴포넌트. Anywhere 플릿의 게임 서버에 연결할 때는 플레이어 검증이 이루어지지 않습니다. 아무 값이나 입력하여 게임 세션에 참여할 수 있습니다.
- 간단한 게임 참여 UI. 클라이언트가 게임에 참여하려고 하면 클라이언트는 사용 가능한 플레이어 슬롯이 있는 활성 게임 세션을 자동으로 찾습니다. 사용 가능한 게임 세션이 없는 경우 클라이언트는 새 게임 세션을 요청합니다. 게임 세션을 사용할 수 있는 경우 클라이언트는 사용 가능한 게임 세션에 참여하도록 요청합니다. 여러 개의 동시 클라이언트로 게임을 테스트하는 경우 첫 번째 클라이언트가 게임 세션을 시작하고 나머지 클라이언트는 자동으로 기존 게임 세션에 참여합니다.
- 플레이어 슬롯이 4개인 게임 세션. 최대 4개의 게임 클라이언트 인스턴스를 동시에 실행할 수 있으며, 이 경우 인스턴스는 동일한 게임 세션에 참여하게 됩니다.

서버 실행 파일에서 실행 (선택 사항)

Anywhere 플릿에서 테스트용 게임 서버 실행 파일을 빌드하고 실행할 수 있습니다.

1. 표준 Unity 빌드 프로세스를 사용하여 서버 실행 파일을 빌드하세요. 파일, 빌드 설정에서 플랫폼을 전용 서버로 전환하고 빌드합니다.
2. Anywhere 플릿 ID 및 AWS 지역을 get-compute-auth-token사용하여 AWS CLI 명령을 호출하여 단기 인증 토큰을 받으세요. 플릿을 생성할 때 Connect to a Anywhere 플릿에 플릿 ID가 표시됩니다. 활성 프로필을 선택하면 프로필 설정에 AWS 지역이 표시됩니다.

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

3. 명령줄에서 새로 빌드한 게임 서버 실행 파일을 실행하고 유효한 인증 토큰을 전달하세요.

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

관리형 EC2 플릿을 사용하여 클라우드 호스팅에 게임을 배포합니다.

이 워크플로에서는 플러그인을 사용하여 GameLift Amazon에서 관리하는 클라우드 기반 컴퓨팅 리소스에서 게임을 호스팅할 수 있도록 준비합니다. Amazon GameLift 기능을 위한 클라이언트 및 서버 게임 코드를 추가한 다음 서버 빌드를 Amazon GameLift 서비스에 업로드하여 호스팅합니다. 이 워크플로가 완료되면 클라우드에서 게임 서버를 실행하고 해당 서버에 연결할 수 있는 작동하는 게임 클라이언트를 갖게 됩니다.

Amazon GameLift 관리형 Amazon EC2 워크플로를 시작하려면:

- Unity 에디터 메인 메뉴에서 GameLiftAmazon을 선택하고 관리형 EC2를 사용하는 호스트를 선택합니다. 이 워크플로는 게임 구성 요소를 통합, 구축, 배포 및 시작하는 6단계 프로세스를 제공합니다.

프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

1. 사용 가능한 프로필 드롭다운 목록에서 프로필을 선택합니다. 아직 프로필이 없거나 새 프로필을 만들려면 Amazon GameLift 메뉴로 이동하여 AWS계정 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 “활성”이 아닌 경우, 프로필 부트스트랩을 선택하고 상태가 “활성”으로 변경될 때까지 기다립니다.

게임을 Amazon과 통합하세요 GameLift

이 작업을 수행하려면 게임 프로젝트의 클라이언트 및 서버 코드를 업데이트해야 합니다.

- 게임 서버는 Amazon GameLift 서비스와 통신하여 게임 세션을 시작하라는 메시지를 수신하고, 게임 세션 연결 정보를 제공하고, 상태를 보고할 수 있어야 합니다.
- 게임 클라이언트는 게임 세션에 대한 정보를 얻고, 게임 세션에 참여하거나 게임을 시작하고, 연결 정보를 얻을 수 있어야 게임에 참가할 수 있습니다.

> **Note**
>
> 샘플 게임을 가져온 경우 이 단계를 건너뛰어도 됩니다. 샘플 게임 에셋에는 이미 필요한 서버 및 클라이언트 코드가 있습니다.

서버 코드 통합

커스텀 씬과 함께 자체 게임 프로젝트를 사용하는 경우 제공된 샘플 코드를 사용하여 게임 프로젝트에 필요한 서버 코드를 추가하세요. 테스트용 게임 프로젝트를 Anywhere 플릿과 통합했다면 이 단계의 지침을 이미 완료한 것입니다.

1. 게임 프로젝트 파일에서 `Assets/Scripts/Server` 폴더를 엽니다. 폴더가 없으면 새로 만드세요.
2. aws/ GitHub amazon-gamelift-plugin-unity 저장소로 가서 경로를 여세요. `Samples~/SampleGame/Assets/Scripts/Server`
3. 파일을 `GameLiftServer.cs` 찾아 게임 프로젝트의 폴더에 복사하세요. `Server` 서버 실행 파일을 빌드할 때는 이 파일을 빌드 타겟으로 사용하십시오.

샘플 코드에는 Amazon GameLift C# 서버 SDK (버전 5) 를 사용하는 다음과 같은 최소 필수 요소가 포함되어 있습니다.

- Amazon GameLift API 클라이언트를 초기화합니다. Amazon Anywhere 플릿에는 서버 파라미터를 포함한 initSDK () 호출이 필요합니다. GameLift 이러한 설정은 플러그인에서 사용하도록 자동으로 설정됩니다.
- , 및 등 `OnStartGameSession` Amazon GameLift 서비스의 요청에 응답하는 데 필요한 콜백 함수를 구현합니다. `OnProcessTerminate` onHealthCheck
- 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 지정된 `ProcessReady()` 포트로 호출하여 Amazon GameLift 서비스에 알립니다.

샘플 서버 코드를 사용자 정의하려면 다음 리소스를 참조하십시오.

- Amazon GameLift를 게임 서버에 추가
- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조

클라이언트 코드 통합

클라우드 기반 게임 서버에 연결하는 게임 클라이언트의 경우 게임 클라이언트에서 직접 호출하는 대신 클라이언트 측 백엔드 서비스를 사용하여 Amazon GameLift 서비스를 호출하는 것이 가장 좋습니다.

관리형 EC2 플릿에서 호스팅하기 위한 플러그인 워크플로의 각 배포 시나리오에는 다음 구성 요소를 포함하는 사전 구축된 백엔드 서비스가 포함됩니다.

- 게임 세션을 요청하고 게임 세션 정보를 검색하는 데 사용되는 Lambda 함수 및 DynamoDB 테이블 세트입니다. 이러한 구성 요소는 API 게이트웨이를 프록시로 사용합니다.
- 고유한 플레이어 ID를 생성하고 플레이어 연결을 인증하는 Amazon Cognito 사용자 풀입니다.

이러한 구성 요소를 사용하려면 게임 클라이언트가 백엔드 서비스에 요청을 전송하여 다음을 수행하는 기능이 필요합니다.

- AWSCognito 사용자 풀에서 플레이어 사용자를 생성하고 인증합니다.
- 게임 세션에 참여하고 연결 정보를 받으세요.
- 매치메이킹을 사용하여 게임에 참여하세요.

다음 리소스를 가이드로 활용하세요.

- 클라이언트를 amazon-gamelift-plugin-unityaws/ GitHub 리포지토리의 GameLiftCoreApi클래스와 통합하십시오. 이 클래스는 플레이어 인증 및 게임 세션 정보 검색을 위한 컨트롤을 제공합니다.
- 샘플 게임 통합을 보려면 aws/, GitHub 리포지토리로 이동하십시오. amazon-gamelift-plugin-unity `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- 유니티 게임 GameLift 클라이언트에 Amazon을 추가하세요.

배포 시나리오 선택

이 단계에서는 지금 배포하려는 게임 호스팅 솔루션을 선택합니다. 어떤 시나리오로든 게임을 여러 번 배포할 수 있습니다.

- 단일 지역 플릿: 활성 프로필의 기본 AWS 지역에 있는 단일 호스팅 리소스 플릿에 게임 서버를 배포합니다. 처음에는 이 시나리오로 AWS와의 서버 통합 및 서버 빌드 구성을 테스트를 시작하는 것이 좋습니다. 다음과 같은 리소스를 배포합니다.
  - 게임 서버 빌드가 설치되어 실행 중인 AWS 플릿(온디맨드).

  - 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
  - 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
  - API 게이트웨이에 대한 과도한 플레이어 호출을 제한하기 위한 WebACl.
  - API 게이트웨이 + 플레이어가 게임 슬롯을 요청할 수 있는 Lambda 함수. 이 함수는 사용할 수 없는 경우 `CreateGameSession()`을 호출합니다.
  - API 게이트웨이 + 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 Lambda 함수.
- FlexMatch 플릿: 게임 서버를 플릿 세트에 배포하고 플레이어 매치를 생성하는 규칙이 포함된 FlexMatch 매치메이커를 설정합니다. 이 시나리오에서는 안정적인 가용성을 위해 다중 플릿, 다중 위치 구조의 저렴한 스팟 호스팅을 사용합니다. 이 접근 방식은 호스팅 솔루션을 위한 매치메이커 구성 요소 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오에서는 이 솔루션의 기본 리소스를 만들고 나중에 필요에 따라 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.
  - FlexMatch 플레이어 요청을 수락하고 매치를 구성하도록 매치메이킹 구성 및 매치메이킹 규칙을 설정합니다.
  - 게임 서버 빌드가 여러 위치에 설치되어 실행 중인 AWS 플릿 3대. 백업용으로 스팟 플릿 2개와 온디맨드 플릿 1개가 포함됩니다.
  - AWS 게임 세션 배치 대기열은 (실행 가능성, 비용, 플레이어 지연 시간 등을) 기반으로 가능한 최상의 호스팅 리소스를 찾고 게임 세션을 시작하여 제안된 매치에 대한 요청을 충족시켜 줍니다.
  - 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
  - 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
  - API 게이트웨이에 대한 과도한 플레이어 호출을 제한하기 위한 WebACl.
  - API 게이트웨이 + 플레이어가 게임 슬롯을 요청할 수 있는 Lambda 함수. `StartMatchmaking()` 함수를 호출합니다.
  - API 게이트웨이 + 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 Lambda 함수.
  - 플레이어용 매치메이킹 티켓과 게임 세션 정보를 저장하는 Amazon DynamoDB 테이블.
  - SNS 주제 + 이벤트를 처리하는 Lambda 함수. GameSessionQueue

게임 파라미터를 설정합니다.

이 단계에서는 AWS에 업로드할 게임을 설명합니다.

- 게임 이름: 게임 프로젝트에 의미 있는 이름을 입력합니다. 이 이름은 플러그인 내에서 사용됩니다.
- 플릿 이름: 관리형 EC2 플릿에 의미 있는 이름을 제공하십시오. GameLift Amazon은 AWS 콘솔에 리소스를 나열할 때 플릿 ID와 함께 이 이름을 사용합니다.

- 빌드 이름: 서버 빌드에 의미 있는 이름을 제공하십시오. AWS는 GameLift Amazon에 업로드되어 배포에 사용되는 서버 빌드의 사본을 가리키기 위해 이 이름을 사용합니다.
- 시작 파라미터: 관리형 EC2 플릿 인스턴스에서 서버 실행 파일을 시작할 때 실행할 선택적 지침을 입력합니다. 최대 길이는 1024자입니다.
- 게임 서버 폴더: 서버 빌드가 들어 있는 로컬 폴더의 경로를 입력합니다.
- 게임 서버 파일: 서버 실행 파일 이름을 지정합니다.

배포 시나리오

이 단계에서는 선택한 배포 시나리오에 따라 게임을 클라우드 호스팅 솔루션에 배포합니다. 이 프로세스는 AWS가 서버 빌드를 검증하고, 호스팅 리소스를 프로비저닝하며, 게임 서버를 설치하고, 서버 프로세스를 시작하며, 게임 세션을 호스팅할 준비를 하는 데 최대 40분 정도 걸릴 수 있습니다.

배포를 시작하려면 [Deploy] 를 선택합니다 CloudFormation. 여기에서 게임 호스팅 상태를 추적할 수 있습니다. 자세한 내용은 AWS의 AWS 관리 콘솔에 로그인하여 이벤트 알림을 확인할 수 있습니다. 플러그인의 활성 사용자 프로필과 동일한 계정, 사용자 및 AWS 리전을 사용하여 로그인해야 합니다.

배포가 완료되면 게임 서버가 AWS EC2 인스턴스에 설치됩니다. 하나 이상의 서버 프로세스가 실행 중이며 게임 세션을 시작할 준비가 되었습니다.

게임 클라이언트 실행

플릿이 성공적으로 배포되면 이제 게임 서버가 실행되고 게임 세션을 호스팅할 수 있습니다. 이제 클라이언트를 빌드하고, 시작하고, 연결하여 게임 세션에 참여할 수 있습니다.

1. 게임 클라이언트를 구성하세요. 이 단계에서는 게임 프로젝트의 `GameLiftClientSettings` 에셋을 업데이트하라는 메시지를 플러그인에 표시합니다. 플러그인은 이 에셋을 사용하여 게임 클라이언트가 Amazon GameLift 서비스에 연결하는 데 필요한 특정 정보를 저장합니다.

    a. 샘플 게임을 임포트하여 초기화하지 않았다면 새 `GameLiftClientSettings` 에셋을 생성하십시오. Unity 에디터 메인 메뉴에서 에셋, 생성 GameLift, 클라이언트 설정을 선택합니다. 프로젝트에 복사본을 여러 개 생성하면 플러그인이 이를 자동으로 감지하여 플러그인이 업데이트할 에셋을 알려줍니다. GameLiftClientSettings

    b. Launch Game에서 [클라이언트 구성: 관리형 EC2 설정 적용] 을 선택합니다. 이 작업을 수행하면 방금 배포한 관리형 EC2 플릿을 사용하도록 게임 클라이언트 설정이 업데이트됩니다.
2. 게임 클라이언트를 빌드하세요. 표준 Unity 빌드 프로세스를 사용하여 클라이언트 실행 파일을 빌드합니다. 파일, 빌드 설정에서 플랫폼을 Windows, Mac, Linux로 전환합니다. 샘플 게임을 가져와서 설정을 초기화한 경우 빌드 목록과 빌드 타겟이 자동으로 업데이트됩니다.

3. 새로 빌드한 게임 클라이언트 실행 파일을 실행합니다. 게임을 시작하려면 2~4개의 클라이언트 인스턴스를 시작하고 각 인스턴스의 UI를 사용하여 게임 세션에 참여하십시오.

샘플 게임 클라이언트를 사용하는 경우 다음과 같은 특징이 있습니다.

- 플레이어 로그인 컴포넌트. Anywhere 플릿의 게임 서버에 연결할 때는 플레이어 검증이 이루어지지 않습니다. 아무 값이나 입력하여 게임 세션에 참여할 수 있습니다.
- 간단한 게임 참여 UI. 클라이언트가 게임에 참여하려고 하면 클라이언트는 사용 가능한 플레이어 슬롯이 있는 활성 게임 세션을 자동으로 찾습니다. 사용 가능한 게임 세션이 없는 경우 클라이언트는 새 게임 세션을 요청합니다. 게임 세션을 사용할 수 있는 경우 클라이언트는 사용 가능한 게임 세션에 참여하도록 요청합니다. 여러 개의 동시 클라이언트로 게임을 테스트하는 경우 첫 번째 클라이언트가 게임 세션을 시작하고 나머지 클라이언트는 자동으로 기존 게임 세션에 참여합니다.
- 플레이어 슬롯이 4개인 게임 세션. 최대 4개의 게임 클라이언트 인스턴스를 동시에 실행할 수 있으며, 이 경우 인스턴스는 동일한 게임 세션에 참여하게 됩니다.

서버 SDK 4.x용 유니티용 아마존 GameLift 플러그인 가이드

> **Note**
>
> 이 주제에서는 Unity용 Amazon GameLift 플러그인의 이전 버전에 대한 정보를 제공합니다. 버전 1.0.0 (2021년 출시) 은 아마존 GameLift 서버 SDK 4.x 또는 이전 버전을 사용합니다. 서버 SDK 5.x를 사용하고 GameLift Anywhere Amazon을 지원하는 최신 버전의 플러그인에 대한 설명서는 을 참조하십시오. 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드

GameLift Amazon은 Amazon에서 실행할 멀티플레이어 게임 서버를 준비하는 도구를 제공합니다 GameLift. Unity용 Amazon GameLift 플러그인을 사용하면 GameLift Amazon을 Unity 게임 프로젝트에 쉽게 통합하고 클라우드 호스팅용 Amazon GameLift 리소스를 배포할 수 있습니다. Unity용 플러그인을 사용하여 Amazon GameLift API에 액세스하고 일반적인 게임 시나리오에 맞는 AWS CloudFormation 템플릿을 배포할 수 있습니다.

플러그인을 설정한 후에는 에서 Amazon GameLift Unity 샘플을 사용해 볼 수 GitHub 있습니다.

주제

- GameLift Amazon을 유니티 게임 서버 프로젝트와 통합
- GameLift Amazon을 Unity 게임 클라이언트 프로젝트와 통합

- 플러그인 설치 및 설정
- 게임을 로컬에서 테스트하세요.
- 시나리오 배포
- GameLift Unity에서 게임을 Amazon과 통합
- 샘플 게임 임포트 및 실행

GameLift Amazon을 유니티 게임 서버 프로젝트와 통합

> **Note**
>
> 이 주제에서는 Unity용 Amazon GameLift 플러그인의 이전 버전에 대한 정보를 제공합니다. 버전 1.0.0 (2021년 출시) 은 아마존 GameLift 서버 SDK 4.x 또는 이전 버전을 사용합니다. 서버 SDK 5.x를 사용하고 GameLift Anywhere Amazon을 지원하는 최신 버전의 플러그인에 대한 설명서는 을 참조하십시오. 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드

이 항목은 Amazon에서 호스팅할 사용자 지정 게임 서버를 준비하는 데 도움이 GameLift 됩니다. 게임 서버는 GameLift Amazon에 상태를 알리고, 메시지가 표시되면 게임 세션을 시작 및 중지하고, 기타 작업을 수행할 수 있어야 합니다. 자세한 정보는 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

사전 조건

게임 서버를 통합하기 전에 먼저 다음 작업을 완료합니다.

- Amazon에 대한 IAM 서비스 역할 설정 GameLift
- Unity용 플러그인 설치

새로운 서버 프로세스 설정

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

GameLift Amazon과의 통신을 설정하고 서버 프로세스가 게임 세션을 호스팅할 준비가 되었다고 보고하십시오.

1. `InitSDK()`를 호출하여 서버 SDK를 초기화합니다.
2. 서버가 게임 세션을 수락할 수 있도록 준비하려면 연결 포트 및 게임 세션 위치 세부 정보를 포함하여 `ProcessReady()`를 호출합니다. Amazon GameLift 서비스가 호출하는 콜백 함수의 이름 (예`OnGameSession()`:,,) 을 포함하십시오. `OnGameSessionUpdate()` `OnProcessTerminate()` `OnHealthCheck()` Amazon에서 콜백을 제공하는 데 몇 분 정도 GameLift 걸릴 수 있습니다.
3. Amazon은 서버 프로세스 상태를 로 GameLift `ACTIVE` 업데이트합니다.
4. Amazon은 `onHealthCheck` 정기적으로 GameLift 전화를 겁니다.

다음 코드 예제는 Amazon에서 간단한 서버 프로세스를 설정하는 방법을 보여줍니다 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;
}
```

게임 세션 시작

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

게임 초기화가 완료된 후 게임 세션을 시작할 수 있습니다.

1. 콜백 함수 `onStartGameSession`을 구현합니다. Amazon은 이 메서드를 GameLift 호출하여 서버 프로세스에서 새 게임 세션을 시작하고 플레이어 연결을 수신합니다.
2. 게임 세션을 활성화하려면 `ActivateGameSession()`을 호출합니다. SDK에 대한 자세한 내용은 Amazon GameLift Server SDK(C#) 참조: 작업 섹션을 참조하세요.

다음 코드 예제는 Amazon에서 게임 세션을 시작하는 방법을 보여줍니다 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();
```

```
}
```

게임 세션 종료

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

게임 세션이 GameLift 종료되면 Amazon에 알리십시오. 호스팅 리소스를 재활용하고 새로 고치려면 게임 세션이 완료된 후 서버 프로세스를 종료하는 것이 가장 좋습니다.

1. Amazon으로부터 요청을 GameLift 수신하고 `onProcessTerminate` 호출하도록 이름이 지정된 함수를 설정합니다`ProcessEnding()`.
2. 프로세스 상태가 `TERMINATED`로 변경됩니다.

다음 예에서는 게임 세션의 프로세스 종료하는 방법에 대해 설명합니다.

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

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

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

서버 빌드 생성 및 Amazon에 업로드 GameLift

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

게임 서버를 Amazon과 통합한 후 GameLift Amazon에서 게임 호스팅용으로 GameLift 배포할 수 있도록 빌드 파일을 플릿에 업로드하십시오. GameLiftAmazon에 서버를 업로드하는 방법에 대한 자세한 내용은 을 참조하십시오Amazon GameLift에 사용자 지정 서버 빌드 업로드.

GameLift Amazon을 Unity 게임 클라이언트 프로젝트와 통합

> **Note**
>
> 이 주제에서는 Unity용 Amazon GameLift 플러그인의 이전 버전에 대한 정보를 제공합니다. 버전 1.0.0 (2021년 출시) 은 아마존 GameLift 서버 SDK 4.x 또는 이전 버전을 사용합니다. 서버 SDK 5.x를 사용하고 GameLift Anywhere Amazon을 지원하는 최신 버전의 플러그인에 대한 설명서는 을 참조하십시오. 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드

이 항목은 백엔드 서비스를 통해 Amazon에서 GameLift 호스팅하는 게임 세션에 연결하도록 게임 클라이언트를 설정하는 데 도움이 됩니다. Amazon GameLift API를 사용하여 매치메이킹을 시작하고 게임 세션 배치를 요청하는 등의 작업을 수행할 수 있습니다.

Amazon 서비스와의 통신을 허용하는 코드를 백엔드 GameLift 서비스 프로젝트에 추가합니다. 백엔드 서비스는 서비스와의 모든 게임 클라이언트 통신을 처리합니다. GameLift 백엔드 서비스에 대한 자세한 내용은 게임 클라이언트 서비스 설계 섹션을 참조하세요.

백엔드 서버는 다음 게임 클라이언트 작업을 처리합니다.

- 플레이어에 맞게 인증을 사용자 지정할 수 있습니다.
- Amazon GameLift 서비스에 활성 게임 세션에 대한 정보를 요청하십시오.
- 새 게임 세션을 생성합니다.
- 기존 게임 세션에 플레이어를 추가합니다.
- 기존 게임 세션에서 플레이어를 제거합니다.

주제

필수 조건

Amazon GameLift 클라이언트와의 게임 서버 통신을 설정하기 전에 다음 작업을 완료하십시오.

- 설정 AWS 계정
- Unity용 플러그인 설치
- GameLift Amazon을 유니티 게임 서버 프로젝트와 통합
- Amazon GameLift 플릿 설정

게임 클라이언트 초기화

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

게임 클라이언트를 초기화하는 코드를 추가합니다. 시작 시 이 코드를 실행합니다. 이 코드는 다른 Amazon GameLift 함수에 필요합니다.

1. `AmazonGameLiftClient`를 초기화합니다. 기본 클라이언트 구성 또는 사용자 지정 구성으로 `AmazonGameLiftClient`를 호출합니다. 클라이언트를 구성하는 방법에 대한 자세한 내용은 백엔드 GameLift 서비스에 Amazon 설정 섹션을 참조하세요.
2. 각 플레이어가 게임 세션에 연결할 수 있도록 고유한 플레이어 ID를 생성합니다. 자세한 내용은 플레이어 ID 생성 섹션을 참조하세요.

   다음 예는 Amazon GameLift 클라이언트를 설정하는 방법을 보여줍니다.

```
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);

    }
}
```

특정 플릿에서 게임 세션 생성

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

코드를 추가하여 배포된 플릿에서 새 게임 세션을 시작하고 플레이어가 사용할 수 있게 허용합니다. GameLift Amazon에서 새 게임 세션을 생성하고 반환한 `GameSession` 후에는 플레이어를 추가할 수 있습니다.

- 새 게임 세션에 대한 요청을 배치합니다.
  - 게임에서 플릿을 사용하는 경우 플릿 또는 별칭 ID, 세션 이름, 게임의 최대 동시 플레이어 수를 포함하여 `CreateGameSession()`을 호출합니다.
  - 게임에서 대기열을 사용하는 경우 `StartGameSessionPlacement()`를 호출합니다.

다음 예제는 게임 세션을 만드는 방법을 보여줍니다.

```
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;
}
```

게임 세션에 플레이어 추가

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

GameLift Amazon에서 새 게임 세션을 생성하고 `GameSession` 객체를 반환한 후 플레이어를 추가할 수 있습니다.

1. 새 플레이어 세션을 생성하여 게임 세션에서 플레이어 슬롯을 예약합니다. `CreatePlayerSession` 또는 게임 세션 ID와 각 플레이어의 고유 ID와 함께 `CreatePlayerSessions`을 사용합니다.
2. 게임 세션에 연결합니다. `PlayerSession` 객체를 검색하여 게임 세션의 연결 정보를 가져옵니다. 이 정보를 사용하여 서버 프로세스에 직접 연결을 설정합니다.

   a. 지정된 포트 및 서버 프로세스의 DNS 이름이나 IP 주소를 사용합니다.

   b. 플릿의 DNS 이름과 포트를 사용합니다. 플릿에 TLS 인증서 생성이 활성화된 경우 DNS 이름과 포트가 필요합니다.

   c. 플레이어 세션 ID를 참조합니다. 게임 서버가 들어오는 플레이어 연결을 검증하는 경우 플레이어 세션 ID가 필요합니다.

다음 예는 게임 세션에서 플레이어 스팟을 예약하는 방법에 대해 보여줍니다.

```
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;
}
```

다음 코드는 플레이어를 게임 세션과 연결하는 방법을 보여줍니다.

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

게임 세션에서 플레이어를 제거합니다.

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

플레이어가 게임을 떠나면 게임 세션에서 플레이어를 제거할 수 있습니다.

1. 플레이어가 서버 프로세스에서 연결이 끊겼음을 Amazon GameLift 서비스에 알립니다. 플레이어의 세션 ID로 RemovePlayerSession을 호출합니다.
2. RemovePlayerSession이 Success를 반환하는지 확인합니다. 그런 다음 Amazon은 플레이어 슬롯을 사용할 수 있도록 GameLift 변경하며, Amazon은 새 플레이어에게 GameLift 할당할 수 있습니다.

다음 예는 플레이어 세션을 제거하는 방법을 보여줍니다.

```
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());
    }
}
```

플러그인 설치 및 설정

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

이 섹션에서는 Unity용 Amazon GameLift 플러그인 버전 1.0.0을 다운로드, 설치 및 설정하는 방법을 설명합니다.

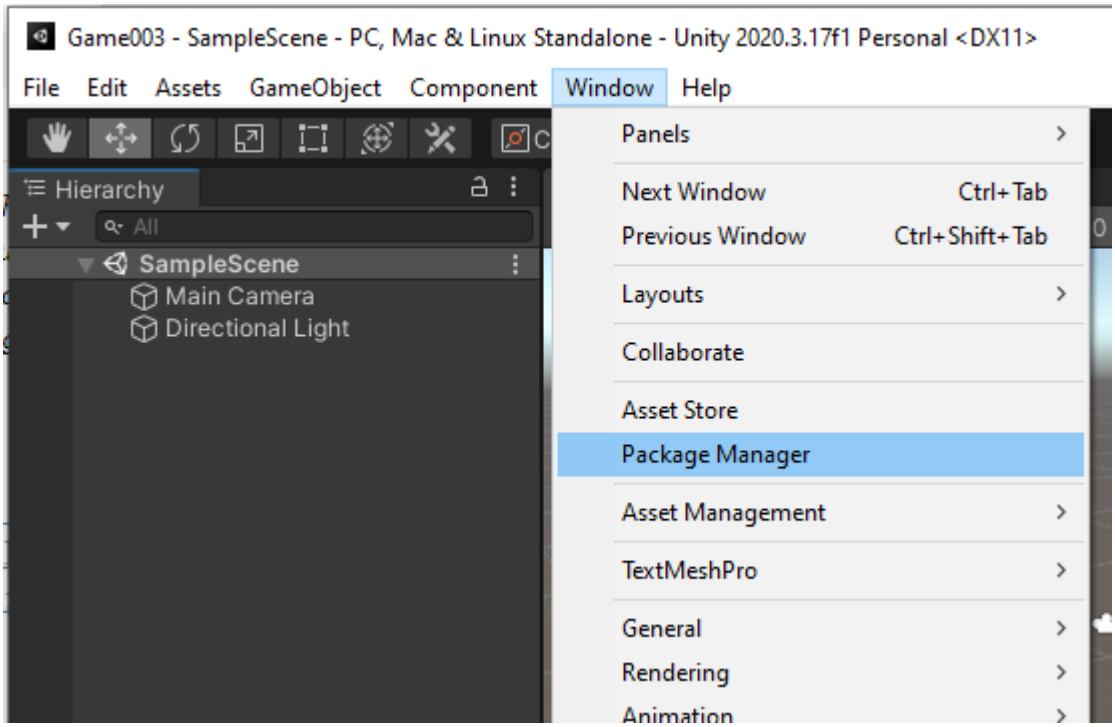
필수 조건

- Windows 2019.4 LTS, Windows 2020.3 LTS,용 Unity 또는 MacOS Unity
- Java의 최신 버전
- .NET 4.x의 현재 버전

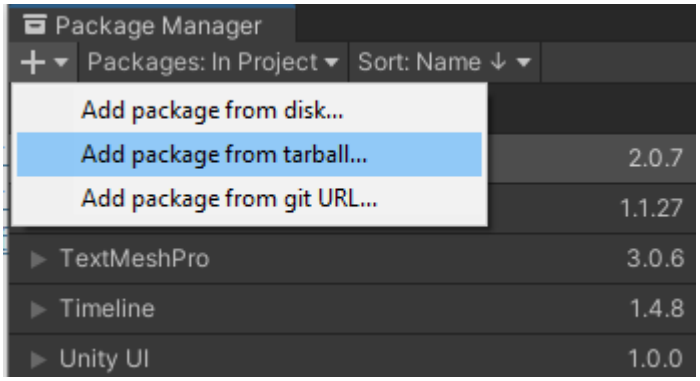
Unity용 플러그인을 다운로드하고 설치하려면

1. Unity용 Amazon GameLift 플러그인을 다운로드하세요. Unity용 Amazon GameLift 플러그인 리포지토리 페이지에서 최신 버전을 찾을 수 있습니다. 최신 릴리스에서 자산을 선택한 다음 `com.amazonaws.gamelift-version.tgz` 파일을 다운로드합니다.

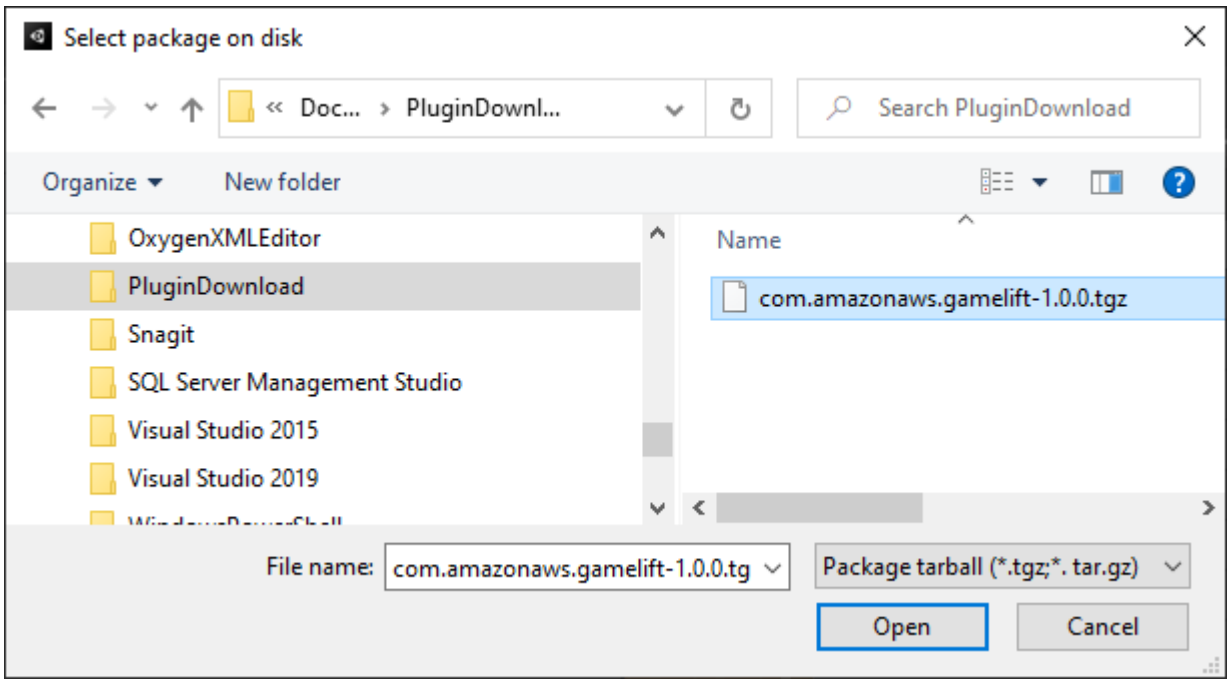
2. Unity를 시작하고 프로젝트를 선택합니다.
3. 상단 탐색 모음의 창에서 패키지 관리자를 선택합니다.



4. 패키지 관리자 탭에서 +를 선택한 다음 tarball에서 패키지 추가...를 선택합니다.



5. 디스크의 패키지 선택 창에서 `com.amazonaws.gamelift` 폴더로 이동하여 `com.amazonaws.gamelift-`*`version`*`.tgz` 파일을 선택한 다음 열기를 선택합니다.



6. Unity가 플러그인을 로드하면 Amazon이 Unity 메뉴에 새 항목으로 GameLift 나타납니다. 스크립트를 설치하고 다시 컴파일하는 데 몇 분 정도 걸릴 수 있습니다. Amazon GameLift 플러그인 설정 탭이 자동으로 열립니다.

7. SDK 창에서 .NET 4.x 사용을 선택합니다.

   구성되면 상태가 구성되지 않음에서 구성됨으로 변경됩니다.

게임을 로컬에서 테스트하세요.

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

Amazon GameLift Local을 사용하여 로컬 GameLift 디바이스에서 Amazon을 실행할 수 있습니다. Amazon GameLift Local을 사용하면 네트워크 연결 없이 몇 초 만에 코드 변경을 확인할 수 있습니다.

로컬 테스트를 구성합니다.

1. Unity용 플러그인 창에서 테스트 탭을 선택합니다.
2. 테스트 창에서 Amazon GameLift Local 다운로드를 선택합니다. Unity용 플러그인은 브라우저 창을 열고 `GameLift_06_03_2021.zip` 파일을 다운로드 폴더로 다운로드합니다.

   다운로드에는 C# Server SDK, .NET 소스 파일 및 Unity와 호환되는.NET 구성 요소가 포함됩니다.
3. 다운로드한 `GameLift_06_03_2021.zip` 파일의 압축을 풉니다.
4. Amazon GameLift 플러그인 설정 창에서 Amazon GameLift Local Path를 선택하고 압축이 풀린 폴더로 이동하여 파일을 **`GameLiftLocal.jar`** 선택한 다음 열기를 선택합니다.

   구성되면 로컬 테스트 상태가 구성되지 않음에서 구성됨으로 변경됩니다.
5. JRE의 상태를 확인합니다. 상태가 구성되지 않음인 경우 JRE 다운로드를 선택하고 권장 Java 버전을 설치합니다.

   Java 환경을 설치하고 구성한 후에는 상태가 구성됨으로 변경됩니다.

로컬 게임을 실행합니다.

1. Unity용 플러그인 탭에서 테스트 탭을 선택합니다.
2. 테스트 창에서 로컬 테스트 UI 열기를 선택합니다.
3. 로컬 테스트 창에서 서버 실행 파일 경로를 지정합니다. ...를 선택하여 서버 애플리케이션의 경로와 실행 파일 이름을 선택합니다.
4. 로컬 테스트 창에서 GL 로컬 포트를 지정합니다.
5. 배포 및 실행을 선택하여 서버를 배포하고 실행합니다.
6. 게임 서버를 중지하려면 중지를 선택하거나 게임 서버 창을 닫습니다.

시나리오 배포

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

시나리오에서는 AWS CloudFormation 템플릿을 사용하여 게임용 클라우드 호스팅 솔루션을 배포하는 데 필요한 리소스를 생성합니다. 이 섹션에서는 Amazon이 GameLift 제공하는 시나리오와 이를 사용하는 방법을 설명합니다.

필수 조건

시나리오를 배포하려면 Amazon GameLift 서비스의 IAM 역할이 필요합니다. Amazon의 역할을 생성하는 방법에 대한 자세한 내용은 GameLift 을 참조하십시오설정 AWS 계정.

각 시나리오에는 다음 리소스에 대한 권한이 필요합니다.

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

시나리오

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

Unity용 Amazon GameLift 플러그인에는 다음과 같은 시나리오가 포함됩니다.

인증 전용

이 시나리오는 게임 서버 기능 없이 플레이어 인증을 수행하는 게임 백엔드 서비스를 생성합니다. 이 템플릿은 계정에 다음 리소스를 생성합니다.

- 플레이어 인증 정보를 저장하는 Amazon Cognito 사용자 풀입니다.
- 게임을 시작하고 게임 연결 정보를 확인할 수 있는 Amazon API Gateway REST 엔드포인트 기반 AWS Lambda 핸들러입니다.

단일 리전 플릿

이 시나리오는 단일 Amazon GameLift 플릿으로 게임 백엔드 서비스를 생성합니다. 다음 리소스를 생성합니다.

- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀입니다.
- 플릿에 열린 플레이어 슬롯이 있는 기존 게임 세션을 검색하는 AWS Lambda 핸들러입니다. 열린 슬롯을 찾을 수 없는 경우 새 게임 세션이 생성됩니다.

대기열 및 사용자 지정 매치메이커가 있는 다중 리전 플릿

이 시나리오는 Amazon GameLift 대기열과 사용자 지정 매치메이커를 사용하여 대기자 명단에서 가장 나이가 많은 플레이어들을 한데 모아 경기를 구성합니다. 다음 리소스를 생성합니다.

- Amazon이 메시지를 GameLift 게시하는 Amazon 단순 알림 서비스 주제. SNS 주제와 알림에 대한 자세한 내용은 게임 세션 배치의 이벤트 알림 설정 섹션을 참조하세요.
- 배치 및 게임 연결 세부 정보를 전달하는 메시지에 의해 호출되는 Lambda 함수입니다.
- 배치 및 게임 연결 세부 정보를 저장하는 Amazon DynamoDB 테이블입니다. `GetGameConnection` 호출은 이 테이블에서 읽은 후 연결 정보를 게임 클라이언트에 반환합니다.

대기열 및 사용자 지정 매치메이커가 있는 스팟 플릿

이 시나리오는 Amazon GameLift 대기열과 사용자 지정 매치메이커를 사용하여 매칭을 구성하고 플릿 3개를 구성합니다. 다음 리소스를 생성합니다.

- 스팟 가용성 손실에 대한 지속성을 제공하기 위해 서로 다른 인스턴스 유형을 포함하는 두 개의 스팟 플릿입니다.
- 다른 스팟 플릿의 백업 역할을 하는 온디맨드 플릿입니다. 플릿 설계에 대한 자세한 내용은 Amazon GameLift 플릿 설계 가이드 섹션을 참조하세요.

- Amazon GameLift 대기열을 통해 서버 가용성은 높이고 비용은 낮게 유지합니다. 대기열에 대한 자세한 내용과 모범 사례에 대해서는 게임 세션 대기열 설계 섹션을 참조하세요.

FlexMatch

이 FlexMatch 시나리오에서는 관리형 매치메이킹 서비스를 사용하여 게임 플레이어를 매칭합니다. 에 대한 FlexMatch 자세한 내용은 Amazon이란 무엇입니까? 를 참조하십시오 GameLift FlexMatch. 이 시나리오에서는 다음 리소스를 생성합니다.

- `StartGame` 요청을 받은 후 매치메이킹 티켓을 생성하는 Lambda 함수입니다.
- 일치 이벤트를 FlexMatch 수신하기 위한 별도의 Lambda 함수.

AWS 계정에 불필요한 요금이 부과되지 않도록 하려면 각 시나리오에서 생성된 리소스를 모두 사용한 후에 제거합니다. 해당 AWS CloudFormation 스택을 삭제합니다.

자격 증명 업데이트 AWS

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

Unity용 Amazon GameLift 플러그인을 사용하려면 시나리오를 배포하려면 보안 자격 증명이 필요합니다. 새 자격 증명을 생성하거나 기존 자격 증명을 사용할 수 있습니다.

자격 증명 구성에 대한 자세한 내용은 AWS 자격 증명의 이해 및 획득을 참조하세요.

AWS 자격 증명을 업데이트하려면

1. Unity용 플러그인 탭에서 배포 탭을 선택합니다.
2. 배포 창에서 AWS 자격 증명을 선택합니다.
3. 새 AWS 자격 증명을 생성하거나 기존 자격 증명을 선택할 수 있습니다.
   - 자격 증명을 생성하려면 새 자격 증명 프로필 생성을 선택한 다음 새 프로필 이름, AWS 액세스 키 ID, AWS 보안 키 및 AWS 리전을 지정합니다.
   - 기존 자격 증명을 선택하려면 기존 자격 증명 프로필 선택을 선택한 다음 프로필 이름 및 AWS 리전을 선택합니다.
4. AWS 자격 증명 업데이트 창에서 자격 증명 프로필 업데이트를 선택합니다.

계정 부트스트랩 업데이트

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

부트스트랩 위치는 배포 중에 사용되는 Amazon S3 버킷입니다. 게임 서버 애셋 및 기타 종속 항목을 저장하는 데 사용됩니다. 버킷에 대해 선택한 AWS 리전은 시나리오 배포에 사용할 리전과 동일해야 합니다.

Amazon S3 버킷에 대한 자세한 내용은 Amazon Simple Storage Service 버킷 생성, 구성, 작업을 참조하세요.

계정 부트스트랩 위치를 업데이트하려면

1. Unity용 플러그인 탭에서 배포 탭을 선택합니다.
2. 배포 창에서 계정 부트스트랩 업데이트를 선택합니다.
3. 계정 부트스트래핑 창에서 기존 Amazon S3 버킷을 선택하거나 새 Amazon S3 버킷을 생성합니다.
   - 기존 버킷을 선택하려면 기존 Amazon S3 버킷 선택 및 업데이트를 선택하여 선택 사항을 저장합니다.
   - 새 Amazon S3 버킷 생성을 선택하여 새 Amazon Simple Storage Service 버킷을 생성한 다음 정책을 선택합니다. 이 정책은 Amazon S3 버킷의 만료 시기를 지정합니다. 생성을 선택하여 버킷을 생성합니다.

게임 시나리오 배포

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

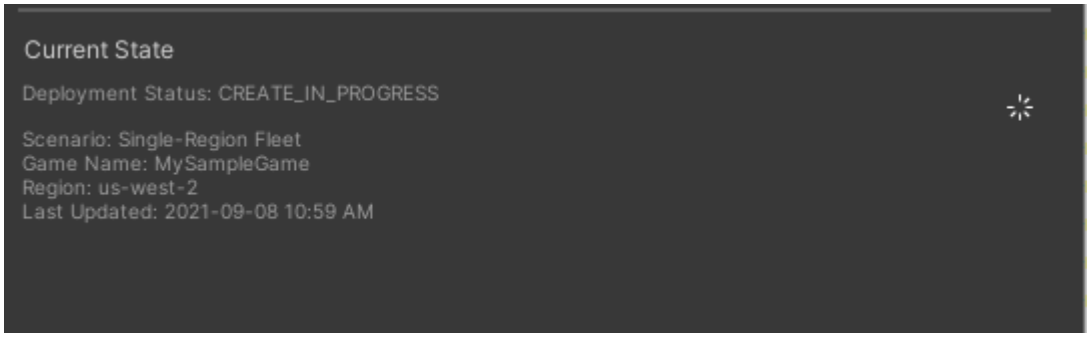
시나리오를 사용하여 Amazon에서 게임을 테스트할 수 GameLift 있습니다. 각 시나리오는 AWS CloudFormation 템플릿을 사용하여 필요한 리소스로 스택을 생성합니다. 대부분의 시나리오에는 게임

서버 실행 파일과 빌드 경로가 필요합니다. 시나리오를 배포하면 Amazon은 배포의 일환으로 게임 자산을 부트스트랩 위치에 GameLift 복사합니다.

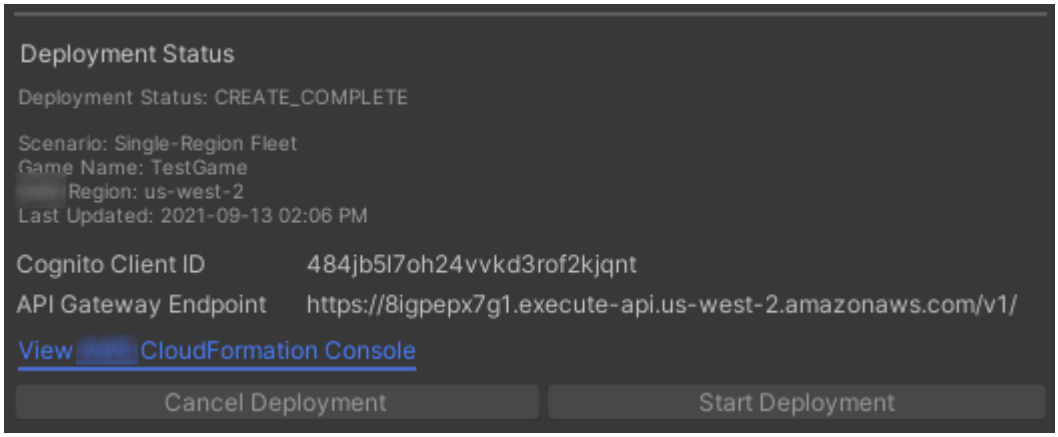
시나리오를 배포하려면 AWS 자격 증명과 AWS 계정 부트스트랩을 구성해야 합니다.

시나리오를 배포하려면

1. Unity용 플러그인 탭에서 배포 탭을 선택합니다.
2. 배포 창에서 배포 UI 열기를 선택합니다.
3. 배포 창에서 시나리오를 선택합니다.
4. 게임 이름을 입력합니다. 이름은 고유해야 합니다. 게임 이름은 시나리오를 배포할 때 AWS CloudFormation 스택 이름의 일부입니다.
5. 게임 서버 빌드 폴더 경로를 선택합니다. 빌드 폴더 경로는 서버 실행 파일 및 종속 파일이 들어 있는 폴더를 가리킵니다.
6. 게임 서버 빌드 .exe 파일 경로를 선택합니다. 빌드 실행 파일 경로는 게임 서버 실행 파일을 가리킵니다.
7. 배포 시작을 선택하여 시나리오 배포를 시작합니다. 배포 창의 현재 상태에서 업데이트 상태를 확인할 수 있습니다. 시나리오를 배포하는 데 최대 30분이 걸릴 수 있습니다.



8. 시나리오 배포가 완료되면 현재 상태가 업데이트되어 게임에 복사하여 붙여넣을 수 있는 Cognito 클라이언트 ID 및 API Gateway 엔드포인트가 포함됩니다.



9. 게임 설정을 업데이트하려면 Unity 메뉴에서 클라이언트 연결 설정으로 이동을 선택합니다. 그러면 Unity 화면 오른쪽에 Inspector 탭이 표시됩니다.
10. 로컬 테스트 모드를 선택 해제합니다.
11. API Gateway 엔드포인트 및 Coginito 클라이언트 ID를 입력합니다. 시나리오 배포에 사용한 것과 동일한 AWS 리전을 선택합니다. 그런 다음 배포된 시나리오 리소스를 사용하여 게임 클라이언트를 다시 빌드하고 실행할 수 있습니다.

시나리오에서 생성된 리소스 삭제

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

시나리오용으로 생성된 리소스를 삭제하려면 해당 AWS CloudFormation 스택을 삭제합니다.

시나리오에서 생성된 리소스를 삭제하려면

1. Unity용 Amazon GameLift 플러그인 배포 창에서 AWS CloudFormation콘솔 보기를 선택하여 AWS CloudFormation 콘솔을 엽니다.
2. AWS CloudFormation 콘솔에서 스택을 선택한 다음 배포 중에 지정된 게임 이름이 포함된 스택을 선택합니다.
3. 스택을 삭제하려면 삭제를 선택합니다. 스택을 삭제하는 데 몇 분 정도 걸립니다. AWS CloudFormation이 시나리오에서 사용하는 스택을 삭제하면 상태가 `ROLLBACK_COMPLETE`로 바뀝니다.

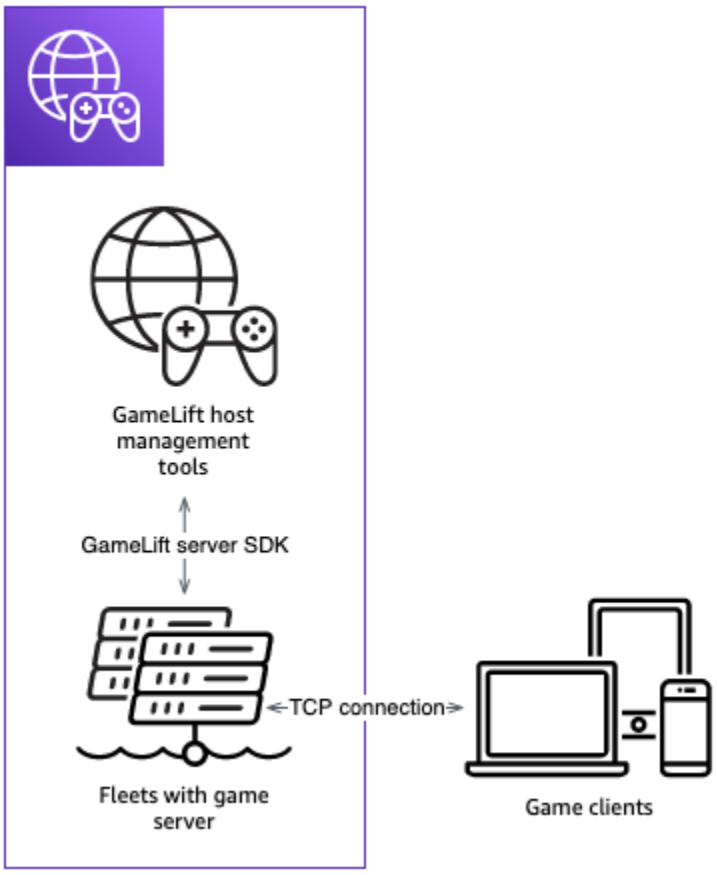
GameLift Unity에서 게임을 Amazon과 통합

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

다음 작업을 GameLift 완료하여 Unity 게임을 Amazon과 통합하십시오.

- GameLift Amazon을 유니티 게임 서버 프로젝트와 통합
- GameLift Amazon을 Unity 게임 클라이언트 프로젝트와 통합

다음 다이어그램은 게임을 통합하는 흐름의 예를 보여 줍니다. 다이어그램에서는 게임 서버가 있는 플릿이 Amazon에 배포되어 GameLift 있습니다. 게임 클라이언트는 Amazon과 통신하는 게임 서버와 통신합니다. GameLift



샘플 게임 임포트 및 실행

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

Unity용 Amazon GameLift 플러그인에는 게임을 Amazon과 통합하는 데 필요한 기본 사항을 탐색하는 데 사용할 수 있는 샘플 게임이 포함되어 있습니다. GameLift 이 섹션에서는 게임 클라이언트와 게임 서버를 빌드한 다음 Amazon GameLift Local을 사용하여 로컬에서 테스트합니다.

필수 조건

- 설정 AWS 계정
- 플러그인 설치 및 설정

샘플 게임 서버 빌드 및 실행

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

샘플 게임의 게임 서버 파일을 설정합니다.

1. Unity의 메뉴에서 GameLiftAmazon을 선택한 다음 샘플 게임 가져오기를 선택합니다.
2. 샘플 게임 가져오기 창에서 가져오기를 선택하여 게임, 애셋, 종속 항목을 가져옵니다.
3. 게임 서버를 빌드합니다. Unity의 메뉴에서 GameLiftAmazon을 선택한 다음 Windows 샘플 서버 빌드 설정 적용 또는 macOS 샘플 서버 빌드 설정 적용을 선택합니다. 게임 서버 설정을 구성한 후 Unity는 애셋을 다시 컴파일합니다.
4. Unity의 메뉴에서 파일을 선택한 다음 빌드를 선택합니다. 서버 빌드를 선택하고 빌드를 선택한 다음 서버 파일용 빌드 폴더를 선택합니다.

   Unity는 샘플 게임 서버를 빌드하고 실행 파일과 필수 애셋을 지정된 빌드 폴더에 저장합니다.

샘플 게임 클라이언트 빌드 및 실행

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

샘플 게임의 게임 클라이언트 파일을 설정합니다.

1. Unity의 메뉴에서 GameLiftAmazon을 선택한 다음 Windows 샘플 클라이언트 빌드 설정 적용 또는 macOS 샘플 클라이언트 빌드 설정 적용을 선택합니다. 게임 클라이언트 설정이 구성된 후 Unity는 애셋을 다시 컴파일합니다.
2. Unity의 메뉴에서 클라이언트 설정으로 이동을 선택합니다. 그러면 Unity 화면 오른쪽에 Inspector 탭이 표시됩니다. Amazon GameLift 클라이언트 설정 탭에서 로컬 테스트 모드를 선택합니다.
3. 게임 클라이언트를 빌드합니다. Unity의 메뉴에서 파일을 선택합니다. 서버 빌드가 선택하지 않았는지 확인하고 빌드를 선택한 다음, 클라이언트 파일용 빌드 폴더를 선택합니다.

   Unity는 샘플 게임 클라이언트를 빌드하고 실행 파일과 필수 애셋을 지정된 클라이언트 빌드 폴더에 저장합니다.
4. 게임 서버와 클라이언트는 아직 빌드하지 않았습니다. 다음 단계에서는 게임을 실행하고 GameLift Amazon과 어떻게 상호 작용하는지 확인합니다.

로컬에서 샘플 게임 테스트

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

Amazon GameLift Local을 사용하여 가져온 샘플 게임을 실행합니다.

1. 게임 서버를 시작합니다. Unity용 플러그인 탭에서 배포 탭을 선택합니다.
2. 테스트 창에서 로컬 테스트 UI 열기를 선택합니다.
3. 로컬 테스트 창에서 게임 서버 .exe 파일 경로를 지정합니다. 경로에는 실행 파일 이름을 포함해야 합니다. 예를 들어 `C:/MyGame/GameServer/MyGameServer.exe`입니다.

4. 배포 및 실행을 선택합니다. Unity용 플러그인은 게임 서버를 시작하고 Amazon GameLift Local 로그 창을 엽니다. 창에는 게임 서버와 Amazon GameLift Local 간에 전송된 메시지를 포함한 로그 메시지가 들어 있습니다.
5. 게임 클라이언트를 실행합니다. 샘플 게임 클라이언트로 빌드 위치를 찾고 실행 파일을 선택합니다.
6. Amazon GameLift 샘플 게임에서 이메일과 비밀번호를 입력한 다음 로그인을 선택합니다. 이메일과 암호는 검증되거나 사용되지 않습니다.
7. Amazon GameLift 샘플 게임에서 [Start] 를 선택합니다. 게임 클라이언트가 게임 세션을 찾습니다. 세션을 찾을 수 없는 경우 세션을 생성합니다. 그러면 게임 클라이언트가 게임 세션을 시작합니다. 로그에서 게임 활동을 볼 수 있습니다.

샘플 게임 서버 로그

```
...
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
```

샘플 아마존 GameLift 로컬 로그

```
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"
```

서버 프로세스 종료

> **Note**
>
> 이 주제에서는 서버 SDK 4.x 또는 이전 버전을 사용하는 Unity 버전 1.0.0용 Amazon GameLift 플러그인을 참조합니다.

샘플 게임을 완료한 후 Unity에서 서버를 종료합니다.

1. 게임 클라이언트에서 종료를 선택하거나 창을 닫아 게임 클라이언트를 중지합니다.
2. Unity의 로컬 테스트 창에서 중지를 선택하거나 게임 서버 창을 닫아 서버를 중지합니다.

언리얼 엔진용 Amazon GameLift 플러그인과 게임 통합

이 섹션의 항목에서는 언리얼 엔진 (UE) 용 Amazon GameLift 플러그인과 이를 사용하여 GameLift Amazon에서 호스팅할 멀티플레이어 게임 프로젝트를 준비하는 방법을 설명합니다. 플러그인의 안내 워크플로를 사용하여 UE 개발 환경에서 전적으로 작업하여 Amazon에서 호스팅하기 위한 기본 요구 사항을 GameLift 완료하십시오.

GameLift Amazon은 게임 개발자가 세션 기반 멀티플레이어 게임 전용 게임 서버를 관리하고 확장할 수 있는 완전 관리형 서비스입니다. Amazon GameLift 호스팅에 대한 자세한 내용은 을 참조하십시오Amazon GameLift 작동 방식.

주제

- 플러그인 정보
- 플러그인 워크플로
- Unreal용 플러그인 설치
- AWS 사용자 프로필 설정
- Amazon에서 테스트할 게임을 설정합니다. GameLift Anywhere
- 관리형 EC2 플릿을 사용하여 클라우드 호스팅에 게임을 배포합니다.

플러그인 정보

플러그인은 Amazon GameLift 도구 및 기능을 UE 편집기에 추가합니다. 플러그인의 가이드 워크플로는 Amazon을 게임 프로젝트에 GameLift 통합하고, 워크스테이션을 테스트용 로컬 호스트로 지정하고, 게임 서버를 Amazon GameLift 클라우드 호스팅에 배포합니다.

플러그인의 사전 구축된 호스팅 솔루션을 사용하여 게임을 배포합니다. 로컬 워크스테이션을 호스트로 사용하여 Amazon GameLift Anywhere 플릿을 설정하십시오. 클라우드 호스팅의 경우 플레이어 지연 시간, 게임 세션 가용성 및 비용의 균형을 서로 다른 방식으로 맞추는 두 가지 일반적인 배포 시나리오 중에서 선택하십시오. 한 가지 시나리오에는 간단한 FlexMatch 매치메이커와 규칙 세트가 포함됩니다. 이러한 솔루션을 사용하면 프로덕션 환경에 바로 사용할 수 있는 호스팅 구조로 빠르게 시작한 다음 필요에 따라 최적화하고 사용자 지정할 수 있습니다.

플러그인에는 다음과 같은 구성 요소가 포함됩니다.

- UE 편집기용 플러그인 모듈. 플러그인이 설치되면 새 메인 메뉴 버튼을 통해 Amazon GameLift 기능에 액세스할 수 있습니다.

- 클라이언트 측 기능을 갖춘 Amazon GameLift 서비스 API용 C++ 라이브러리.
- Amazon GameLift 서버 SDK용 언리얼 라이브러리 (버전 5)
- 서버 통합 테스트에 사용할 기본 블루프린트와 UI 요소가 포함된 스타트업 게임 맵과 테스트 맵 2개를 포함한 테스트용 콘텐츠.
- 호스팅용 게임 서버를 배포할 때 플러그인이 사용하는 AWS CloudFormation 템플릿 형태의 편집 가능한 구성.

플러그인 워크플로

다음 단계는 언리얼 엔진용 Amazon GameLift 플러그인과 게임 프로젝트를 통합하고 배포하는 일반적인 접근 방식을 설명합니다. UE 편집기와 게임 코드에서 작업하여 이 단계를 완료합니다.

1. 계정에 연결되고 Amazon을 사용할 권한이 있는 유효한 AWS 계정 사용자의 액세스 자격 증명을 제공하는 사용자 프로필을 GameLift 생성하십시오.
2. 게임 프로젝트에 서버 코드를 추가하여 실행 중인 게임 서버와 with Amazon GameLift 서비스 간의 통신을 설정합니다.
3. 게임 클라이언트가 Amazon에 요청을 보내 새 게임 세션을 시작한 다음 GameLift 연결하도록 하는 클라이언트 코드를 게임 프로젝트에 추가합니다.
4. Anywhere 워크플로를 사용하여 로컬 워크스테이션을 게임 서버의 Anywhere 호스트로 설정합니다. 플러그인을 통해 게임 서버와 클라이언트를 로컬로 시작하고, 게임 세션에 연결하며, 통합을 테스트합니다.
5. EC2 호스팅 워크플로를 사용하여 통합된 게임 서버를 업로드하고 클라우드 호스팅 솔루션을 배포합니다. 게임 서버가 준비되면 플러그인을 통해 게임 클라이언트를 로컬로 시작하고 게임 세션에 연결하여 게임을 플레이합니다.

플러그인에서 작업할 때는 AWS 리소스를 생성하고 사용하게 됩니다. 이러한 작업을 수행하면 사용 중인 AWS 계정에 요금이 부과될 수 있습니다. 처음 사용하는 AWS 경우 이러한 작업은 AWS프리 티어에 포함될 수 있습니다.

Unreal용 플러그인 설치

이 섹션에서는 Unreal Engine 프로젝트에 플러그인을 추가하기 위한 초기 설치 작업을 설명합니다. 플러그인 기능은 Unreal 편집기에서 프로젝트를 열었을 때 사용할 수 있습니다.

> **Note**
>
> Amazon GameLift 플러그인을 표준 버전의 UE 에디터와 함께 사용할 수 있지만, 게임 서버 빌드를 패키징할 때는 소스 빌드 버전을 사용해야 합니다.

시작하기 전에

언리얼 엔진용 Amazon GameLift 플러그인을 사용하는 데 필요한 것은 다음과 같습니다.

- 언리얼 엔진 출시 패키지용 Amazon GameLift 플러그인. [다운로드 사이트].
- Microsoft Visual Studio 2019 이상 최신 버전.
- Unreal Engine 편집기의 소스 빌드 버전입니다. 멀티플레이어 게임용 서버 구성 요소를 패키징하려면 소스 빌드 버전이 필요합니다. 추가 사전 요구 사항을 포함한 자세한 내용은 Unreal Engine 문서를 참조하세요.
  - GitHub You Need GitHub 및 에픽게임즈 계정에서 언리얼 엔진 소스 코드에 액세스할 수 있습니다.
  - 소스에서 Unreal Engine 빌드 자습서.
- C++ 게임 코드를 사용하는 멀티플레이어 게임 프로젝트. 블루프린트 프로젝트로 작업하는 경우 프로젝트의 C++ 소스 코드 생성 방법에 대한 언리얼 문서를 참조하십시오.

플러그인을 게임 프로젝트에 추가

다음 작업을 완료하여 게임 프로젝트에 플러그인을 추가하세요.

아마존 GameLift C++ 서버 SDK 구축

1. 언리얼 엔진용 Amazon GameLift 플러그인 릴리스 패키지를 압축 해제하여 zip 파일 두 개를 추출합니다.

   - `amazon-gamelift-plugin-unreal-<>-sdk-<>.zip`
   - `GameLift-Cpp-ServerSDK-<>.zip.`

   이들 파일의 압축을 풉니다.
2. `GameLift-Cpp-ServerSDK-<>`폴더를 열고 사용 중인 플랫폼에 맞는 다음 지침을 완료하십시오: 리눅스 또는 Microsoft Windows.

Linux

1. 다음 명령을 실행합니다.

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

   이 명령은 /lib/aws-cpp-sdk-gamelift-server.so 파일을 빌드합니다.

2. /lib/aws-cpp-sdk-gamelift-server.soamazon-gamelift-plugin-unreal/GameLiftPlugin/Source/GameliftServer/ThirdParty/GameLiftServerSDK/Linux/x86_64-unknown-linux-gnu/디렉터리에 복사합니다.

Microsoft Windows

1. 다음 명령을 실행합니다.

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

   이 명령은 다음과 같은 바이너리 파일을 빌드합니다.

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

2. 파일을 amazon-gamelift-plugin-unreal\GameLiftPlugin\Source\GameliftServer\ThirdParty\GameLiftServerSDK\Win64\ 디렉터리에 복사합니다.

게임 프로젝트 파일에서 작업하면서 다음 작업을 완료합니다.

1. 플러그인 파일을 설치합니다.

   a. 게임 프로젝트 루트 폴더(예: ... > Unreal Projects/[project-name]/)를 찾습니다. Plugins 폴더가 없으면 새로 만드세요.

b. 압축을 푼 amazon-gamelift-plugin-unreal 폴더로 이동합니다. amazon-gamelift-plugin-unreal-<>-sdk-<>.zip GameLiftPlugin폴더의 폴더를 게임 프로젝트 gamelift-plugin-unreal 디렉터리의 Plugins 폴더로 복사합니다.

2. **.uproject** 파일에 플러그인을 추가합니다.

a. 게임 프로젝트 루트 폴더에서 .uproject 파일을 엽니다.

b. 파일을 업데이트하여 Plugins 섹션에 "GameLiftPlugin" 및 WebBrowserWidget ""를 추가하고 활성화하십시오. 다음 코드는 MyGame ""라는 게임의 업데이트된 .uproject 파일을 보여줍니다.

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

3. 프로젝트의 UE 편집기 버전을 변경합니다.

한 편집기 버전용 프로젝트를 만들고 이제 다른 버전(예: 소스 빌드 버전)으로 변경하려면 프로젝트를 업데이트해야 합니다.

게임 프로젝트 루트 폴더에서 .uproject 파일을 선택하고 Unreal Engine 버전 전환 옵션을 선택합니다. 새 편집기 버전을 선택합니다.

4. 업데이트한 내용으로 프로젝트 솔루션을 다시 빌드합니다.

a. 프로젝트 루트 폴더에서 솔루션(`*.sln`) 파일을 찾습니다. 파일이 없는 경우 `.uproject` 파일을 선택하고 Visual Studio 프로젝트 파일 생성 옵션을 선택합니다.

b. 솔루션 파일을 열고 프로젝트를 빌드하거나 다시 빌드합니다.

5. UE 편집기에서 플러그인이 활성화되어 있는지 확인합니다.

> **Note**
>
> 이미 편집기가 열려 있는 경우, 편집기가 새 플러그인을 인식하기 전에 편집기를 다시 시작해야 할 수 있습니다.

a. 선택한 UE 편집기에서 프로젝트를 엽니다.

b. 기본 편집기 도구 모음에서 새 Amazon GameLift 메뉴 버튼 [이미지 필요] 를 확인하십시오.

c. 콘텐츠 브라우저에서 Amazon GameLift 플러그인 애셋을 찾아보세요. 보기 옵션 설정에 플러그인 콘텐츠 표시 옵션이 선택되어 있는지 확인합니다.

AWS 사용자 프로필 설정

플러그인을 설치한 후 프로필을 설정하고 유효한 AWS 계정 사용자에 연결합니다. 프로파일을 여러 개 유지할 수 있지만 한 번에 한 프로필만 활성화할 수 있습니다. 플러그인에서 작업할 때마다 사용할 프로필을 선택합니다.

프로필을 여러 개 유지하면 서로 다른 호스팅 시나리오 간에 전환할 수 있습니다. 예를 들어 AWS 자격 증명은 같지만 다른 AWS 리전인 프로필을 설정할 수 있습니다. 또는 다른 AWS 계정이나 다른 사용자/권한 집합으로 프로필을 설정할 수도 있습니다.

> **Note**
>
> 워크스테이션에 AWS CLI를 설치했고 프로필이 이미 구성되어 있는 경우 Amazon GameLift 플러그인이 이를 감지하여 기존 프로필로 나열합니다. 플러그인은 `[default]` 이름이 지정된 프로필을 자동으로 선택합니다. 기존의 데이터 세트를 사용하거나 새 프로필을 생성할 수 있습니다.

AWS 프로필을 관리하려면

1. 언리얼 에디터 메인 툴바에서 Amazon GameLift 메뉴를 선택하고 Set User Profiles (AWS사용자 프로필 설정) 을 선택합니다. 이 작업을 수행하여 플러그인의 프로젝트 설정을 엽니다. AWS 사용자 프로필 섹션을 확장합니다.
2. 플러그인이 기존 프로필을 감지하지 못하면 프로필을 만들라는 메시지가 표시됩니다. 새 계정이나 기존 AWS 계정을 사용하여 새 프로필을 만들 수 있습니다.

   > **Note**
   >
   > AWS Management Console을 사용하여 새 AWS 계정을 생성하고 적절한 권한 설정으로 사용자를 생성하거나 업데이트해야 합니다.

   프로필을 설정할 때는 다음 정보가 필요합니다.

   - AWS 계정. 새 AWS 계정을 생성해야 하는 경우 메시지에 따라 계정을 생성합니다. 자세한 내용은 AWS 계정 생성을 참조하세요.
   - Amazon GameLift 및 기타 필수 AWS 서비스를 사용할 권한이 있는 AWS 사용자 Amazon GameLift 권한 및 장기 자격 증명을 통한 프로그래밍 액세스 권한을 가진 AWS Identity and Access Management (IAM) 사용자를 설정하는 방법에 설정 AWS 계정 대한 지침은 을 참조하십시오.
   - AWS 사용자의 자격 증명. 이러한 자격 증명은 AWS 액세스 키와 AWS 보안 키로 구성되어 있습니다. 자세한 내용은 액세스 키 가져오기를 참조하세요.
   - AWS 리전. 이는 호스팅용 AWS 리소스를 생성하려는 지리적 위치입니다. 개발 중에는 실제 위치와 가까운 리전을 사용하는 것이 좋습니다. 지원되는 AWS 리전 목록을 참조하세요.
3. 플러그인이 기존 프로필을 감지하는 경우 프로필을 만들라는 메시지가 표시되지 않습니다. 프로필을 업데이트하거나 새 프로필을 만들려면 프로필 관리를 선택합니다.

프로필을 부트스트랩하려면

Amazon GameLift 플러그인과 함께 사용하려면 모든 프로필을 부트스트랩해야 합니다. 부트스트래핑은 프로필과 관련된 Amazon S3 버킷을 생성합니다. 프로젝트 구성, 빌드 아티팩트 및 기타 종속성을 저장하는 데 사용됩니다. 버킷은 다른 프로필 간에 공유되지 않습니다.

부트스트래핑에는 새 AWS 리소스 생성이 포함되며 비용이 발생할 수 있습니다.

1. 언리얼 에디터 메인 툴바에서 Amazon GameLift 아이콘을 선택하고 Set User Profiles (AWS사용자 프로필 설정) 을 선택합니다. 이 작업을 수행하여 플러그인의 프로젝트 설정을 엽니다. AWS 사용자 프로필 섹션을 확장합니다.
2. 프로필 부트스트랩 섹션의 드롭다운 목록에서 프로필을 선택하고 부트스트랩 상태를 확인합니다. 상태가 버킷이 없는 것으로 표시되면 프로필 부트스트랩 및 생성 버튼을 선택하여 선택한 프로필에 대한 Amazon S3 버킷을 생성합니다.
3. 부트스트랩 상태가 “활성”으로 변경될 때까지 기다립니다. 몇 분 정도 소요될 수 있습니다.

Amazon에서 테스트할 게임을 설정합니다. GameLift Anywhere

이 워크플로우에서는 Amazon GameLift 기능을 위한 클라이언트 및 서버 게임 코드를 추가하고 플러그인을 사용하여 로컬 워크스테이션을 테스트 게임 서버 호스트로 지정합니다. 통합 작업을 완료하면 플러그인을 사용하여 게임 클라이언트 및 서버 구성 요소를 빌드합니다.

Amazon GameLift Anywhere 워크플로를 시작하려면:

- 언리얼 에디터 메인 툴바에서 Amazon GameLift 메뉴를 선택하고 Host with Anywhere를 선택합니다. 이 작업을 수행하면 Anywhere 배포 플러그인 페이지가 열리고 게임 구성 요소를 통합, 빌드 및 시작하는 6단계 프로세스를 제공합니다.

1단계: 프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필 AWS 계정과 연결되며 프로필의 기본 AWS 지역에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

1. 사용 가능한 프로필 드롭다운 목록에서 프로필을 선택합니다. 아직 프로필이 없거나 새 프로필을 만들려면 Amazon GameLift 메뉴로 이동하여 AWS사용자 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 “활성”이 아닌 경우, Bootstrap 프로필을 선택하고 상태가 “활성”으로 변경될 때까지 기다리십시오.

2단계: 게임 코드 설정

이 단계에서는 클라이언트와 서버 코드에 대한 일련의 업데이트를 수행하여 호스팅 기능을 추가합니다. 언리얼 에디터의 소스 빌드 버전을 아직 설정하지 않은 경우 플러그인에서 지침과 소스 코드에 대한 링크를 제공합니다.

플러그인을 사용하면 게임 코드를 통합할 때 몇 가지 편리함을 활용할 수 있습니다. 최소한의 통합을 통해 기본 호스팅 기능을 설정할 수 있습니다. 더 광범위한 사용자 지정 통합을 수행할 수도 있습니다. 이 섹션의 정보는 최소 통합 옵션에 대해 설명합니다. 플러그인에 포함된 테스트 맵을 사용하여 게임 프로젝트에 클라이언트 Amazon GameLift 기능을 추가할 수 있습니다. 서버 통합의 경우 제공된 코드 샘플을 사용하여 프로젝트의 게임 모드를 업데이트합니다.

서버 게임 모드 통합

게임 서버와 Amazon GameLift 서비스 간의 통신을 지원하는 서버 코드를 게임에 추가합니다. 게임 서버는 새 게임 세션 시작과 같은 GameLift Amazon의 요청에 응답하고 게임 서버 상태 및 플레이어 연결 상태를 보고할 수 있어야 합니다.

1. 코드 편집기에서 일반적으로 프로젝트 루트 폴더에 있는 게임 프로젝트의 솔루션(`.sln`) 파일을 엽니다. 예를 들면 `GameLiftUnrealApp.sln`입니다.
2. 솔루션이 열린 상태에서 프로젝트 게임 모드 헤더 파일인 `[project-name]GameMode.h` 파일을 찾습니다. 예를 들면 `GameLiftUnrealAppGameMode.h`입니다.
3. 헤더 파일을 다음 예제 코드에 맞게 변경합니다. 반드시 GameLiftServer ""를 자신의 프로젝트 이름으로 바꾸십시오. 이러한 업데이트는 게임 서버에만 적용되므로 클라이언트에서 사용할 수 있도록 원본 게임 모드 파일의 백업 복사본을 만드는 것이 좋습니다.

```
// 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. 관련 소스 파일인 [project-name]GameMode.cpp 파일(예: GameLiftUnrealAppGameMode.cpp)을 엽니다. 코드를 다음 예제 코드에 맞게 변경합니다. 반드시 GameLiftUnrealApp ""를 자신의 프로젝트 이름으로 바꾸십시오. 이러한 업데이트는 게임 서버에만 적용되므로 클라이언트에서 사용할 수 있도록 원본 파일의 백업 복사본을 만드는 것이 좋습니다.

   다음 예제 코드는 Amazon과의 서버 통합에 필요한 최소 요소를 추가하는 방법을 보여줍니다 GameLift.

   - Amazon GameLift API 클라이언트를 초기화합니다. Amazon GameLift Anywhere 플릿에는 서버 파라미터를 사용한 InitSDK() 호출이 필요합니다. Anywhere 플릿에 연결하면 플러그인은 서버 파라미터를 콘솔 인수로 저장합니다. 샘플 코드는 런타임 시 값에 액세스할 수 있습니다.
   - OnStartGameSessionOnProcessTerminate, 및 onHealthCheck 등 Amazon GameLift 서비스의 요청에 응답하는 데 필요한 콜백 함수를 구현하십시오.
   - 게임 세션을 호스팅할 준비가 되면 지정된 ProcessReady() 포트로 전화하여 Amazon GameLift 서비스에 알리십시오.

```
// 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
}
```

클라이언트 게임 맵 통합

스타트업 게임 맵에는 게임 세션을 요청하고 연결 정보를 사용하여 게임 세션에 연결하는 기본 코드가 이미 포함되어 있는 블루프린트 로직과 UI 요소가 포함되어 있습니다. 맵을 그대로 사용하거나 필요에 따라 수정할 수 있습니다. 스타트업 게임 맵을 다른 게임 애셋(예: Unreal Engine에서 제공하는 3인칭 템플릿 프로젝트)과 함께 사용합니다. 이러한 애셋은 콘텐츠 브라우저에서 사용할 수 있습니다. 이를 사용하여 플러그인의 배포 워크플로를 테스트하거나 게임용 사용자 지정 백엔드 서비스를 생성하기 위한 가이드로 사용할 수 있습니다.

스타트업 맵에는 다음과 같은 특성이 있습니다.

- 여기에는 Anywhere 플릿과 관리형 EC2 플릿 모두에 대한 로직이 포함됩니다. 클라이언트를 실행할 때 두 플릿 중 하나에 연결하도록 선택할 수 있습니다.
- 클라이언트 기능에는 게임 세션 찾기 (SearchGameSessions()), 새 게임 세션 생성 (CreateGameSession()), 게임 세션에 직접 참여하기 등이 포함됩니다.
- 프로젝트의 Amazon Cognito 사용자 풀(배포된 Anywhere 솔루션의 일부)에서 고유한 플레이어 ID를 가져옵니다.

스타트업 게임 맵을 사용하려면

1. UE 편집기에서 프로젝트 설정, 맵 및 모드 페이지를 열고 기본 맵 섹션을 확장합니다.
2. 에디터 스타트업 맵의 경우 드롭다운 목록에서 StartupMap ""를 선택합니다. `... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Plugin Content/Maps`에 있는 파일을 검색해야 할 수도 있습니다.
3. 게임 디폴트 맵의 경우 드롭다운 목록에서 동일한 StartupMap ""를 선택합니다.
4. 서버 기본 맵의 경우 ThirdPersonMap ""를 선택합니다. 이는 게임 프로젝트에 포함된 기본 맵입니다. 이 맵은 게임에 참여하는 두 명의 플레이어를 위해 설계되었습니다.
5. 서버 기본 맵의 세부 정보 패널을 엽니다. GameMode 오버라이드를 “없음”으로 설정합니다.
6. 기본 모드 섹션을 확장하고 전역 기본 서버 게임 모드를 서버 통합을 위해 업데이트한 게임 모드로 설정합니다.

프로젝트를 이렇게 변경했으면 게임 컴포넌트를 빌드할 준비가 된 것입니다.

게임 구성 요소 구축

1. 새 서버 및 클라이언트 대상 파일 생성

   a. 게임 프로젝트 폴더에서 소스 폴더로 이동하여 `Target.cs` 파일을 찾습니다.

   b. `[project-name]Editor.Target.cs` 파일을 이름이 `[project-name]Client.Target.cs` 및 `[project-name]Server.Target.cs`인 새 파일 두 개에 복사합니다.

   c. 다음과 같이 각 새 파일을 편집하여 클래스 이름과 대상 유형 값을 업데이트합니다.

```
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. .Build.cs 파일을 업데이트합니다.

   a. 프로젝트에 대한 .Build.cs 파일을 엽니다. 이 파일은 UnrealProjects/[project name]/Source/[project name]/[project name].Build.cs에 위치합니다.

   b. 다음 코드 샘플에 표시된 대로 ModuleRules 클래스를 업데이트합니다.

```
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. 게임 프로젝트 솔루션을 다시 빌드합니다.
4. Unreal Engine 편집기의 소스 빌드 버전에서 게임 프로젝트를 엽니다.
5. 클라이언트와 서버 모두에 대해 다음을 수행합니다.

   a. 대상을 선택합니다. 플랫폼, Windows로 이동하여 다음 중 하나를 선택합니다.

      - 서버: `[your-application-name]Server`
      - 클라이언트: `[your-application-name]Client`

   b. 빌드를 시작합니다. 플랫폼, Windows, 패키지 프로젝트로 이동합니다.

각 패키징 프로세스는 실행 파일(`[your-application-name]Client.exe` 또는 `[your-application-name]Server.exe`)을 생성합니다.

플러그인에서 로컬 워크스테이션에 있는 클라이언트 및 서버 빌드 실행 파일의 경로를 설정합니다.

3단계: Anywhere 플릿에 연결

이 단계에서는 사용할 Anywhere 플릿을 지정합니다. Anywhere 플릿은 게임 서버 호스팅을 위해 어디에서나 배치할 수 있는 컴퓨팅 리소스 모음을 정의합니다.

- 현재 사용 중인 AWS 계정에 기존 Anywhere 플릿이 있는 경우 플릿 이름 드롭다운 필드를 열고 플릿을 선택합니다. 이 드롭다운에는 현재 활성 사용자 프로필의 AWS 리전 내 Anywhere 플릿만 표시합니다.
- 기존 플릿이 없거나 새 플릿을 생성하려는 경우 새 Anywhere 플릿 생성을 선택하고 플릿 이름을 입력합니다.

프로젝트에 대해 Anywhere 플릿을 선택하면 Amazon은 플릿 상태가 활성 상태인지 GameLift 확인하고 플릿 ID를 표시합니다. 언리얼 에디터의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

4단계: 워크스테이션 등록

이 단계에서는 로컬 워크스테이션을 새로운 Anywhere 플릿의 컴퓨팅 리소스로 등록합니다.

1. 로컬 시스템의 컴퓨팅 이름을 입력합니다. 플릿에 두 개 이상의 컴퓨팅을 추가하는 경우 이름은 고유해야 합니다.
2. 로컬 시스템의 IP 주소를 입력합니다. 이 필드의 기본값은 컴퓨터의 퍼블릭 IP 주소입니다. 게임 클라이언트와 서버를 같은 컴퓨터에서 실행하는 한 localhost (127.0.0.1) 를 사용할 수도 있습니다.
3. 컴퓨팅 등록을 선택합니다. 언리얼 에디터의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

이 조치에 대한 응답으로 Amazon은 컴퓨팅에 연결할 수 GameLift 있는지 확인하고 새로 등록된 컴퓨팅에 대한 정보를 반환합니다. 또한 Amazon GameLift 서비스와의 통신을 초기화할 때 게임 실행 파일에 필요한 콘솔 인수를 생성합니다.

5단계: 인증 토큰 생성

Anywhere 컴퓨팅에서 실행되는 게임 서버 프로세스에는 서비스를 호출하기 위한 인증 토큰이 GameLift 필요합니다. 플러그인에서 게임 서버를 실행할 때마다 플러그인은 Anywhere 플릿에 대한 인증 토큰을 자동으로 생성하고 저장합니다. 인증 토큰 값은 명령줄 인수로 저장되며, 서버 코드를 런타임 시 검색할 수 있습니다.

이 단계에서는 어떤 조치도 취하지 않아도 됩니다.

6단계: 게임 실행

이제 GameLift Amazon을 사용하여 로컬 워크스테이션에서 멀티플레이어 게임을 시작하고 플레이하는 데 필요한 모든 작업을 완료했습니다.

1. 게임 서버를 시작합니다. 게임 서버는 게임 세션을 호스팅할 준비가 GameLift 되면 Amazon에 알립니다.
2. 게임 클라이언트를 시작하고 새 기능을 사용하여 새 게임 세션을 시작합니다. 이 요청은 새 백엔드 서비스를 GameLift 통해 Amazon으로 전송됩니다. 이에 대한 응답으로 GameLift Amazon은 로컬 시스템에서 실행되는 게임 서버를 호출하여 새 게임 세션을 시작합니다. 게임 세션이 플레이어를 받아들일 준비가 되면 GameLift Amazon은 게임 클라이언트가 게임 세션에 참여할 수 있도록 연결 정보를 제공합니다.

관리형 EC2 플릿을 사용하여 클라우드 호스팅에 게임을 배포합니다.

이 워크플로우에서는 GameLift Amazon에서 관리하는 클라우드 기반 컴퓨팅 리소스에서 호스팅하도록 플러그인을 사용하여 게임을 수정합니다. Amazon GameLift 기능을 위한 클라이언트 및 서버 게임 코드를 추가한 다음, 서버 빌드를 Amazon GameLift 서비스에 업로드하여 클라우드 기반 리소스에 배

포합니다. 이 워크플로가 완료되면 클라우드의 게임 서버에 연결할 수 있는 게임 클라이언트가 작동하게 됩니다.

Amazon GameLift 관리형 Amazon EC2 워크플로를 시작하려면:

- 언리얼 에디터 메인 툴바에서 Amazon GameLift 메뉴를 선택하고 관리형 EC2를 사용하는 호스트를 선택합니다. 이 작업을 수행하면 Amazon EC2 플릿 배포 플러그인 페이지가 열리고 게임 구성 요소를 통합, 빌드, 배포 및 시작하는 6단계 프로세스를 제공합니다.

1단계: 프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필 AWS 계정과 연결되며 프로필의 기본 AWS 지역에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

1. 사용 가능한 프로필 드롭다운 목록에서 프로필을 선택합니다. 아직 프로필이 없거나 새 프로필을 만들려면 Amazon GameLift 메뉴로 이동하여 AWS사용자 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 “활성”이 아닌 경우, Bootstrap 프로필을 선택하고 상태가 “활성”으로 변경될 때까지 기다리십시오.

2단계: 게임 코드 설정

이 단계에서는 클라이언트와 서버 코드에 대한 일련의 업데이트를 수행하여 호스팅 기능을 추가합니다. 언리얼 에디터의 소스 빌드 버전을 아직 설정하지 않은 경우 플러그인에서 지침과 소스 코드에 대한 링크를 제공합니다.

Anywhere 플릿과 함께 사용하기 위해 게임을 통합한 경우 게임 코드를 변경할 필요가 없습니다. 스타트업 게임 맵을 사용하는 경우 EC2 배포에도 사용할 수 있습니다.

- 게임 코드 설정(Anywhere)
- 게임 구성 요소 구축

게임 서버를 구축한 후 다음 작업을 완료하여 GameLift Amazon에 업로드할 준비를 하십시오.

클라우드 배포를 위해 서버 빌드를 패키징하려면

Unreal 편집기가 기본적으로 서버 빌드 파일을 패키징하는 `WindowsServer` 폴더에 다음을 추가합니다.

1. 플러그인 다운로드에 포함된 설치 스크립트를 `WindowsServer` 폴더의 루트에 복사합니다. `[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat` 파일을 찾습니다. GameLift Amazon은 이 파일을 사용하여 각 EC2 호스팅 리소스에 서버 빌드를 설치합니다.
2. Visual Studio 설치에 포함된 `VC_redist.x64.exe` 파일을 `WindowsServer` 폴더의 루트에 복사합니다. 이 파일은 일반적으로 `C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`에 있습니다.
3. 게임 서버 빌드의 OpenSSL DLL을 `WindowsServer/MyGame/Binaries/Win64` 폴더에 복사합니다. DLL이 서버 빌드에 사용된 버전과 동일한지 확인합니다. 다음 파일을 복사합니다.
   - `libssl-3-x64.dll`
   - `libcrypto-3-x64.dll`

3단계: 배포 시나리오 선택

이 단계에서는 지금 배포하려는 게임 호스팅 솔루션을 선택합니다. 어떤 시나리오로든 게임을 여러 번 배포할 수 있습니다.

- 단일 지역 집합: 활성 프로필의 기본 지역에 있는 단일 호스팅 리소스 집합에 게임 서버를 배포합니다. AWS 처음에는 이 시나리오로 AWS와의 서버 통합 및 서버 빌드 구성을 테스트를 시작하는 것이 좋습니다. 다음과 같은 리소스를 배포합니다.
  - 게임 서버 빌드가 설치되어 실행 중인 AWS 플릿(온디맨드).
  - 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
  - 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
  - API 게이트웨이에 대한 과도한 플레이어 호출을 제한하기 위한 WebACl.
  - API 게이트웨이 + 플레이어가 게임 슬롯을 요청할 수 있는 Lambda 함수. 이 함수는 사용할 수 없는 경우 `CreateGameSession()`을 호출합니다.
  - API 게이트웨이 + 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 Lambda 함수.
- FlexMatch 플릿: 게임 서버를 플릿 세트에 배포하고 플레이어 매치를 생성하는 규칙이 포함된 FlexMatch 매치메이커를 설정합니다. 이 시나리오에서는 안정적인 가용성을 위해 다중 플릿, 다중 위치 구조의 저렴한 스팟 호스팅을 사용합니다. 이 접근 방식은 호스팅 솔루션을 위한 매치메이커 구성 요소 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오에서는 이 솔루션의 기본 리소스를 만들고 나중에 필요에 따라 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.
  - FlexMatch 플레이어 요청을 수락하고 매치를 구성하도록 매치메이킹 구성 및 매치메이킹 규칙을 설정합니다.

- 게임 서버 빌드가 여러 위치에 설치되어 실행 중인 AWS 플릿 3대. 백업용으로 스팟 플릿 2개와 온디맨드 플릿 1개가 포함됩니다.
- AWS 게임 세션 배치 대기열은 (실행 가능성, 비용, 플레이어 지연 시간 등을) 기반으로 가능한 최상의 호스팅 리소스를 찾고 게임 세션을 시작하여 제안된 매치에 대한 요청을 충족시켜 줍니다.
- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
- 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
- API 게이트웨이에 대한 과도한 플레이어 호출을 제한하기 위한 WebACl.
- API 게이트웨이 + 플레이어가 게임 슬롯을 요청할 수 있는 Lambda 함수. `StartMatchmaking()` 함수를 호출합니다.
- API 게이트웨이 + 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 Lambda 함수.
- 플레이어용 매치메이킹 티켓과 게임 세션 정보를 저장하는 Amazon DynamoDB 테이블.
- SNS 주제 + 이벤트를 처리하는 Lambda 함수 GameSessionQueue

4단계: 게임 파라미터 설정

이 단계에서는 AWS에 업로드할 게임을 설명합니다.

- 서버 빌드 이름: 게임 서버 빌드에 의미 있는 이름을 제공하십시오. AWS이 이름은 업로드되어 배포에 사용되는 서버 빌드의 사본을 가리킬 때 사용합니다.
- 서버 빌드 OS: 서버가 실행되도록 빌드된 운영 체제를 입력합니다. 이는 게임을 호스팅하는 데 사용할 컴퓨팅 리소스 유형을 AWS에 알려줍니다.
- 게임 서버 폴더: 로컬 서버 빌드 폴더의 경로를 식별합니다.
- 게임 서버 빌드: 게임 서버 실행 파일의 경로를 식별합니다.
- 게임 클라이언트 경로: 게임 클라이언트 실행 파일의 경로를 식별합니다.
- 클라이언트 구성 출력: 이 필드는 AWS 구성이 포함된 클라이언트 빌드의 폴더를 가리켜야 합니다. 다음 위치에서 찾습니다. `[client-build]/[project-name]/Content/CloudFormation`

5단계: 배포 시나리오

이 단계에서는 선택한 배포 시나리오에 따라 게임을 클라우드 호스팅 솔루션에 배포합니다. 이 프로세스는 AWS가 서버 빌드를 검증하고, 호스팅 리소스를 프로비저닝하며, 게임 서버를 설치하고, 서버 프로세스를 시작하며, 게임 세션을 호스팅할 준비를 하는 데 최대 40분 정도 걸릴 수 있습니다.

배포를 시작하려면 [Deploy] 를 선택합니다. CloudFormation 여기에서 게임 호스팅 상태를 추적할 수 있습니다. 자세한 내용은 AWS의 AWS 관리 콘솔에 로그인하여 이벤트 알림을 확인할 수 있습니다. 플러그인의 활성 사용자 프로필과 동일한 계정, 사용자 및 AWS 리전을 사용하여 로그인해야 합니다.

배포가 완료되면 게임 서버가 AWS EC2 인스턴스에 설치됩니다. 하나 이상의 서버 프로세스가 실행 중이며 게임 세션을 시작할 준비가 되었습니다.

6단계: 클라이언트 시작

이제 Amazon에서 호스팅하는 멀티플레이어 게임을 시작하고 플레이하는 데 필요한 모든 작업을 GameLift 완료했습니다. 게임을 플레이하려면 게임 클라이언트의 인스턴스를 시작합니다.

단일 플릿 시나리오를 배포한 경우 한 명의 플레이어가 있는 단일 클라이언트 인스턴스를 열고 서버 맵을 입력 및 이동할 수 있습니다. 게임 클라이언트의 추가 인스턴스를 열어 동일한 서버 게임 맵에 두 번째 플레이어를 추가합니다.

FlexMatch 시나리오를 배포한 경우 솔루션은 플레이어가 서버 맵에 들어가기 전에 게임 세션 배치를 위해 최소 두 개의 클라이언트가 대기할 때까지 기다립니다.

Amazon GameLift 인스턴스의 플릿 데이터를 가져옵니다.

사용자 지정 게임 빌드 또는 Realtime 서버 스크립트에 Amazon GameLift 플릿에 대한 정보가 필요한 경우가 있습니다. 예를 들어 게임 빌드나 스크립트에는 다음과 같은 코드가 포함될 수 있습니다.

- 플릿 데이터를 기반으로 활동을 모니터링합니다.
- 지표를 집계하여 플릿 데이터별로 활동을 추적합니다. (대부분의 게임이 이 데이터를 LiveOps 활동에 사용합니다.)
- 매치메이킹, 추가 용량 확장 또는 테스트와 같은 사용자 지정 게임 서비스에 관련 데이터를 제공합니다.

플릿 정보는 다음 위치의 각 인스턴스에서 JSON 파일로 제공됩니다.

- Windows: `C:\GameMetadata\gamelift-metadata.json`
- Linux: `/local/gamemetadata/gamelift-metadata.json`

이 `gamelift-metadata.json` 파일에는 Amazon GameLift 플릿 리소스의 속성이 포함되어 있습니다.

JSON 파일 예제

```
{
    "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"
}
```

FlexMatch 매치메이킹 추가

Amazon GameLift FlexMatch를 사용하여 Amazon GameLift에서 호스팅하는 게임에 플레이어 매치메이킹 기능을 추가할 수 있습니다. FlexMatch는 사용자 지정 게임 서버 또는 Realtime 서버와 함께 사용할 수 있습니다.

FlexMatch는 매치메이킹 서비스를 사용자 지정 가능한 규칙 엔진과 연결합니다. 게임에 적합한 플레이어 속성과 게임 모드를 기반으로 플레이어를 매칭하는 방법을 설계합니다. FlexMatch는 게임을 찾는 플레이어를 평가하고, 하나 이상의 팀과 매치를 구성하며, 매치를 주최하기 위해 게임 세션을 시작하는 등의 기본 사항을 관리합니다.

전체 FlexMatch 서비스를 사용하려면 호스팅 리소스에 대기열을 설정해야 합니다. Amazon GameLift는 대기열을 사용하여 여러 리전 및 컴퓨팅 유형에서 게임에 가장 적합한 호스팅 위치를 찾습니다. 특히 Amazon GameLift 대기열은 게임 클라이언트가 제공하는 지연 시간 데이터를 사용하여 게임 세션을 배치하므로, 플레이어가 게임을 플레이할 때 최대한 낮은 지연 시간을 경험하게 됩니다.

게임에 매치메이킹을 통합하는 자세한 지원이 있는 FlexMatch에 대한 자세한 내용은 Amazon GameLift FlexMatch 개발자 가이드의 다음 항목을 참조하세요.

- Amazon GameLift FlexMatch 작동 방식
- FlexMatch 통합 단계

Amazon GameLift 컨테이너를 사용한 호스팅 관리

이 설명서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

GameLift Amazon은 게임 서버 호스팅을 위한 컨테이너식 솔루션을 지원하는 완벽한 클라우드 호스팅 서비스를 제공합니다. Amazon GameLift 컨테이너 플릿을 사용하면 이동성, 민첩성, 내결함성과 같은 컨테이너 이점을 활용할 수 있습니다.

주요 기능

Amazon GameLift 컨테이너 플릿에서 사용할 수 있는 기능은 다음과 같습니다.

- Amazon GameLift 호스팅 리소스에서 게임 서버 소프트웨어를 실행할 수 있는 경량 컨테이너로 사용자 지정 컨테이너 아키텍처를 개발하십시오.
- Amazon GameLift Agent를 포함하면 컨테이너 내 게임 서버 프로세스의 수명 주기를 관리할 수 있습니다. 온 컴퓨팅 에이전트는 서버 프로세스를 시작하는 시기와 방법, 게임 세션 호스팅을 위해 유지 관리할 프로세스 수에 대한 지침을 수행합니다.
- GameLift Amazon에서 제공하는 리소스를 사용자 지정하여 게임 서버 애플리케이션으로 컨테이너 이미지를 빌드할 수 있습니다. 제공된 dockerfile을 사용하여 Linux 기반 컨테이너 이미지를 생성합니다. 컨테이너 플릿의 이미지를 Amazon Elastic 컨테이너 레지스트리 (Amazon ECR) 사설 리포지토리에 저장합니다.
- GameLift Amazon이 지원하는 모든 영역 AWS 리전 또는 로컬 영역에 컨테이너 플릿 리소스를 배포하여 지연 시간이 짧은 플레이어 경험을 제공합니다. 효율적인 플릿 관리를 위해 여러 위치에 있는 컨테이너 플릿을 생성하십시오. 아마존 GameLift 호스팅 위치를 참조하세요.
- Amazon GameLift Anywhere 플릿으로 컨테이너식 게임 호스팅 솔루션을 테스트하십시오. Anywhere를 사용하여 Amazon GameLift SDK 통합 및 컨테이너 이미지 구성을 포함한 솔루션 개발을 로컬에서 테스트할 수 있습니다.
- 컨테이너별 성능 지표로 게임 호스팅 성능을 추적할 수 있습니다. 하드웨어 메트릭을 사용하여 플릿 리소스의 상태를 모니터링하세요.
- 대기열 및 FlexMatch 매치메이킹을 포함한 Amazon GameLift 게임 세션 배치 도구를 사용하여 컨테이너 플릿에서 호스팅되는 가능한 최상의 게임 세션과 플레이어를 연결하세요.
- Amazon용 AWS CloudFormation 템플릿을 사용하여 컨테이너 플릿 리소스를 관리합니다 GameLift.

공개 미리 보기 중 컨테이너 플릿 사용

새 컨테이너 플릿 기능은 현재 공개 미리 보기 단계에 있습니다. 이 단계에서는 다음과 같은 Amazon GameLift 기능이 지원됩니다.

- 컨테이너 플릿을 사용하여 Linux용으로 구축된 게임 서버를 호스팅하십시오. 컨테이너 플릿은 Linux 컨테이너 이미지를 `Amazon_Linux_2023` 사용하고 지원합니다. Windows 컨테이너는 지원되지 않습니다.
- Amazon 서버 SDK 버전 5+에만 게임 GameLift 서버 프로젝트를 통합할 수 있습니다. 이전 버전은 지원되지 않습니다.
- Amazon이 지원하는 모든 Amazon EC2 온디맨드 인스턴스 유형을 사용할 수 있습니다. GameLift 스팟 플릿은 현재 지원되지 않습니다.

Amazon에서 컨테이너가 작동하는 방식 GameLift

이 설명서는 공개 프리뷰 릴리스에 포함된 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

Amazon GameLift 컨테이너 플릿은 컨테이너식 애플리케이션을 배포하고 확장하는 방법에 유연성을 제공하도록 설계되었습니다. Amazon Elastic Container Service (Amazon ECS) 를 사용하여 아마존 플릿의 작업 배포 및 실행을 관리합니다. GameLift 이 주제에서는 Amazon GameLift 관리형 플릿에서 컨테이너를 실행하기 위한 기본 구조 요소를 설명하고, 일반적인 아키텍처를 설명하고, 몇 가지 핵심 개념을 설명합니다.

컨테이너 플릿 구성 요소

플릿

컨테이너 플릿은 Amazon에서 관리하며 컨테이너식 게임 서버를 실행하는 Amazon GameLift EC2 인스턴스의 모음입니다. 플릿을 생성할 때 컨테이너 아키텍처와 게임 서버 소프트웨어를 각 플릿 인스턴스에 배포하는 방법을 구성합니다. 컨테이너 플릿을 단일 AWS 리전 또는 여러 지리적 위치에 배포할 수 있습니다. Amazon GameLift 수동 또는 자동 조정 도구를 사용하여 게임 세션 및 플레이어를 호스팅하도록 컨테이너 플릿의 용량을 확장할 수 있습니다.

Instance

Amazon EC2 인스턴스는 게임 호스팅을 위한 컴퓨팅 파워를 제공하는 가상 서버입니다. GameLiftAmazon에서는 다양한 인스턴스 유형 중에서 선택할 수 있습니다. 각 인스턴스 유형은 CPU, 메모리, 스토리지 및 네트워킹 용량의 다양한 조합을 제공합니다.

컨테이너 플릿을 생성하면 Amazon은 선택한 인스턴스 유형과 플릿 구성을 기반으로 인스턴스를 GameLift 배포합니다. 배포된 각 플릿 인스턴스는 동일하며 컨테이너식 게임 서버 소프트웨어를 동일한 방식으로 실행합니다. 플릿의 인스턴스 수에 따라 플릿의 크기와 게임 호스팅 용량이 결정됩니다.

컨테이너 그룹

GameLift Amazon은 컨테이너 그룹의 개념을 사용하여 컨테이너 세트를 설명하고 관리합니다. 컨테이너 그룹은 컨테이너 "태스크" 또는 "포드"와 유사합니다. 각 컨테이너 그룹 내에서 컨테이너가 사용 가능한 CPU 및 메모리 리소스를 공유하는 방법을 정의할 수 있습니다. 또한 컨테이너 간 종속성을 설정하고 컨테이너 그룹의 수명 주기를 관리할 수 있습니다.

컨테이너 그룹은 각 플릿 인스턴스에서 복제하여 리소스 사용을 최적화할 수 있습니다. 다음과 같이 컨테이너 그룹의 스케줄링 전략을 설정하여 복제를 관리할 수 있습니다.

- 레플리카 컨테이너 그룹은 게임 서버 애플리케이션과 지원 소프트웨어를 실행하는 컨테이너를 관리합니다. 모든 컨테이너 플릿은 복제본 컨테이너 그룹을 정의해야 합니다. 컨테이너 그룹의 요구 사항 및 사용 중인 인스턴스 유형의 리소스에 따라 각 플릿 인스턴스에 복제본 그룹이 여러 복사본을 가질 수 있습니다. 복제 그룹의 모든 컨테이너는 인스턴스 전체에서 자동으로 함께 확장됩니다.
- 선택 사항인 데몬 컨테이너 그룹은 모니터링과 같은 백그라운드 서비스 또는 유틸리티 프로그램을 실행하는 데 유용할 수 있습니다. 게임 서버 소프트웨어는 데몬 그룹의 프로세스에 직접적으로 의존하지 않습니다. 데몬 컨테이너 그룹은 복제되지 않습니다. 각 플릿 인스턴스에는 최대 한 개의 데몬 그룹 사본이 있습니다. 즉, 데몬 그룹의 컨테이너는 복제 그룹의 컨테이너와 함께 플릿 인스턴스 전체에 걸쳐 확장되지 않습니다.

컨테이너 플릿에는 하나의 복제본 컨테이너 그룹이 있어야 하며 선택적으로 하나의 데몬 그룹을 가질 수 있습니다.

컨테이너

컨테이너는 컨테이너 기반 아키텍처의 가장 기본적인 요소입니다. 소프트웨어 실행 파일 및 종속 파일이 포함된 컨테이너 이미지로 구성됩니다. GameLiftAmazon에서 사용할 컨테이너를 정의할 때는 컨테이너에서 소프트웨어가 실행되는 방식을 구성합니다.

컨테이너 플릿의 각 컨테이너 그룹에는 "필수"로 지정된 컨테이너가 한 개 있어야 합니다. 필수 컨테이너는 컨테이너 그룹의 라이프사이클을 좌우합니다. 필수 컨테이너에 장애가 발생하면 전체 컨테이너 그룹이 다시 시작됩니다.

컨테이너 유형에는 다음이 포함됩니다.

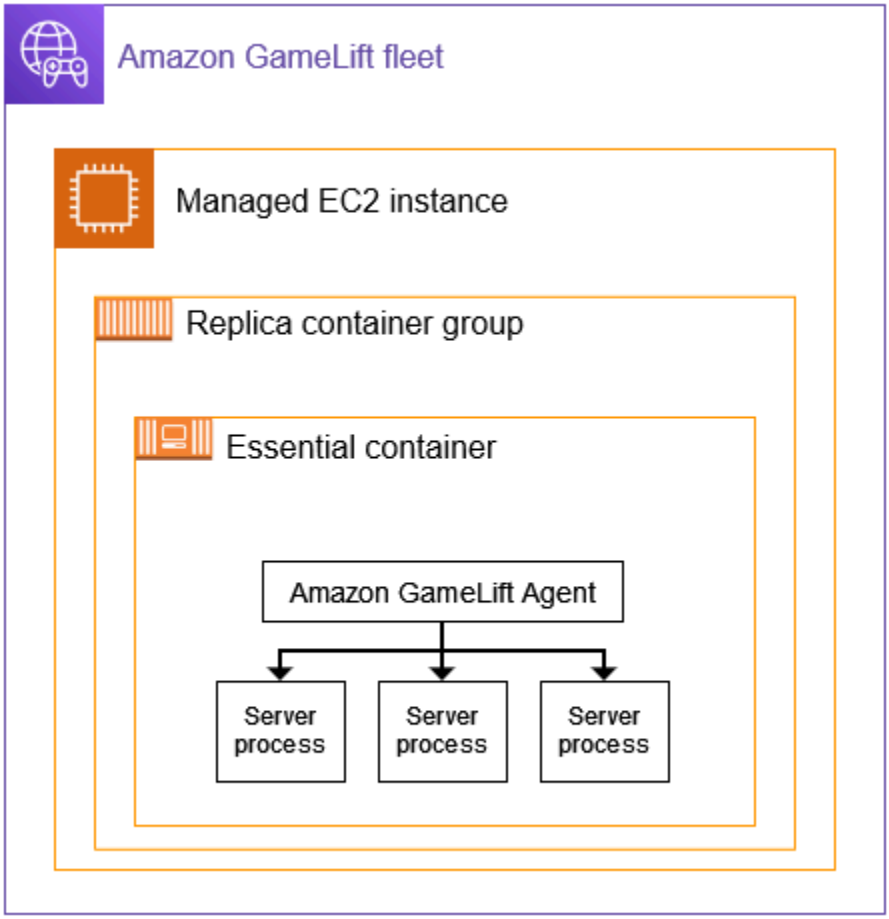
- 필수 레플리카 컨테이너에는 게임 서버 프로세스를 실행하고 플레이어를 위한 게임 세션을 호스팅하는 데 필요한 모든 것이 포함되어 있습니다. 여기에는 Amazon 서버 SDK와 통합된 게임 GameLift 서버 빌드와 종속 소프트웨어가 포함됩니다. 또한 게임 서버 프로세스의 수명 주기를 관리하는 Amazon GameLift Agent도 포함됩니다. 플릿의 복제 컨테이너 그룹에는 정확히 하나의 필수 복제 컨테이너가 있습니다.
- 필수적이지 않은 복제본 컨테이너 (“사이드카” 컨테이너라고도 함) 는 게임 서버 애플리케이션을 지원하는 소프트웨어를 실행합니다. 사이드카 컨테이너를 사용하면 게임 서버와 함께 지원 소프트웨어를 실행하고 확장할 수 있지만 별도의 컨테이너로 관리할 수 있습니다. 이 유형의 컨테이너에 장애가 발생하는 경우 컨테이너 자체만 다시 시작되며 컨테이너 그룹은 영향을 받지 않습니다.
- 데몬 컨테이너는 데몬 서비스를 실행하여 백그라운드 프로세스를 관리합니다. 데몬 컨테이너의 일반적인 용도는 Amazon CloudWatch (CloudWatch) 에이전트를 실행하여 컨테이너에 대한 지표, 로그 및 추적을 수집하는 것입니다. 컨테이너 장애로 인해 컨테이너 그룹이 다시 시작되어야 하는 시기에 따라 데몬 컨테이너는 필수일 수도 있고 필수적이지 않을 수도 있습니다.

컴퓨팅

컴퓨팅은 Amazon GameLift 서비스에 등록되어 서비스와 통신할 수 있는 플릿 호스팅 리소스입니다. 컨테이너 플릿에서 컴퓨트는 컴퓨팅 등록 프로세스를 관리하는 프로세스가 포함된 컨테이너입니다. Amazon GameLift Agent는 컨테이너 플릿의 필수 복제 컨테이너에서 이 컨테이너를 컴퓨팅으로 자동 등록합니다.

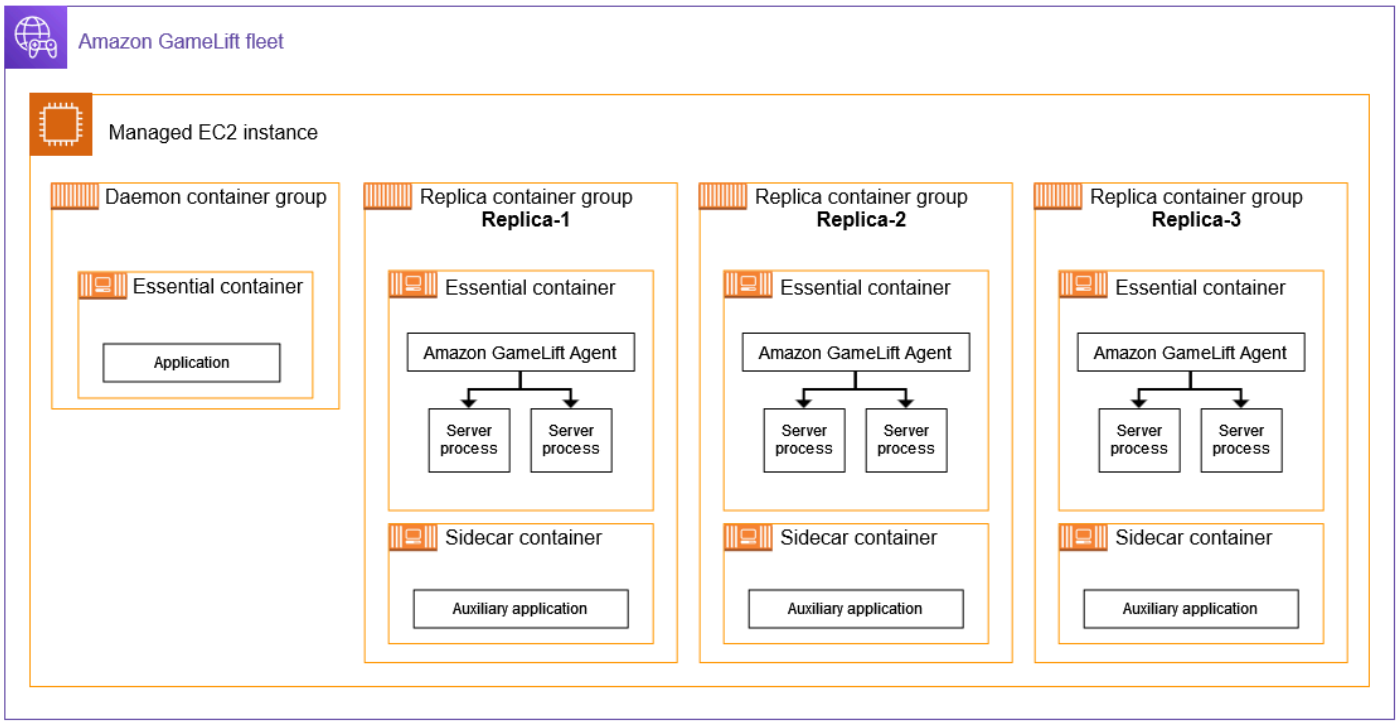
공통 아키텍처

다음 다이어그램은 가장 단순한 컨테이너 플릿 구조를 보여줍니다. 이 구조에서는 플릿의 각 인스턴스가 복제본 컨테이너 그룹의 복사본 하나를 유지 관리합니다. 컨테이너 그룹에는 Amazon GameLift Agent, 게임 서버 애플리케이션 및 게임 세션 호스팅을 지원하는 모든 소프트웨어를 실행하는 단일 필수 컨테이너가 있습니다. 에이전트는 플릿별 지침을 구현하여 세 개의 서버 프로세스를 동시에 실행합니다. 인스턴스당 하나의 복제본 컨테이너 그룹이 있기 때문에 각 플릿 인스턴스는 세 개의 서버 프로세스를 동시에 실행합니다.



이 두 번째 예는 더 복잡한 컨테이너 플릿 설계를 보여줍니다. 이 예제에서 플릿에는 여러 컨테이너가 있는 복제 컨테이너 그룹과 컨테이너 1개가 있는 데몬 컨테이너 그룹이 있습니다. 플릿 구성은 각 플릿 인스턴스에 복제본 컨테이너 그룹의 복사본 3개를 배치합니다. 데몬 컨테이너 그룹은 복제되지 않습니다.

에 있는 각 복제 그룹 컨테이너 집합에는 각 인스턴스에 세 개의 복사본이 있습니다. 각 필수 복제본 컨테이너에서 에이전트는 두 개의 서버 프로세스를 동시에 실행하도록 지시받습니다. 따라서 각 플릿 인스턴스는 6개의 서버 프로세스를 동시에 실행합니다 (세 개의 필수 복제본 컨테이너에서 각각 두 개의 프로세스).



핵심 개념

이 섹션에서는 Amazon이 몇 가지 기본 컨테이너 GameLift 개념을 구현하는 방법을 요약합니다. 컨테이너 플릿을 사용하는 방법에 대한 지침은 이 가이드의 관련 주제를 참조하십시오.

컨테이너 그룹 포장

컨테이너 플릿에 배치할 컨테이너 구조를 개발할 때 공통적인 목표는 가용 컴퓨팅 파워의 사용을 최적화하는 것입니다. 이 목표를 달성하려면 게임 서버 성능에 영향을 주지 않으면서 플릿 인스턴스에 배치할 수 있는 가장 많은 수의 복제 컨테이너 그룹을 찾아야 합니다.

Amazon이 이를 도와드릴 GameLift 수 있습니다. 다음 정보를 기반으로 인스턴스당 최대 복제 그룹 수를 계산합니다.

- 플릿의 인스턴스 유형과 사용 가능한 CPU 및 메모리 리소스
- 복제본 그룹의 모든 컨테이너에 대해 설정한 CPU 및 메모리 요구 사항.

  데몬 그룹의 모든 컨테이너에 대해 설정한 CPU 및 메모리 요구 사항 (있는 경우)
- 각 인스턴스의 컨테이너 및 기타 중요 애플리케이션을 관리하기 위해 예약된 리소스입니다.

컨테이너 플릿을 생성할 때 계산된 최대 수를 사용하거나 원하는 수를 지정하여 계산된 수를 재정의할 수 있습니다. 가장 좋은 방법은 컨테이너화된 게임 서버 소프트웨어를 사용해 실험하여 정확한 리소스 요구 사항을 결정하는 것입니다. 이 데이터를 사용하여 게임 서버 성능을 위한 최적의 패킹 전략을 찾을 수 있습니다.

게임 서버 및 Amazon GameLift 에이전트

필수 복제 컨테이너를 빌드할 때는 게임 서버 소프트웨어와 Amazon GameLift Agent를 동일한 컨테이너 이미지에 함께 패키징합니다. 이 온컴퓨팅 에이전트는 컨테이너 내 게임 서버의 수명 주기를 제어합니다. 각 복제본 컨테이너 그룹에서 필수 복제본 컨테이너는 에이전트와 모든 게임 서버 프로세스를 실행합니다.

Amazon GameLift Agent는 컨테이너 플릿의 런타임 구성에서 명령을 실행합니다. 런타임 구성은 (1) 실행을 시작할 실행 파일, (2) 선택적 시작 파라미터 세트, (3) 동시에 실행할 프로세스 수를 식별합니다. 런타임 구성에는 여러 실행 파일에 대한 지침이 있을 수 있습니다. 게임 서버 실행 파일용 명령이 하나 이상 있어야 합니다. 예를 들어 런타임 구성에서 에이전트에게 프로덕션 용도로 게임 서버 실행 파일 10개, 테스트용 특수 실행 매개 변수가 있는 동일한 실행 파일의 프로세스 1개, 로깅 유틸리티용 프로세스 1개를 유지 관리하도록 에이전트에 지시할 수 있습니다.

플릿의 런타임 구성은 언제든지 수정할 수 있습니다. Amazon GameLift Agent는 정기적으로 서비스에 업데이트를 요청합니다. 업데이트된 런타임 구성을 사용할 수 있게 되면 에이전트는 이를 수신하고 지침을 구현하기 시작합니다. 작업에는 서버 프로세스 추가 또는 종료가 포함될 수 있습니다.

Amazon GameLift Agent는 Amazon이 관리형 EC2 플릿에 GameLift 사용하는 온컴퓨팅 에이전트의 오픈 소스 버전입니다. 이 안내서는 소스에서 에이전트를 빌드하고 컨테이너 이미지로 빌드하는 방법에 대한 지침을 제공합니다. 에이전트는 다음 작업을 처리합니다.

서버 프로세스 관리:

- 런타임 구성에 따라 서버 프로세스를 시작, 종료 및 교체합니다.
- 제때 활성화되지 않으면 서버 프로세스를 종료하십시오.
- 서버 프로세스가 GameLift 종료되면 Amazon에 보고하십시오.
- 서버 프로세스에 대한 플릿 이벤트를 내보냅니다.

컨테이너 관리:

- GameLiftAmazon의 프롬프트에 따라 서버 프로세스를 종료합니다.
- 컨테이너 상태를 보고하십시오.

로그 업로드 작업:

- 지정된 Amazon S3 버킷에 게임 세션 로그를 업로드합니다.
- 컴퓨팅 상의 에이전트 로그를 지정된 Amazon S3 버킷에 업로드합니다.

플릿 용량 조정

플릿 용량은 플릿이 한 번에 호스팅할 수 있는 게임 세션 수를 측정합니다. 플릿이 동시에 지원할 수 있는 플레이어 수를 기준으로 용량을 측정할 수도 있습니다.

플릿의 호스팅 용량을 늘리거나 줄이려면 플릿 인스턴스를 추가하거나 제거합니다. 컨테이너 플릿의 패킹 전략에 따라 각 플릿 인스턴스에서 동시에 실행되는 게임 세션 수가 결정됩니다. 이 숫자는 플릿 용량을 늘리거나 줄일 때 추가하거나 빼는 게임 세션 (및 플레이어 슬롯) 의 수를 나타냅니다.

컨테이너 플릿을 사용하면 GameLift Amazon에서 제공하는 모든 규모 조정 방법을 사용할 수 있습니다. 다음이 포함됩니다.

- 원하는 특정 플릿 인스턴스 수를 설정하여 플릿 용량을 수동으로 설정합니다.
- 사용 가능한 인스턴스의 원하는 버퍼를 대상으로 지정하여 자동 크기 조정을 설정합니다 (대상 추적). 이 방법은 유휴 호스팅 리소스 세트를 자동으로 유지하므로 들어오는 플레이어가 항상 빠르게 게임에 접속할 수 있습니다. 플레이어 수요가 증가하거나 감소함에 따라 이 버퍼의 크기도 지속적으로 조정됩니다.
- 커스텀 스케일링 규칙 (고급 기능) 으로 자동 스케일링을 설정하세요.

게임 클라이언트/서버 연결

관리형 EC2 플릿과 컨테이너 플릿은 게임 클라이언트와 클라우드 호스팅 게임 서버 간의 연결을 비슷한 방식으로 처리합니다. Amazon이 새 게임 세션을 GameLift 생성하면 서비스가 게임 세션의 연결 정보를 전달합니다. 게임 클라이언트는 이 정보를 사용하여 게임 세션을 호스팅하는 게임 서버에 직접 연결합니다. 모든 유형의 플릿에 대한 연결 정보는 IP 주소와 포트 할당으로 구성됩니다.

컨테이너 플릿을 생성할 때는 두 세트의 포트 범위를 정의합니다. 먼저 게임 클라이언트가 게임에 연결할 수 있는 외부 연결 포트 범위를 정의합니다. 두 번째로, 컨테이너에서 실행되는 각 게임 서버 프로세스에 할당되는 내부 전용 컨테이너 포트 세트를 정의합니다. Amazon은 내부 컨테이너 포트를 외부 연결 포트에 GameLift 동적으로 매핑하여 플레이어가 게임에 액세스할 수 있도록 합니다. 이 접근 방식은 게임 서버가 컨테이너 포트에 직접 액세스하지 못하도록 보호하여 보안을 한층 강화합니다.

컨테이너 플릿의 포트 범위를 정의할 때는 인스턴스의 컨테이너에서 동시에 실행되는 모든 서버 프로세스를 수용할 수 있을 만큼 충분한 포트가 포함된 범위를 제공해야 합니다.

추가 제어를 위해 플릿에 대한 인바운드 권한도 설정합니다. 인바운드 권한은 들어오는 트래픽에 사용할 수 있는 연결 포트를 결정합니다. 플릿의 인바운드 권한은 언제든지 변경할 수 있습니다. 인바운드 권한을 사용하면 필요에 따라 모든 연결 포트를 빠르게 종료하거나, 일부 포트를 열거나, 모두 열 수 있습니다.

Amazon GameLift 컨테이너를 위한 개발 로드맵

이 문서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

다음 워크플로는 Amazon GameLift 컨테이너 플릿에서 게임 서버를 실행하기 위한 단계를 요약합니다.

1단계: 게임을 Amazon과 통합 GameLift

게임 서버를 컨테이너 플릿에 배포할 때 Amazon GameLift 서비스와 통신할 수 있도록 게임 서버에 기능을 추가합니다. FlexMatch 매치메이킹을 사용하는 경우 이 기능을 게임 서버와 클라이언트에 추가하세요. 자세한 내용은 게임을 Amazon과 통합하세요 GameLift섹션을 참조하세요.

- Amazon GameLift 서버 SDK (버전 5+) 를 다운로드하여 게임 프로젝트와 함께 설정하십시오. 서버 SDK는 C++, C#, Go로 제공됩니다.
- 게임 서버 코드를 수정하여 필요한 서버 SDK 기능을 추가하세요.
- Linux용 게임 서버 빌드를 패키징하세요. Windows에서 개발하는 경우 이 단계에서 Linux 환경을 설정하기 위한 추가 작업이 필요할 수 있습니다.
- (선택 사항) Amazon GameLift Anywhere 플릿을 사용하여 게임 서버 통합을 테스트합니다. 컨테이너 이미지를 준비하기 전에 테스트하여 통합 작업과 관련된 문제를 찾아내십시오. 게임 클라이언트/서버 연결을 테스트하려면 게임 클라이언트도 통합하세요.

> **Note**
>
> Windows에서 개발하는 경우 별도의 Linux 작업 영역을 설정하거나 Linux용 Windows 하위 시스템 (WSL) 과 같은 도구를 사용하세요. 게임 서버 빌드를 테스트하고 컨테이너 이미지를 빌드하고 테스트하려면 Linux 환경이 필요합니다.

2단계: 게임 서버 컨테이너 이미지 준비

게임 서버 프로세스를 실행하는 컨테이너 이미지를 생성하고 Amazon에서 사용할 수 있도록 Amazon Elastic Container 레지스트리 (Amazon ECR) 리포지토리에 저장합니다. GameLift 자세한 지침은 게임 서버 소프트웨어로 컨테이너 이미지 준비 섹션을 참조하십시오.

- Linux 게임 빌드, 설치 스크립트, 모든 지원 소프트웨어 및 종속 항목을 사용하여 컨테이너 이미지의 작업 디렉터리를 설정합니다.
- Amazon GameLift Agent 소스 코드를 가져와서 빌드하고 작업 디렉터리에 `jar` 파일을 추가합니다.
- 기본 Dockerfile을 가져와서 게임 서버 소프트웨어로 컨테이너 이미지를 구성하도록 수정하십시오.
- 컨테이너 이미지를 빌드하세요. Linux 환경에서 이 단계를 수행하십시오.
- Amazon ECR 사설 리포지토리를 생성하고 컨테이너 이미지를 여기에 푸시합니다. 컨테이너 플릿을 배포할 계획인 동일한 AWS 계정 AWS 리전 위치에 리포지토리를 생성하십시오.
- (선택 사항) Anywhere 플릿을 사용하여 컨테이너 이미지를 테스트하십시오. Amazon GameLift Agent에 지침을 전달하도록 런타임 구성을 설정할 수 있습니다.

3단계: 컨테이너 및 컨테이너 그룹 생성

Amazon에서 게임을 호스팅하기 위한 컨테이너 아키텍처를 GameLift 설계하십시오. Amazon GameLift 컨테이너 플릿 설계 및 Amazon 컨테이너 플릿에 대한 GameLift 컨테이너 그룹 정의 생성 단원을 참조하세요.

- 컨테이너 구성을 정의하십시오. 각 컨테이너에 대해 런타임 프로세스, 메모리 할당, 상태 점검, 네트워크 포트 등과 같은 문제를 정의합니다.
- Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 컨테이너 구성으로 컨테이너 그룹 정의를 생성합니다. 컨테이너 그룹 정의를 생성하면 GameLift Amazon은 해당 시점에 각 컨테이너 이미지의 스냅샷을 찍습니다.

4단계: 컨테이너식 게임 서버를 컨테이너 플릿에 배포

이전 단계에서 생성한 컨테이너 그룹 정의를 사용하여 컨테이너 플릿을 생성하고 컨테이너식 게임 서버 소프트웨어를 배포합니다. Amazon GameLift 컨테이너 플릿 생성를 참조하세요.

- Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 컨테이너 플릿을 생성합니다.
- 플릿 인스턴스가 배포되고 활성화될 때 플릿 상태를 추적합니다. 플릿 생성 이벤트를 확인하여 플릿이 모든 위치에 성공적으로 배포되고 있는지 확인하십시오.
- 게임 클라이언트가 게임 세션을 요청하고 참여하고 게임을 플레이할 수 있는지 확인하십시오. 매치메이킹을 설정했다면 해당 시나리오를 테스트해 보세요.

5단계: 플릿 관리

프로덕션 수준의 사용을 준비하면서 게임 호스팅 솔루션을 구축하고 호스팅 라이프사이클을 관리하세요.

- 여러 곳에 위치한 플릿과 AWS 리전 플릿을 다른 곳에 만들어 플레이어 기반을 지원하세요.
- 대기열 또는 매치메이킹으로 게임 호스팅 배치를 구성하세요. FlexMatch 다음 리소스를 참조하십시오.
  - 게임 세션 배치에 대한 Amazon GameLift 대기열 설정
  - FlexMatch 개발자 가이드
- 자동 크기 조정을 설정하여 게임 세션에 대한 플레이어 수요에 따라 플릿 용량을 관리하세요.
- 컨테이너 플릿에 대한 모니터링을 설정하세요. Amazon GameLift Metrics로 작업하고, 게임 세션 로그 및 컨테이너 로그를 검색하고, 개별 컨테이너에 대한 원격 액세스를 설정합니다.
- 컨테이너 플릿의 장기 관리를 설정하세요. 플릿 별칭을 사용하면 컨테이너 플릿 업데이트 프로세스를 간소화할 수 있습니다. AWS CloudFormation 템플릿을 생성하여 플릿 라이프사이클을 관리하세요. 다음 리소스를 참조하십시오.
  - Amazon GameLift 플릿에 별칭 추가
  - AWS CloudFormation을 사용하여 리소스 관리

게임을 Amazon과 통합하세요 GameLift

이 문서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

게임 서버 소프트웨어로 컨테이너 이미지를 생성하고 클라우드 호스팅을 GameLift 위해 Amazon에 배포하려면 먼저 게임 프로젝트를 Amazon GameLift 서버 SDK와 통합하고 Linux에서 실행할 게임 서버를 구축하십시오. 이 주제에서는 Amazon에서 GameLift 제공하는 다양한 통합 도구를 소개합니다.

호스팅된 게임 서버는 Amazon GameLift 서비스와 통신할 수 있어야 합니다. Amazon GameLift 서버 SDK (버전 5+) 를 게임 프로젝트에 추가하고 게임의 서버 코드를 수정하여 통신을 설정합니다. GameLift Amazon은 여러 언어 및 게임 엔진을 지원하는 서버 SDK 리소스 및 설명서를 제공합니다.

컨테이너식 게임 서버의 통합 프로세스는 관리형 EC2 또는 Amazon 플릿에서 호스팅하기 위한 게임 서버를 통합하는 것과 거의 동일합니다. GameLift Anywhere

통합 도구

GameLift Amazon은 통합을 위해 다음과 같은 도구 및 언어 지원을 제공합니다.

언리얼 엔진 개발자용

언리얼용 경량 플러그인을 사용하세요. 이 플러그인에는 필수 Amazon GameLift 기능이 있는 C++ 서버 SDK 라이브러리가 포함되어 있습니다. 설명서를 사용하여 플러그인용 언리얼 게임 프로젝트를 구성하고 제공된 코드 블록으로 게임 코드를 업데이트하여 서버 및 클라이언트 빌드에 필요한 기능을 추가하세요.

- SDK 플러그인 다운로드
- 가이드: 언리얼 프로젝트를 Amazon과 통합하세요 GameLift
- 레퍼런스 가이드: 언리얼용 C++ 서버 SDK 5

참고: 언리얼 엔진용 Amazon GameLift 스탠드얼론 플러그인은 컨테이너 플릿 사용을 지원하지 않습니다.

Unity 개발자용

Unity용 경량 플러그인을 사용하세요. 이 플러그인에는 필수 Amazon GameLift 기능이 있는 C# 서버 SDK 라이브러리가 포함되어 있습니다. 설명서를 사용하여 플러그인용 언리얼 게임 프로젝트를 구성하고 제공된 코드 블록으로 게임 코드를 업데이트하여 서버 및 클라이언트 빌드에 필요한 기능을 추가하세요.

- SDK 플러그인 다운로드
- 가이드: Unity 프로젝트를 Amazon과 통합 GameLift
- 레퍼런스 가이드: 유니티용 C# 서버 SDK 5

참고: Unity용 Amazon GameLift 스탠드얼론 플러그인은 컨테이너 플릿 사용을 지원하지 않습니다.

다른 게임 엔진을 사용하는 개발자용

다음 일반 서버 및 클라이언트 통합 지침을 따르십시오.

- 게임 서버 통합
- 게임 클라이언트 통합

GameLift Amazon은 다음 언어를 위한 서버 SDK 5 라이브러리를 제공합니다.

- C++용 서버 SDK 5 [SDK 다운로드] [참조 가이드]
- C #용 서버 SDK 5 [SDK 다운로드] [참조 가이드]

- Go용 서버 SDK 5 [SDK 다운로드] [참조 가이드]

리눅스용 게임 서버 구축

Amazon GameLift 컨테이너 플릿은 Linux 플랫폼에서 실행되는 게임 서버를 지원합니다. 다음은 Linux 타겟용 게임 서버를 구축하기 위한 몇 가지 팁입니다.

- Unity 게임 엔진으로 게임을 개발하는 경우 게임 에디터는 Linux용으로 빌드하기 위한 특별한 요구 사항 없이 빌트인 지원을 제공합니다.
- C++로 게임을 개발하는 경우, C++용 GameLift Amazon 서버 SDK를 빌드할 때와 게임 서버를 구축할 때 Linux용 OpenSSL 라이브러리를 포함해야 합니다. 또한 게임 서버 컨테이너 이미지에 동일한 라이브러리를 포함하십시오.
- Windows에서 언리얼 엔진으로 게임을 개발하는 경우 다음 옵션을 고려해 보세요.
  - 언리얼 엔진으로 크로스 컴파일 툴 체인을 구성해 보세요.
  - 별도의 Linux 작업 공간을 설정하거나 Linux용 Windows 서브시스템 (WSL) 과 같은 툴을 사용하세요. 이 환경을 사용하여 Linux에서 언리얼 에디터를 실행하여 게임 서버를 빌드할 수 있습니다.

통합을 로컬에서 테스트하세요.

Amazon GameLift Anywhere 플릿을 사용하여 로컬에서 게임 통합을 테스트할 수 있습니다. 이 접근 방식은 통합과 직접 관련된 문제를 분리하는 데 도움이 되는 모범 사례입니다. Anywhere플릿은 게임 세션 시작/중지, 플레이어 연결 추적과 같은 테스트 앱 및 게임 시나리오를 실행하는 데 유용한 도구입니다. Anywhere플릿을 사용하면 훨씬 빠르게 빌드하고 테스트할 수 있으므로 호스팅 활동을 더 잘 파악할 수 있습니다.

통합 테스트를 Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트 위한 Amazon GameLift Anywhere 플릿 사용에 대한 도움이 필요하면 을 참조하십시오. 테스트 환경을 설정하는 워크플로는 다음과 같습니다.

1. Linux를 실행하는 로컬 기기를 설정합니다.
2. Anywhere플릿을 설정하세요. 로컬 디바이스의 사용자 지정 위치를 생성하고 Anywhere 플릿을 생성한 다음 로컬 디바이스를 플릿의 컴퓨팅으로 등록합니다.
3. 게임 서버용 인증 토큰을 받으세요. 통합 서버 프로세스에는 Amazon GameLift 서비스 인증을 위한 토큰이 필요합니다. 동시에 실행되는 여러 서버 프로세스에 동일한 토큰을 재사용할 수 있습니다. 이 단계는 통합 테스트를 위해 Anywhere 플릿을 사용하는 경우에만 필요합니다.

> **Note**
>
> 인증 토큰은 일시적이므로 정기적으로 새로 고쳐야 합니다. 새 토큰을 요청하려면 서버 빌드 패키지에 스크립트를 추가하는 것을 고려해 보세요.

4. 의 게임 서버 코드를 업데이트하세요Anywhere. Anywhere 플릿에서 실행하는 경우 게임 서버는 다음 서버 파라미터를 사용하여 서버 SDK 액션 `InitSdk()` (C++) (C#) (Unreal) 을 호출해야 합니다. 이 단계는 통합 테스트를 위해 Anywhere 플릿을 사용하는 경우에만 필요합니다. Amazon GameLift Agent를 컨테이너 이미지에 추가하면 이러한 매개변수를 자동으로 처리합니다.

   환경 변수 또는 시작 시 지정한 콘솔 인수에서 이러한 값을 가져오도록 서버 코드를 설정하는 것이 가장 좋습니다.

   - `webSocketUrl`— 의 호출에서 반환되는 의 `GameLiftServiceSdkEndpoint` 값을 사용합니다`register-compute`.
   - `processId`— 서버 프로세스에 고유한 식별자를 할당합니다.
   - `fleetId`— 의 호출에서 반환되는 Anywhere 플릿 식별자입니다`create-fleet`.
   - `authToken`— 의 호출에서 반환되는 유효한 인증 토큰`get-compute-auth-token`.
5. 로컬 컴퓨터에서 게임 서버 빌드 소프트웨어를 설정하고 서버 프로세스를 시작합니다.

   서버 통합이 성공하면 서버 프로세스가 서버 SDK 작업을 `InitSDK()` 호출하여 Amazon GameLift 서비스와의 연결을 설정하고, 이어서 호출을 통해 서비스에 게임 세션을 호스팅할 준비가 되었음을 알립니다. `ProcessReady()`
6. 게임 세션을 시작합니다. 게임 세션을 요청하도록 게임 클라이언트를 통합한 경우 이를 사용하여 새 게임 세션을 요청할 수 있습니다. 그렇지 않은 경우 AWS CLI 명령을 사용하십시오. create-game-session Amazon은 `GameSession` 객체를 GameLift 생성하고 새 게임 세션을 시작하는 프로세스를 시작합니다.

   통합이 작동하는 경우 Amazon은 로컬 워크스테이션의 서버 프로세스를 GameLift 호출하여 `onStartGameSession()` 콜백을 사용하여 새 게임 세션을 시작합니다. 플레이어를 위한 게임 세션이 준비되면 서버는 호출을 처리합니다. `ActivateGameSession()` 이에 대해 Amazon은 게임 클라이언트가 게임 세션에 연결하여 게임을 플레이할 수 있도록 `GameSession` 상태 및 연결 정보를 GameLift 업데이트합니다.

게임 서버 소프트웨어로 컨테이너 이미지 준비

이 문서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

컨테이너는 Amazon GameLift 컨테이너 플릿의 가장 기본적인 요소입니다. 컨테이너에는 SDK, 소프트웨어, 디렉터리, 파일 등의 종속 항목과 함께 게임 서버가 포함됩니다.

컨테이너 플릿에서 작동하려면 게임 서버가 Linux에서 실행되고 서버 SDK 5.x와 통합되어야 합니다.

주제

워킹 디렉터리 설정

작업 디렉터리는 컨테이너 이미지를 빌드하고 Amazon에서 GameLift 실행하는 방법을 정의하는 데 필요한 모든 파일을 저장하는 곳입니다.

컨테이너 작업 디렉터리를 설정하려면

1. Amazon GameLift 컨테이너 이미지를 사용할 디렉터리를 생성합니다.

   Example

   예:

   ```
   [~/]$ mkdir -p work/glc/gamebuild && cd work && find .
   .
   ./glc
   ./glc/gamebuild
   ```

2. Amazon GameLift 에이전트를 복제합니다.

   Example

   예:

   ```
   [~/work]$ git clone https://github.com/aws/amazon-gamelift-agent.git
   ```

```
Cloning into 'amazon-gamelift-agent'...
```

3. Maven을 GameLiftAgent 사용하여 빌드하세요.

   Example

   예:

   ```
   [~/work]$ cd amazon-gamelift-agent
   ```

   Example

   ```
   [~/work/amazon-gamelift-agent]$ mvn clean compile assembly:single && \
   mv target ../glc && cd .. && find glc
   ```

4. 서버 SDK 5.x와 통합되어 빌드되고 파일로 패키징된 게임 서버를 추가하세요. `.ZIP`
5. `.ZIP`파일을 에 복사하세요. `~/work/glc/gamebuild/`

SDK 5.x 게임 서버가 없는 경우 샘플 SimpleServer게임을 다운로드하여 사용하여 컨테이너 플릿을 사용해 볼 수 있습니다.

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
```

컨테이너 이미지를 빌드하세요.

Dockerfile은 컨테이너를 빌드하기 위한 환경, 소프트웨어 및 지침을 지정합니다.

Dockerfile을 만들려면

1. 하위 디렉토리로 이동합니다. `glc`

   Example

   ```
   [~/work]$ cd glc && find

   .
   ./target
   ./target/GameLiftAgent-1.0.jar
   ./gamebuild
   ```

2. 새 Dockerfile을 만들고 엽니다.

   Example

   예:

   ```
   [~/work/glc]$ nano Dockerfile
   ```

3. 다음 템플릿 중 하나에서 복사한 다음 콘텐츠를 Dockerfile에 붙여넣습니다.

게임 서버용 Dockerfile 템플릿

이 템플릿에는 Amazon GameLift 플릿에서 컨테이너를 사용하기 위해 필요한 최소 지침이 포함되어 있습니다. 게임 서버에 필요한 대로 콘텐츠를 수정하십시오.

```
# 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
```

샘플용 도커파일 **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**
>
> 참고: Dockerfile의 일부 환경 변수는 로 재정의할 수 있습니다. ContainerDefinition

컨테이너 이미지를 빌드하려면

1. 컨테이너 이미지를 만드세요.

   자체 SDK 5.x 서버를 사용하는 경우

   원하는 로컬 저장소 이름을 지정할 수 있습니다.

   Example

   ```
   [~/work/glc]$ docker build -t <local repository name>:<optional tag> .
   ```

   SimpleServer샘플을 사용하는 경우

   Example

   ```
   [~/work/glc]$ docker build -t simple-server:version-1 .
   Successfully built 0123456789012
   Successfully tagged simple-server:version-1
   ```

> **Note**
>
> 다음 예제에서는 *simpler-server#* REPOSITORY 초기값과 값으로 사용합니다. version-1 TAG

2. 이미지 목록을 보고 및 값을 기록해 REPOSITORY 둡니다. IMAGE ID 아래 절차에 따라 이 정보가 필요합니다.

   Example

```
[~/work/glc]$ docker images
REPOSITORY       TAG          IMAGE ID        CREATED           SIZE
simple-server    version-1    0123456789012    14 minutes ago    1.24GB
```

컨테이너 이미지를 Amazon ECR로 푸시합니다.

Amazon ECR의 프라이빗 리포지토리에 컨테이너 이미지를 업로드합니다. 컨테이너 그룹 정의를 생성할 때 Amazon이 컨테이너 이미지의 스냅샷을 찍어 컨테이너 플릿을 배포할 때 사용할 GameLift 수 있도록 이 저장소 위치를 참조합니다.

> **Note**
>
> Amazon ECR 프라이빗 리포지토리가 아직 없다면 새로 만드십시오.

Amazon ECR 자격 증명을 받으려면

- 컨테이너 이미지를 Amazon ECR로 푸시하려면 먼저 임시 양식으로 AWS 자격 증명을 획득하여 Docker에 제공해야 합니다. Docker가 로그인할 수 있도록 Amazon ECR 자격 증명을 가져오십시오.

  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
```

컨테이너 이미지를 Amazon ECR로 푸시하려면

1. 사용하려는 Amazon ECR 사설 리포지토리의 URI를 복사합니다.
2. Amazon ECR 태그를 컨테이너 이미지에 적용합니다.

   Example

   ```
   [~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository
    URI>:<optional tag>
   ```
3. 컨테이너 이미지를 Amazon ECR로 푸시합니다.

   Example

   ```
   [~/work/glc]$ docker image push  <Amazon ECR private repository URI>
   ```

Amazon GameLift 컨테이너 플릿 설계

이 설명서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

이 주제에서는 Amazon GameLift 컨테이너 플릿을 설정할 때 내리는 주요 결정을 설명합니다. 의사 결정은 컨테이너, 컨테이너 그룹 및 플릿에 대한 설정을 구성하는 방법에 영향을 줍니다.

주제

- 플릿 컨테이너 구조를 설계하세요.
- 리소스 제한 설정
- 필수 컨테이너를 지정하세요.
- 네트워크 연결을 구성합니다.
- 컨테이너 상태 점검 설정
- 컨테이너 종속성 설정
- 컨테이너 플릿 구성

플릿 컨테이너 구조를 설계하세요.

첫 번째 단계로 다음을 포함하여 게임 서버를 호스팅하는 데 필요한 소프트웨어와 리소스를 파악하세요.

- 게임 서버 애플리케이션. 호스팅용 Amazon GameLift 기능 (서버 SDK 버전 5+ 포함) 과 애플리케이션을 통합해야 합니다. 게임을 Amazon과 통합하세요 GameLift를 참조하세요.
- 아마존 GameLift 에이전트. 이 컴퓨팅 에이전트는 Amazon GameLift 서비스와의 통신을 유지하고 모든 게임 서버 프로세스의 수명 주기를 관리합니다. 자세한 내용은 게임 서버 및 Amazon GameLift 에이전트를 참조하세요.
- 필요에 따른 추가 소프트웨어 및 리소스. 여기에는 게임 서버 애플리케이션을 실행하는 데 필요한 소프트웨어가 포함될 수 있습니다. 로깅 및 모니터링, 보안, 콘텐츠 전송, 데이터 동기화에는 공통 지원 소프트웨어가 사용됩니다.

다음으로 Amazon GameLift 컨테이너 플릿을 위한 소프트웨어와 리소스를 구성하는 방법을 결정하십시오. GameLift Amazon은 컨테이너 그룹을 사용하여 컨테이너를 구성합니다. 플릿에는 항상 하나의 복제 컨테이너 그룹이 있으며 선택적으로 데몬 컨테이너 플릿을 포함할 수 있습니다. 자세한 내용은 컨테이너 플릿 구성 요소를 참조하세요.

- 먼저 복제 컨테이너 그룹을 설계하세요. 다음 지침을 참고하세요.
  - 게임 서버 애플리케이션과 Amazon GameLift Agent를 동일한 컨테이너로 번들하십시오. 이 컨테이너를 복제본 그룹의 유일한 필수 컨테이너로 만드십시오.
  - 게임 서버의 다른 모든 소프트웨어를 컨테이너로 정리하세요. 모든 항목을 레플리카 그룹의 단일 컨테이너에 넣을 수도 있습니다. 또는 사이드카 컨테이너를 하나 이상 생성할 수도 있습니다. 사이드카를 사용하는 몇 가지 이유는 다음과 같습니다.
    - 개별 소프트웨어의 시작/종료 시퀀스 설정하기. 소프트웨어를 별도의 컨테이너에 배치하고 컨테이너 간에 종속성을 설정하여 이를 달성할 수 있습니다.
    - 메모리 및 CPU 사용에 대한 컨테이너별 제한을 설정합니다.
    - 시작 명령, 진입점, 작업 디렉터리, 환경 변수 또는 상태 확인과 같은 각 컨테이너에 대해 서로 다른 컨테이너 구성 설정을 지정합니다.
- 플릿에 데몬 컨테이너 그룹이 필요한지 여부를 결정하십시오. 다음을 고려하세요.
  - 데몬 컨테이너는 일반적으로 백그라운드 또는 모니터링 프로세스를 실행하는 데 사용됩니다.
  - 데몬 그룹의 컨테이너는 플릿 인스턴스에 복제되지 않습니다. 즉, 데몬 그룹의 컨테이너는 복제본 컨테이너 그룹과 함께 확장되지 않습니다.

- 데몬 그룹에는 컨테이너가 여러 개 있을 수 있습니다. 데몬 그룹의 모든 컨테이너를 필수 컨테이너로 지정할 수 있습니다.

리소스 제한 설정

각 컨테이너 그룹에 대해 그룹이 소프트웨어를 실행하는 데 필요한 메모리와 CPU 양을 결정합니다. GameLift Amazon은 이 정보를 사용하여 컨테이너 그룹의 리소스를 관리합니다. 또한 이 정보를 사용하여 플릿 이미지가 보유할 수 있는 복제 컨테이너 그룹 수를 계산합니다. 개별 컨테이너에 대한 한도를 설정할 수도 있습니다.

컨테이너에 대한 선택적 제한을 설정합니다.

컨테이너별 리소스 제한을 설정하면 개별 컨테이너가 그룹의 리소스를 사용하는 방법을 더 잘 제어할 수 있습니다. 컨테이너별 제한을 설정하지 않으면 그룹 내 모든 컨테이너가 그룹 리소스를 공유합니다. 공유를 통해 필요한 곳에 리소스를 더 유연하게 사용할 수 있습니다. 또한 프로세스가 서로 경쟁하여 컨테이너 고장으로 이어질 가능성도 커집니다.

모든 컨테이너에 대해 다음 `ContainerDefinition` 속성 중 하나를 설정하십시오.

- `SoftLimit`(메모리) - 컨테이너 전용으로 최소 메모리 용량을 예약합니다. 컨테이너는 항상 예약된 양만큼 사용할 수 있습니다. 추가 리소스를 사용할 수 있는 경우 언제든지 이 최소값을 초과할 수 있습니다.
- `HardLimit`(메모리) - 컨테이너의 최대 메모리 제한을 설정합니다. 컨테이너가 이 제한을 초과하면 컨테이너가 다시 시작됩니다.
- `Cpu`제한 — 컨테이너 전용으로 최소 CPU 리소스를 예약합니다. 컨테이너는 항상 예약된 양만큼 사용할 수 있습니다. 추가 리소스를 사용할 수 있는 경우 언제든지 이 최소값을 초과할 수 있습니다. (1024개의 CPU 유닛은 vCPU 1개와 동일합니다.)

컨테이너 그룹의 총 리소스 제한을 설정합니다.

각 컨테이너 그룹에 필요한 메모리 및 CPU 리소스 GameLift 양을 Amazon에 알려주세요. 목표는 게임 서버 성능을 최적화하기에 충분한 리소스를 할당하는 것입니다. GameLift Amazon은 이러한 한도를 사용하여 플릿 인스턴스에 복제 컨테이너 그룹을 패킹하는 방법을 계산합니다. 컨테이너 플릿의 인스턴스 유형을 선택할 때도 이를 사용하게 됩니다.

그룹 내 각 컨테이너의 모든 프로세스에 필요한 총 메모리와 CPU를 계산합니다. 다음을 고려하세요.

- 컨테이너 그룹의 모든 컨테이너에서 실행되는 프로세스는 무엇입니까? 이러한 프로세스에 필요한 리소스를 합산하세요.

- 각 컨테이너 그룹에서 몇 개의 동시 게임 서버 프로세스를 실행할 계획입니까? 이 값은 플릿의 런타임 구성의 일부로 설정하지만 여기서는 충분한 메모리를 계획해야 합니다 (런타임 구성 최적화 참조).

예상 컨테이너 그룹 요구 사항에 따라 다음 `ContainerGroupDefinition` 속성을 설정합니다.

- `TotalMemoryLimit`— 컨테이너 그룹의 최대 메모리 제한을 설정합니다. 그룹의 모든 컨테이너는 할당된 메모리를 공유합니다. 개별 컨테이너 제한을 설정하는 경우 총 메모리 한도는 다음과 같아야 합니다.
  - 모든 컨테이너 소프트 메모리 한도의 합계보다 크거나 같음
  - 그룹 내 컨테이너의 최대 하드 메모리 제한과 같거나 큽니다.
- `TotalCpuLimit` — 컨테이너 그룹의 최대 CPU 제한을 설정합니다. 그룹의 모든 컨테이너는 할당된 CPU 리소스를 공유합니다. 개별 컨테이너 제한을 설정하는 경우 총 CPU 한도는 다음과 같아야 합니다.
  - 모든 컨테이너 CPU 한도의 합계와 같거나 더 큽니다. 가장 좋은 방법은 이 값을 컨테이너 CPU 제한 합계의 두 배로 설정하는 것입니다.

예제 시나리오

다음 세 개의 컨테이너로 복제본 컨테이너 그룹을 정의한다고 가정해 보겠습니다.

- 컨테이너 A는 필수 복제 컨테이너입니다. 게임 서버 프로세스와 Amazon GameLift Agent를 실행합니다. 게임 서버 하나에 필요한 리소스 요구 사항은 512MiB 및 1024 CPU로 추정됩니다. 컨테이너에서 10개의 서버 프로세스를 실행하도록 할 계획입니다. 이 컨테이너는 가장 중요한 소프트웨어를 실행하므로 소프트 메모리 예비를 6144MiB로 설정하고 하드 메모리 제한이나 CPU 예약 한도는 설정하지 않았습니다.
- 컨테이너 B는 리소스 요구량이 1024MiB 및 1536CPU로 추정되는 지원 소프트웨어를 실행합니다. 소프트 메모리 예약 한도는 1024MiB, 하드 메모리 제한은 2048MiB, CPU 예비 한도는 1024MiB로 설정했습니다.
- 컨테이너 C는 중요하지 않은 로깅 및 기타 모니터링 유틸리티를 실행합니다. 하드 메모리 제한은 512MiB로, CPU 예비 한도는 512CPU로 설정했습니다.

이 정보를 사용하여 컨테이너 그룹에 대해 다음과 같은 총 한도를 설정합니다.

- 총 메모리 제한: 7680 MiB. 이 값은 (1) 소프트 메모리 제한의 합계 (6144+1024MiB) 및 (2) 최대 하드 메모리 제한 (1024MiB) 을 초과합니다.
- 총 CPU 제한: 13312 CPU. 이 값은 CPU 제한 (1024+512 CPU) 의 합계를 초과합니다.

필수 컨테이너를 지정하세요.

각 용기에 대해 해당 용기를 필수 또는 비필수 용기로 지정하십시오. 모든 컨테이너 그룹에는 필수 컨테이너가 하나 이상 있어야 합니다. 필수 컨테이너는 게임 서버 호스팅과 같은 컨테이너 그룹의 중요한 작업을 수행합니다. 필수 컨테이너는 항상 실행 중이어야 합니다. 실패하면 전체 컨테이너 그룹이 다시 시작됩니다.

- 플릿의 복제 컨테이너 그룹에는 정확히 하나의 필수 컨테이너가 있을 수 있습니다. 이 컨테이너는 Amazon GameLift Agent를 실행하고 해당 컨테이너는 관리하는 게임 서버 프로세스를 실행합니다.
- 플릿에 데몬 컨테이너 그룹이 있는 경우 필수 컨테이너를 여러 개 지정할 수 있습니다. 컨테이너 장애가 발생하여 컨테이너 그룹이 다시 시작되도록 하려면 데몬 컨테이너를 필수로 설정하세요.

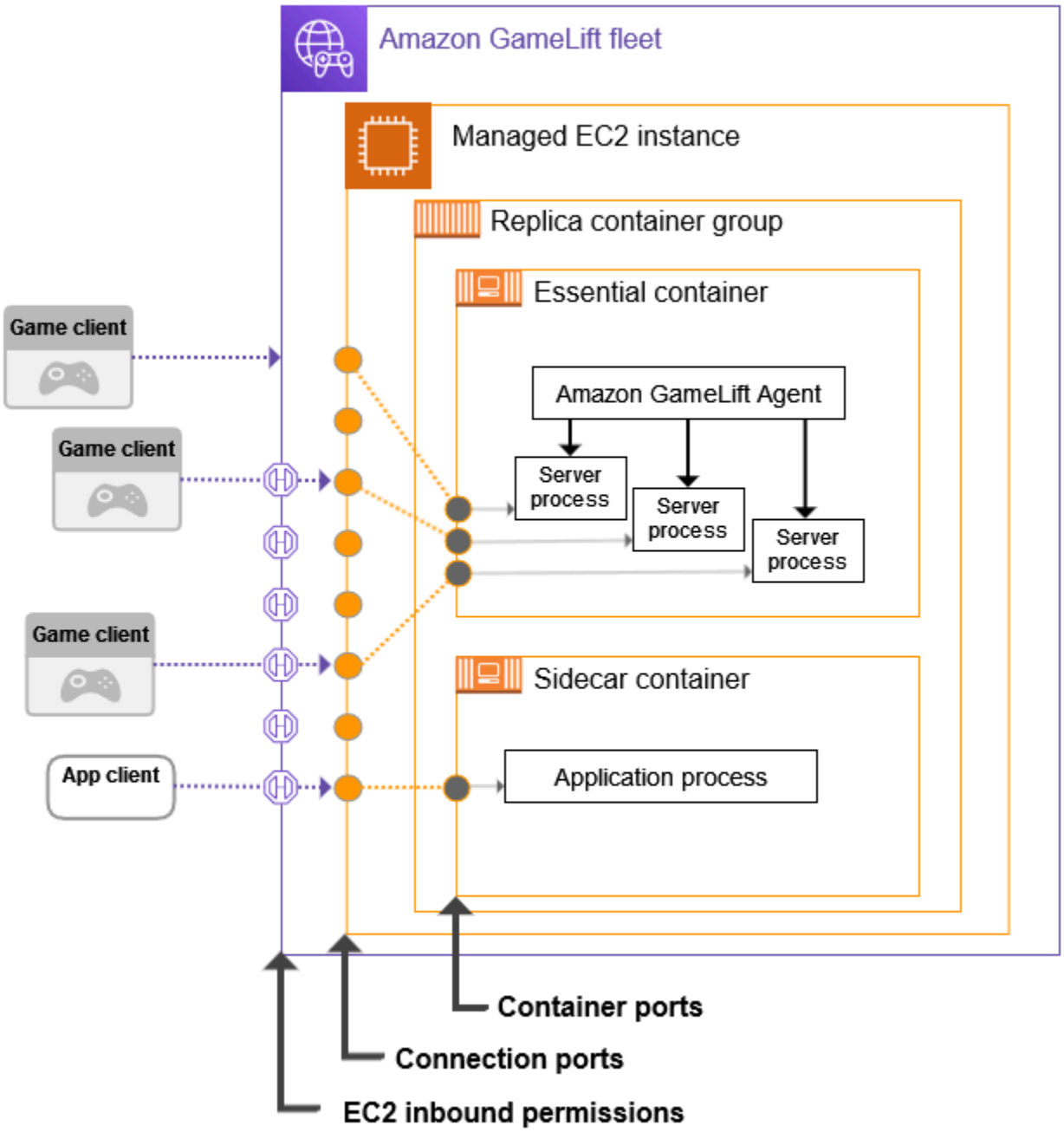
각 컨테이너에 대해 `ContainerDefinition` 속성을 true 또는 `Essential` false로 설정합니다.

네트워크 연결을 구성합니다.

외부 트래픽이 컨테이너 플릿의 모든 컨테이너에 연결되도록 네트워크 액세스를 설정할 수 있습니다. 예를 들어 게임 클라이언트가 게임에 참여하고 플레이할 수 있도록 게임 서버 프로세스를 실행하는 컨테이너에 네트워크 연결을 설정해야 합니다. 게임 클라이언트는 포트와 IP 주소를 사용하여 게임 서버에 연결합니다.

컨테이너 플릿에서는 클라이언트와 서버 간의 연결이 직접적이지 않습니다. 내부적으로 컨테이너의 프로세스는 컨테이너 포트에서 수신합니다. 외부에서는 들어오는 트래픽이 연결 포트를 사용하여 플릿 인스턴스에 연결됩니다. Amazon은 내부 컨테이너 포트와 외부 연결 포트 간의 매핑을 GameLift 유지 관리하여 들어오는 트래픽이 인스턴스의 올바른 프로세스로 라우팅되도록 합니다.

GameLift Amazon은 네트워크 연결을 위한 추가 제어 계층을 제공합니다. 각 컨테이너 플릿에는 각 외부 연결 포트에 대한 액세스를 제어할 수 있는 인바운드 권한 설정이 있습니다. 기존 플릿의 포트 구성을 변경할 수는 없지만 인바운드 권한을 조정하여 필요에 따라 액세스를 허용하거나 제한할 수 있습니다. 예를 들어 모든 연결 포트에 대한 권한을 제거하여 플릿 컨테이너에 대한 모든 액세스를 차단할 수 있습니다.



컨테이너 포트 범위를 설정합니다.

외부 액세스가 필요한 모든 프로세스에 충분한 컨테이너 포트를 포함하는 컨테이너 정의를 구성하십시오. 일부 컨테이너에는 포트가 필요하지 않습니다. 다른 곳에는 필요한 모든 프로세스에 포트를 할당할 수 있을 만큼 충분한 포트가 있어야 합니다.

게임 서버를 실행하는 필수 복제 컨테이너 그룹에는 동시에 실행되는 모든 게임 서버 프로세스 (플릿에 구성된 대로) 를 위한 포트가 필요합니다. `RuntimeConfiguration` 게임 서버 프로세스는 할당된 포트에서 수신 대기하고 GameLift Amazon에 보고합니다.

컨테이너 그룹 정의를 생성할 때 네트워크 액세스가 필요한 각 컨테이너의 컨테이너 포트 범위를 정의하십시오 (참조 ContainerDefinitionInput:PortConfiguration). 포트가 필요한 각 프로세스에 포트를 할당할 수 있을 만큼 범위가 충분히 넓어야 합니다. 컨테이너의 포트 구성에서 프로세스에 포트 번호를 할당해야 합니다.

연결 포트 범위를 설정합니다.

연결 포트 세트를 사용하여 컨테이너 플릿을 구성하십시오. 연결 포트는 컨테이너를 실행하는 플릿 인스턴스에 대한 외부 액세스를 제공합니다. Amazon은 연결 포트를 GameLift 할당하고 필요에 따라 컨테이너 포트에 매핑합니다.

컨테이너 플릿을 생성할 때 연결 포트 범위를 정의하십시오 (참조 ContainerGroupsConfiguration:ConnectionPortRange). 범위에 플릿 인스턴스의 모든 컨테이너 포트에 매핑할 수 있는 충분한 포트가 있는지 확인하십시오. 필요한 최소 연결 포트를 계산하려면 다음 공식을 사용하십시오.

```
[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]
```

가장 좋은 방법은 최소 연결 포트 수를 두 배로 늘리는 것입니다.

> **Note**
>
> 연결 포트 수에 따라 인스턴스당 복제 컨테이너 그룹 수가 제한될 수 있습니다. 플릿에 인스턴스당 하나의 복제 컨테이너 그룹을 위한 충분한 연결 포트가 있는 경우, GameLift Amazon은 인스턴스에 여러 복제 컨테이너 그룹을 위한 충분한 컴퓨팅 파워가 있더라도 하나의 복제본 컨테이너 그룹만 배포합니다.

인바운드 권한 설정

인바운드 권한은 들어오는 트래픽에 사용할 연결 포트를 지정하여 컨테이너 플릿에 대한 외부 액세스를 제어합니다. 이 설정을 사용하여 필요에 따라 플릿의 네트워크 액세스를 켜거나 끌 수 있습니다.

컨테이너 플릿을 생성할 때 인바운드 권한 세트를 정의하십시오 (참조: CreateFleet EC2). InboundPermissions 플릿의 연결 포트 설정에 있는 일부 또는 모든 값을 포함하도록 인바운드 권한 포트 속성을 설정합니다. 기존 컨테이너 플릿의 인바운드 권한을 변경하려면 를 호출하십시오. UpdateFleetPortSettings

예제 시나리오

이 예제는 세 가지 네트워크 연결 속성을 모두 설정하는 방법을 보여줍니다.

- 플릿의 레플리카 컨테이너 그룹에는 게임 서버 프로세스를 실행하는 컨테이너가 1개 있습니다. 런타임 구성은 컨테이너에 10개의 동시 게임 서버 프로세스를 실행하도록 지시합니다.

  레플리카 컨테이너 그룹 정의에서는 이 컨테이너의 PortConfiguration 파라미터를 다음과 같이 설정합니다.

  ```
  "PortConfiguration": {
    "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP"} ]
    }
  ```

- 또한 플릿에는 컨테이너 1개로 구성된 데몬 컨테이너 그룹이 있습니다. 네트워크 액세스가 필요한 프로세스가 하나 있습니다. 데몬 컨테이너 그룹 정의에서 이 컨테이너의 PortConfiguration 파라미터를 다음과 같이 설정합니다.

  ```
  "PortConfiguration": {
    "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ]  }
  ```

- 플릿은 플릿 인스턴스당 3개의 복제 컨테이너 그룹으로 구성되어 있습니다. 이 정보가 제공되면 다음 공식을 사용하여 필요한 연결 포트 수를 계산할 수 있습니다.
  - 최소: 31개 포트 [10개의 복제본 컨테이너 포트* 인스턴스당 3개의 복제본 컨테이너 그룹 + 1개의 데몬 컨테이너 포트]
  - 모범 사례: 62개 포트 [최소 포트* 2]

  컨테이너 플릿을 생성할 때 ConnectionPortRange 파라미터를 다음과 ContainerGroupsConfiguration 같이 설정합니다.

  ```
  "ConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
  ```

- 사용 가능한 모든 연결 포트에 대한 액세스를 허용하고 싶습니다. 컨테이너 플릿을 생성할 때 EC2InboundPermissions 파라미터를 다음과 같이 설정합니다.

  ```
  "EC2InboundPermissions": [
  ```

```
 {"FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":
"TCP"} ]
```

컨테이너 상태 점검 설정

터미널 장애가 발생하고 실행이 중지되면 컨테이너가 자동으로 다시 시작됩니다. 컨테이너가 필수인 경우 전체 컨테이너 그룹이 다시 시작됩니다.

컨테이너 상태를 측정하기 위한 추가 사용자 지정 기준을 정의하고 상태 점검을 사용하여 해당 기준을 테스트할 수 있습니다. 컨테이너 상태 확인을 설정하려면 docker 컨테이너 이미지 또는 컨테이너 정의에서 정의하면 됩니다. 컨테이너 정의에서 상태 확인을 설정하면 컨테이너 이미지의 모든 설정보다 우선합니다.

다음과 같이 컨테이너 유형에 따라 선택적 상태 확인을 설정합니다.

- 필수 복제 컨테이너의 경우 상태 확인을 구성하지 마세요. Amazon GameLift Agent는 이 컨테이너의 상태 보고를 자동으로 처리합니다.
- 필수가 아닌 복제 컨테이너 및 모든 데몬 컨테이너의 경우 선택적으로 상태 확인 파라미터를 설정할 수 있습니다.

컨테이너 상태 점검을 위해 다음 `ContainerDefinition` 속성을 설정합니다.

- `Command`— 컨테이너 상태의 일부 측면을 확인하는 명령을 제공합니다. 상태를 측정하는 데 사용할 기준을 결정합니다. 명령의 종료 값은 1 (비정상) 또는 0 (정상) 이어야 합니다.
- `StartPeriod`— 상태 점검 실패 횟수 집계가 시작되기 전의 초기 지연 시간을 지정합니다. 이 지연으로 인해 컨테이너는 프로세스를 부트스트랩할 시간이 생깁니다.
- `Interval`— 상태 점검 명령의 실행 빈도를 결정합니다. 컨테이너 장애를 얼마나 빨리 감지하고 해결하고 싶으신가요?
- `Timeout`— 상태 점검 명령을 다시 시도하기 전에 성공 또는 실패를 얼마나 기다려야 할지 결정하십시오. 상태 점검 명령을 완료하는 데 시간이 얼마나 걸립니까?
- `Retries`— 실패를 등록하기 전에 상태 점검 명령을 몇 번이나 다시 시도해야 합니까?

컨테이너 종속성 설정

각 컨테이너 그룹 내에서 컨테이너 상태에 따라 컨테이너 간 종속성을 설정할 수 있습니다. 종속성은 다른 컨테이너의 상태에 따라 종속 컨테이너가 시작되거나 종료될 수 있는 시기에 영향을 줍니다.

종속성의 주요 사용 사례는 컨테이너 그룹의 시작 및 종료 시퀀스를 만드는 것입니다.

예를 들어 컨테이너 B와 C가 시작되기 전에 컨테이너 A가 먼저 시작되고 성공적으로 완료되기를 원할 수 있습니다. 이를 위해서는 먼저 컨테이너 A가 성공적으로 완료되어야 한다는 조건으로 컨테이너 A에 컨테이너 B에 대한 종속성을 생성하십시오. 그런 다음 동일한 조건으로 컨테이너 A에 컨테이너 C의 종속성을 생성합니다. 종료 시 시작 순서는 역순으로 발생합니다.

컨테이너 플릿 구성

컨테이너 플릿을 생성할 때는 다음 결정 사항을 고려하십시오. 이러한 요점은 대부분 컨테이너 아키텍처 및 구성에 따라 달라집니다.

플릿을 어디에 배치할지 결정하세요.

일반적으로 지연 시간을 최소화하기 위해 플레이어 근처에 플릿을 지리적으로 배치하는 것이 좋습니다. Amazon이 GameLift 지원하는 모든 Each 서버에 컨테이너 플릿을 배포할 수 있습니다. AWS 리전 동일한 게임 서버를 추가 지리적 위치에 배포하려는 경우 Local Zones를 포함한 AWS 리전 원격 위치를 플릿에 추가할 수 있습니다. 여러 위치에 있는 플릿의 경우 각 플릿 위치에서 용량을 독립적으로 조정할 수 있습니다. 지원되는 플릿 위치에 대한 자세한 내용은 을 참조하십시오아마존 GameLift 호스팅 위치.

플릿의 인스턴스 유형과 크기를 선택합니다.

Amazon은 다양한 Amazon EC2 인스턴스 유형을 GameLift 지원하며, 모두 컨테이너 플릿과 함께 사용할 수 있습니다. 인스턴스 유형, 가용성 및 가격은 위치에 따라 다릅니다. Amazon GameLift 콘솔 (리소스, 인스턴스 및 서비스 할당량 아래) 에서 위치별로 필터링된 지원되는 인스턴스 유형 목록을 볼 수 있습니다.

인스턴스 유형을 선택할 때는 먼저 인스턴스 패밀리를 고려하십시오. 인스턴스 패밀리는 CPU, 메모리, 스토리지 및 네트워킹 기능의 다양한 조합을 제공합니다. EC2 인스턴스 패밀리에 대한 자세한 정보를 확인하십시오. 각 패밀리 내에서 다양한 인스턴스 크기 중에서 선택할 수 있습니다. 인스턴스 크기를 선택할 때는 다음 문제를 고려하십시오.

- 워크로드를 지원할 수 있는 최소 인스턴스 크기는 얼마입니까? 이 정보를 사용하여 너무 작은 인스턴스 유형을 제거하십시오.
- 컨테이너 아키텍처에 적합한 인스턴스 유형 크기는 무엇입니까? 공간 낭비를 최소화하면서 복제 컨테이너 그룹의 여러 복사본을 수용할 수 있는 크기를 선택하는 것이 가장 좋습니다.
- 게임에 적합한 스케일링 세분성은 어느 정도인가요? 플릿 용량 스케일링에는 인스턴스 추가 또는 제거가 포함되며, 각 인스턴스는 특정 수의 게임 세션을 호스팅할 수 있는 능력을 나타냅니다. 각 인스턴스에서 추가 또는 제거하려는 용량을 고려하세요. 플레이어 수요가 분마다 수천 개씩

달라진다면 수백 또는 수천 개의 게임 세션을 호스팅할 수 있는 초대형 인스턴스를 사용하는 것이 합리적일 수 있습니다. 이와는 대조적으로, 더 작은 인스턴스 유형으로 좀 더 세분화된 크기 조정 제어를 선호할 수도 있습니다.

- 규모에 따라 비용을 절감할 수 있나요? 특정 인스턴스 유형의 비용은 가용성으로 인해 위치별로 다를 수 있습니다.

런타임 구성을 최적화하세요.

플릿의 런타임 구성은 게임 세션 호스팅을 위한 서버 프로세스를 실행하는 방법에 대한 일련의 지침입니다. 이 지침은 플릿의 각 복제 컨테이너 그룹에서 Amazon GameLift Agent에 의해 구현됩니다.

플릿의 런타임 구성에 따라 각 복제 컨테이너 그룹에서 동시에 실행되는 서버 프로세스 수가 결정됩니다. 이 설정은 컨테이너 그룹 리소스 한도를 계산하는 방법과 플릿의 인스턴스 유형을 선택하는 방법에 영향을 줍니다. 플릿을 설계할 때는 이 세 가지 요소의 균형을 맞춰야 합니다.

런타임 구성 사용 방법에 대한 자세한 내용은 을 참조하십시오Amazon GameLift가 게임 서버를 시작하는 방법 관리.

기타 선택적 플릿 설정을 설정합니다.

컨테이너 플릿을 구성할 때 다음과 같은 선택적 기능을 사용할 수 있습니다.

- 다른 AWS 리소스에 액세스할 수 있도록 게임 서버를 설정합니다. 플릿에서 다른 AWS 리소스와 통신를 참조하세요.
- 스케일 다운 이벤트 중에 활성 플레이어와의 게임 세션이 조기에 종료되지 않도록 보호하세요.
- 제한된 시간 내에 한 개인이 플릿에서 생성할 수 있는 게임 세션의 수를 제한하십시오.

Amazon 컨테이너 플릿에 대한 GameLift 컨테이너 그룹 정의 생성

이 설명서는 공개 프리뷰 릴리스에 포함된 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

컨테이너 그룹 정의는 컨테이너화된 게임 서버 애플리케이션을 컨테이너 플릿에 배포하는 방법을 설명합니다. 플릿에서 실행할 컨테이너 세트와 이를 실행하는 방법을 식별하는 청사진입니다. 컨테이너 플릿을 생성할 때 플릿에 배포할 컨테이너 그룹 정의를 지정합니다. 컨테이너 그룹에 대한 자세한 내용은 을 참조하십시오컨테이너 플릿 구성 요소.

시작하기 전에

다음 단계를 완료합니다.

- 게임 서버를 호스팅하기 위한 컨테이너 아키텍처를 설계하세요. Amazon GameLift 컨테이너 플릿 설계를 참조하세요.
- 컨테이너 그룹에 포함할 컨테이너 정의를 계획하십시오. AWS CLI를 사용하는 경우 JSON 파일에 컨테이너 정의를 생성하십시오.
- 최종 컨테이너 이미지를 컨테이너 그룹을 생성하려는 곳과 AWS 리전 동일한 Amazon Elastic Container 레지스트리 (Amazon ECR) 레지스트리에 푸시합니다. Amazon은 컨테이너 그룹 정의를 생성할 때 각 이미지의 스냅샷을 GameLift 저장하고 컨테이너 플릿에 배포할 때 사본을 사용합니다. 게임 서버 소프트웨어로 컨테이너 이미지 준비를 참조하세요.
- AWS 사용자에게 Amazon ECR 리포지토리에 액세스할 수 있는 IAM 권한이 있는지 확인하십시오. Amazon의 사용자 권한 관리 GameLift를 참조하세요. 최소한 다음 작업에 대한 권한이 필요합니다.
  - `ecr:DescribeImages`
  - `ecr:BatchGetImage`
  - `ecr:GetDownloadUrlForLayer`

컨테이너 그룹 정의를 복제하십시오.

Amazon GameLift 콘솔을 사용하여 기존 컨테이너 그룹 정의를 복제할 수 있습니다.

컨테이너 그룹을 복제하려면

1. Amazon GameLift 콘솔에서 왼쪽 탐색 창으로 이동하여 컨테이너 그룹을 선택합니다.
2. 컨테이너 그룹 목록 페이지에서 복제하려는 기존 컨테이너 그룹을 선택합니다. 컨테이너 그룹을 선택하면 복제 버튼이 활성화됩니다.
3. 복제를 선택합니다. 이 작업을 수행하면 설정이 미리 채워진 컨테이너 그룹 생성 마법사가 열립니다.
4. 복제된 컨테이너 그룹의 새 이름을 입력합니다. 같은 지역의 컨테이너 그룹은 고유한 이름을 가져야 합니다.
5. 컨테이너 그룹 및 컨테이너 정의 페이지를 단계별로 살펴보고 새 컨테이너 그룹을 생성하십시오.

복제 컨테이너 그룹 정의를 생성합니다.

레플리카 컨테이너 그룹은 게임 서버 소프트웨어를 관리합니다. 복제 컨테이너 그룹에는 Amazon GameLift Agent와 게임 서버 프로세스를 실행하는 컨테이너가 하나 이상 있습니다. 그룹에는 지원 소프트웨어를 실행하기 위한 추가 “사이드카” 컨테이너가 있을 수 있습니다.

이 주제에서는 Amazon GameLift 콘솔 또는 AWS CLI 도구를 사용하여 컨테이너 그룹 정의를 생성하는 방법을 설명합니다. 컨테이너 그룹 구성 설정에 대한 자세한 내용은 을 Amazon GameLift 컨테이너 플릿 설계 참조하십시오.

Console

Amazon GameLift 콘솔에서 컨테이너 그룹을 생성할 AWS 리전 위치를 선택합니다.

콘솔의 왼쪽 탐색 표시줄을 열고 컨테이너 그룹을 선택합니다. 컨테이너 그룹 페이지에서 컨테이너 그룹 생성을 선택합니다.

1단계: 그룹 세부 정보를 정의합니다.

1. 컨테이너 그룹 정의 이름을 입력합니다. 이 이름은 AWS 계정 및 지역에 고유해야 합니다. 콘솔에는 그룹 정의가 이름별로 나열되므로 의미 있는 레이블을 지정하는 데 유용할 수 있습니다.
2. 복제본 스케줄링 전략을 선택합니다.
3. 총 메모리 제한에는 컨테이너 그룹에 사용할 수 있는 최대 메모리를 입력합니다. 이 값을 계산하는 데 도움이 필요하면 을 참조하십시오리소스 제한 설정.
4. 총 CPU 제한에는 컨테이너 그룹에 사용할 수 있는 최대 컴퓨팅 성능을 입력합니다. 이 값을 계산하는 데 도움이 필요하면 을 참조하십시오리소스 제한 설정.

2단계: 컨테이너 정의 추가

게임 서버 애플리케이션과 Amazon GameLift Agent를 사용하여 컨테이너를 정의합니다. 필수 복제 컨테이너입니다.

1. 컨테이너 정의 이름을 입력합니다. 그룹에 정의된 각 컨테이너는 고유한 이름 값을 가져야 합니다.
2. 컨테이너 이미지의 Amazon ECR 이미지 URI를 식별합니다. 다음 형식 중 하나를 입력합니다.
   - 이미지 URI만 해당: `[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]`
   - 이미지 URI+ 다이제스트: `[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]@[digest]`
   - 이미지 URI+ 태그: `[AWS ##].dkr.ecr.[AWS ##].amazonaws.com/[repository ID]:[tag]`

3. 기본 컨테이너의 경우 첫 번째 컨테이너 정의에서 Yes가 자동으로 선택됩니다. 다른 컨테이너 정의를 추가하는 경우 각 정의에 대해 이 설정을 켜거나 끌 수 있습니다. 자세한 내용은 필수 컨테이너를 지정하세요.를 참조하세요.
4. 하나 이상의 내부 컨테이너 포트 범위를 설정합니다. 이 컨테이너는 게임 서버를 호스팅하므로 컨테이너 그룹에서 각 서버 프로세스를 실행하기에 충분한 포트가 있는 범위를 정의하십시오. 자세한 내용은 네트워크 연결을 구성합니다.를 참조하세요.
5. 선택적 설정 Overrides 및 Environment 변수를 사용하면 실행 시 컨테이너에 전달할 값을 지정할 수 있습니다. 여기서 설정한 값은 컨테이너 이미지에 이미 있는 모든 설정보다 우선합니다.
6. 선택적 컨테이너 제한을 설정하여 이 컨테이너의 리소스 할당을 관리하세요. 자세한 내용은 리소스 제한 설정를 참조하세요.
7. 필요에 따라 필수가 아닌 컨테이너를 추가로 정의하십시오.

   - 컨테이너 정의 이름 및 ECR 이미지 URI를 입력합니다. 필수적이지 않은 컨테이너는 Amazon GameLift Agent를 실행해서는 안 됩니다.
   - 컨테이너에 네트워크 액세스가 필요한 프로세스가 있는 경우에만 내부 컨테이너 포트 범위를 설정하십시오.
   - 선택적으로 컨테이너에 대한 건강 검사를 설정합니다. 필수가 아닌 컨테이너가 상태 점검에 실패하면 오류가 발생한 컨테이너만 다시 시작하라는 메시지가 표시됩니다.
   - 필요에 따라 오버라이드, 환경 변수, 리소스 할당 제한을 설정할 수도 있습니다.

3단계: 종속성 구성

컨테이너 그룹 정의에 컨테이너가 두 개 이상 있는 경우 컨테이너 간 종속성을 정의할 수 있습니다. 종속성을 사용하여 컨테이너 조건에 따라 시작 및 종료 시퀀스를 설정합니다. 자세한 내용은 컨테이너 종속성 설정를 참조하세요.

1. 종속성을 추가하려는 컨테이너 이름을 식별하십시오. 이 컨테이너는 종속성 조건이 충족될 때까지 시작되지 않습니다.
2. 종속성 컨테이너 이름 및 조건을 식별하십시오. 이 컨테이너는 조건을 충족해야 종속 컨테이너를 시작할 수 있습니다.
3. 필요에 따라 추가 종속성을 설정합니다. 모든 컨테이너에 대해 여러 종속성을 만들 수 있습니다. 순환 종속성을 만들지 마세요.

4단계: 검토 및 생성

1. 모든 컨테이너 그룹 정의 설정을 검토하십시오. 컨테이너 그룹 정의를 생성한 후에는 구성을 변경할 수 없습니다. 편집을 사용하여 그룹의 각 컨테이너 정의를 비롯한 모든 섹션을 변경할 수 있습니다.
2. 검토를 마치면 [Create] 를 선택합니다.

   요청이 성공하면 콘솔에 새 컨테이너 그룹 정의 리소스의 세부 정보 페이지가 표시됩니다. Amazon이 COPYING 그룹의 모든 컨테이너 이미지에 대한 스냅샷을 GameLift 생성하기 시작하는 시점의 초기 상태입니다. 이 단계가 완료되면 컨테이너 그룹 정의의 상태가 로 READY 변경됩니다. 컨테이너 그룹 정의를 사용하여 컨테이너 플릿을 생성하려면 먼저 컨테이너 그룹 정의가 READY 상태에 있어야 합니다.

AWS CLI

AWS CLI를 사용하여 컨테이너 그룹 정의를 생성할 때는 컨테이너 정의 구성을 별도의 JSON 파일에 유지 관리하십시오. CLI 명령에서 파일을 참조할 수 있습니다. 스키마 예제는 컨테이너 정의 파일 생성 JSON 을 참조하십시오.

컨테이너 그룹 정의 생성

새 컨테이너 그룹 정의를 생성하려면 `create-container-group-definition` CLI 명령을 사용합니다. 이 명령에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하십시오 create-container-group-definition.

Example : 복제 컨테이너 그룹

이 예제는 복제 컨테이너 그룹 정의에 대한 요청을 보여줍니다. 복제본과 데몬 그룹 정의를 만들기 위한 명령 구조는 기본적으로 동일합니다. 각 그룹 유형에 대한 구체적인 세부 정보는 개별 컨테이너 정의에 설명되어 있습니다.

이 예제에서는 이 그룹의 컨테이너 정의가 포함된 JSON 파일을 생성했다고 가정합니다.

```
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
```

컨테이너 정의 파일 생성 JSON

컨테이너 그룹 정의를 생성할 때 그룹의 컨테이너도 정의합니다. 컨테이너 정의는 컨테이너 이미지가 저장되는 Amazon ECR 리포지토리, 네트워크 포트에 대한 선택적 구성, CPU 및 메모리 사용 제한, 기타 설정을 지정합니다. 컨테이너 그룹 내 모든 컨테이너에 대한 구성이 포함된 단일 JSON 파일을 생성하는 것이 좋습니다. 파일을 유지 관리하는 것은 이러한 중요한 구성을 저장, 공유, 버전 추적하는 데 유용합니다. AWS CLI를 사용하여 컨테이너 그룹 정의를 생성하는 경우 명령에서 파일을 참조할 수 있습니다.

컨테이너 정의를 만들려면

1. 새 .JSON 파일을 만들고 엽니다. 예:

   ```
   [~/work/glc]$ vim SimpleServer.json
   ```

2. 그룹의 각 컨테이너에 대해 별도의 컨테이너 정의를 생성합니다. 다음 예제 콘텐츠를 복사하고 컨테이너에 필요한 대로 수정하십시오. 컨테이너 정의 구문에 대한 자세한 내용은 Amazon GameLift API 참조를 참조하십시오 ContainerDefinitionInput.
3. AWS CLI 명령에서 참조할 수 있도록 파일을 로컬에 저장합니다.

예: 필수 복제본 컨테이너 정의

Example

이 예제에서는 복제 컨테이너 그룹의 필수 컨테이너를 설명합니다. 필수 복제 컨테이너에는 게임 서버 애플리케이션인 Amazon GameLift Agent가 포함되며 게임 호스팅을 위한 기타 지원 소프트웨어가 포함될 수 있습니다. 정의에는 이름, 이미지 URI 및 포트 구성이 포함되어야 합니다. 이 예제에서는 일부 컨테이너별 리소스 제한도 설정합니다.

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

Amazon GameLift 컨테이너 플릿 생성

이 설명서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

컨테이너 그룹 정의를 생성했으면 Amazon GameLift 콘솔 또는 AWS Command Line Interface (AWS CLI) 를 사용하여 컨테이너 플릿을 생성하십시오.

새 플릿을 생성한 후 Amazon이 컨테이너 그룹을 각 플릿 인스턴스에 GameLift 배포하고 게임 서버를 시작하면서 플릿의 상태가 여러 단계를 거칩니다. 플릿이 상태에 `ACTIVE` 도달하면 게임 세션을 호스팅할 준비가 된 것입니다. 플릿 생성 문제에 도움을 받으려면 Amazon GameLift 플릿 문제 디버깅 섹션을 참조하세요.

Console

Amazon GameLift 콘솔에서 플릿을 생성할 AWS 리전 위치를 선택합니다. 컨테이너 그룹 정의는 플릿을 생성하려는 지역과 동일한 지역에 있어야 합니다.

콘솔의 왼쪽 내비게이션 바를 열고 플릿을 선택합니다. 플릿 페이지에서 플릿 생성을 선택합니다.

1단계: 컴퓨팅 유형 선택

- 컨테이너 컴퓨팅 유형을 선택합니다.

2단계: 플릿 세부 정보 정의

1. 플릿 세부 정보 섹션에서 플릿 이름과 설명을 입력합니다.
2. 컨테이너 그룹 세부 정보 섹션에서 플릿에 배포할 컨테이너 그룹을 식별합니다. 복제본 컨테이너 그룹을 추가해야 합니다. 선택적으로 데몬 컨테이너 그룹을 추가할 수 있습니다. 각 그룹은 상태에 있어야 합니다. `READY`

3. 플릿의 연결 포트 범위를 설정합니다. 자세한 내용은 네트워크 연결을 구성합니다.를 참조하세요.
4. 필요에 따라 배포할 인스턴스당 원하는 복제본을 지정합니다. 원하는 수를 지정하거나 Amazon에서 가능한 최대 수를 GameLift 계산하도록 할 수 있습니다. 계산된 최대값보다 큰 원하는 수를 지정하면 플릿 생성이 실패합니다. 플릿을 생성한 후에는 이 설정을 변경할 수 없습니다. 복제 컨테이너 그룹 패킹에 대한 자세한 내용은 을 참조하십시오핵심 개념.
5. (선택 사항) 추가 세부 정보에서:
   a. 인스턴스 역할에서 게임 빌드의 애플리케이션이 계정의 다른 AWS 리소스에 액세스할 수 있도록 승인하는 IAM 역할을 지정합니다. 자세한 정보는 플릿에서 다른 AWS 리소스와 통신을 참조하세요. 인스턴스 역할을 사용하여 플릿을 생성하려면 계정에 IAM `PassRole` 권한이 있어야 합니다. 자세한 정보는 Amazon GameLift에 사용되는 IAM 권한 예제을 참조하세요.
   b. 지표 그룹에는 신규 또는 기존 플릿 지표 그룹의 이름을 입력합니다. 동일한 지표 그룹에 추가하여 복수 플릿에 대한 지표를 집계할 수 있습니다.

3단계: 인스턴스 세부 정보 정의

1. 인스턴스 배포에서 인스턴스를 배포할 원격 위치를 하나 이상 선택합니다. 홈 지역 (플릿을 생성할 지역) 이 자동으로 선택됩니다. 위치를 추가로 선택하면 플릿 인스턴스도 해당 위치에 배포됩니다.

> **Important**
>
> 기본적으로 활성화되지 않은 지역을 사용하려면 내 지역에서 해당 지역을 활성화하세요 AWS 계정.
>
> - 2022년 2월 28일 이전에 생성한 리전 중 활성화되지 않은 리전의 플릿은 영향을 받지 않습니다.
> - 새 다중 위치 플릿을 생성하거나 기존의 다중 위치 플릿을 업데이트하려면 먼저 사용하려는 리전을 모두 활성화해야 합니다.
>
> 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전관리를 참조하세요.

2. 플릿의 인스턴스 구성을 선택합니다. 콘솔은 필요한 최소 vCPU 및 메모리를 자동으로 계산합니다 (각 컨테이너 그룹에 설정한 총 한도 기준). 리소스 요구 사항 및 입력한 위치를 기반으로 사용 가능한 인스턴스 유형의 전체 목록을 필터링합니다. 필요에 따라 필터를 더 추가할 수 있습니다.

   인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 컨테이너 플릿 구성 섹션을 참조하세요. 선택한 인스턴스 유형의 크기는 복제 컨테이너 그룹이 각 플릿 인스턴스에 패킹되는 방식에 영향을 줍니다. 선택에 따라 인스턴스당 원하는 복제본에 대한 설정을 검토해 보십시오.

4단계: 런타임 구성

런타임 구성은 게임 서버 프로세스의 시작 및 실행 방법을 결정합니다. 이러한 지침은 Amazon Agent로 전달되며, Amazon GameLift Agent는 각 복제 컨테이너 그룹에서 동일한 방식으로 지침을 구현합니다. 를 호출하여 플릿의 런타임 구성을 업데이트할 수 있습니다. UpdateRuntimeConfiguration

1. Launch path에는 게임 실행 파일의 경로를 입력합니다.
2. (선택 사항) 시작 파라미터에는 게임 실행 파일에 전달할 정보를 명령줄 파라미터 세트로 입력합니다.
3. 각 복제 컨테이너 그룹에서 계속 실행할 동시 프로세스 수를 지정합니다. 인스턴스당 서버 프로세스 수에 대한 Amazon GameLift 할당량을 검토하십시오. 인스턴스당 동시 서버 프로세스 한도는 모든 구성에 대한 총 동시 프로세스 수에 적용됩니다. 한도를 초과하도록 플릿을 구성하면 플릿이 활성화되지 않습니다.
4. 동시 게임 세션 활성화에 대한 선택적 제한을 설정합니다. 이 설정을 통해 새 게임 세션을 시작할 때 소비되는 리소스의 양을 제한할 수 있습니다. 게임 세션 활성화는 기존 게임 세션의 성능에 영향을 미칠 수 있습니다.
5. 외부 트래픽이 플릿에서 실행되는 프로세스에 액세스할 수 있도록 EC2 포트 설정을 설정합니다. 플릿에 정의된 일부 또는 모든 연결 포트 번호를 지정합니다. 플릿을 생성할 때 이러한 포트를 설정할 필요는 없지만 이러한 포트가 없으면 트래픽이 게임 서버에 연결되지 않습니다. 나중에 플릿의 포트 설정을 업데이트하려면 를 호출하세요. UpdateFleetPortSettings
6. 게임 세션 리소스 설정에서 다음과 같은 선택적 기능을 구성합니다.

   a. 게임 스케일링 보호 정책을 켜거나 끕니다. 보호 기능이 켜져 있는 경우 Amazon은 활성 게임 세션을 호스팅하는 경우 축소 이벤트 중에 인스턴스를 종료하지 GameLift 않습니다.

   b. 최대 리소스 생성 한도를 설정하여 지정된 기간 동안 플레이어 한 명이 생성할 수 있는 게임 세션 수를 제한하십시오.

5단계: 태그 구성

- (선택 사항) 키 및 값 쌍을 입력하여 빌드에 태그를 추가합니다. 플릿 생성 검토를 계속하려면 다음을 선택합니다.

6단계: 검토 및 생성.

- 플릿 구성 설정을 검토하십시오.

  플릿 상태와 상관없이 플릿의 메타데이터 및 구성을 언제든 업데이트할 수 있습니다. 자세한 정보는 Amazon GameLift 플릿 관리을 참조하세요. 플릿이 활성 상태가 된 후에 플릿 용량을 업데이트할 수 있습니다. 자세한 정보는 Amazon GameLift 호스팅 용량 확장을 참조하세요. 원격 위치를 추가하거나 제거할 수도 있습니다.

  검토를 마치면 [Create] 를 선택합니다.

  요청이 성공하면 콘솔에 새 플릿 리소스의 세부 정보 페이지가 표시됩니다. Amazon이 플릿 생성 프로세스를 GameLift 시작할 때의 초기 상태는 입니다NEW. 플릿 페이지에서 새 플릿의 상태를 볼 수 있습니다. 플릿은 상태가 되면 게임 세션을 호스팅할 준비가 된 ACTIVE 것입니다.

AWS CLI

를 사용하여 컨테이너 플릿을 AWS CLI생성하려면 명령줄 창을 열고 `create-fleet` 명령을 사용하십시오. 이 명령에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하십시오 `create-fleet`.

아래 예제 `create-fleet` 요청은 다음과 같은 특징을 가진 새 컨테이너 플릿을 생성합니다.

- 는 단일 복제 컨테이너 그룹 정의를 ContainerGroupsConfiguration 지정합니다. `MegaFrogRaceServer.NA.v2` 복제본 그룹의 복사본 3개가 각 플릿 인스턴스에 배포됩니다. 각 인스턴스에는 인스턴스의 프로세스에 액세스할 수 있는 30개의 연결 포트가 있습니다.
- 플릿은 c5.large 온디맨드 인스턴스를 사용합니다.
- 컨테이너 그룹을 다음 위치에 배포합니다.
  - us-west-2(홈 리전)
  - ca-central-1 (원격 위치)
- 인스턴스의 각 복제 컨테이너 그룹은 5개의 게임 서버 프로세스를 동시에 실행하므로 각 인스턴스는 한 번에 최대 15개의 게임 세션을 호스팅할 수 있습니다.

- Amazon은 각 복제 컨테이너 그룹에서 두 개의 새 게임 세션을 동시에 활성화할 수 있도록 GameLift 허용합니다. 또한 300초 이내에 플레이어를 호스팅할 준비가 되지 않은 경우 모든 활성 게임 세션을 종료합니다.
- 이 플릿의 인스턴스에서 호스팅되는 모든 게임 세션에는 게임 세션 보호가 설정되어 있습니다.
- 각 플레이어는 15분 내에 3개의 새 게임 세션을 만들 수 있습니다.

```
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" \
```

플릿 생성 요청이 성공하면 Amazon은 요청한 구성 설정과 새 플릿 ID가 포함된 플릿 속성 세트를 GameLift 반환합니다. GameLift 그러면 Amazon은 플릿 상태와 위치 상태를 신규로 설정하고 플릿 활성화 프로세스를 시작합니다. 이러한 CLI 명령을 사용하여 플릿의 상태를 추적하고 다른 플릿 정보를 확인할 수 있습니다.

- 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

이러한 명령을 사용하여 필요에 따라 플릿의 용량 및 기타 구성 설정을 변경할 수 있습니다.

- update-fleet-attributes
- update-fleet-capacity
- update-fleet-port-settings
- update-runtime-configuration
- create-fleet-locations
- delete-fleet-locations

Amazon GameLift 컨테이너 플릿 관리

이 설명서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

컨테이너 플릿에 대한 정보를 가져오거나 변경하려면 다음 작업을 사용하여 컨테이너 플릿을 관리할 수 있습니다.

리소스 보기

다음은 컨테이너 플릿의 리소스에 대한 정보를 얻을 수 있는 몇 가지 방법입니다.

- DescribeCompute- 컴퓨팅으로 등록된 컨테이너에 대한 세부 정보를 반환합니다.
- DescribeContainerGroupDefinition- 컨테이너 그룹 정의에 대한 세부 정보를 반환합니다. 이 리소스는 그룹과 해당 컨테이너의 구성 방법을 설명합니다.
- DescribeFleetAttributes- 연결 포트 범위 및 기타 속성을 포함하는 플릿 속성을 가져옵니다.
- DescribeFleetCapacity- 플릿의 복제 컨테이너 그룹 수와 상태를 가져옵니다.
- DescribeRuntimeConfiguration- 각 복제 컨테이너 그룹에서 실행되는 서버 프로세스를 설명합니다.
- GetComputeAccess- 컨테이너 그룹을 호스팅하는 인스턴스에 대한 원격 액세스를 제공합니다.
- GetComputeAuthToken- Amazon으로부터 컨테이너 플릿의 컴퓨팅 리소스에 GameLift 대한 인증 토큰을 요청합니다.

- ListCompute- 컴퓨트로 등록된 컨테이너 그룹을 나열합니다.
- ListContainerGroupDefinitions- 컨테이너 그룹 정의를 나열합니다.
- ListFleets- 특정 컨테이너 그룹을 사용하는 플릿을 나열합니다.

리소스 업데이트

다음은 컨테이너 플릿 리소스를 생성하고 수정할 수 있는 몇 가지 방법입니다.

- CreateContainerGroupDefinition- 컨테이너 그룹 정의를 생성합니다.
- CreateFleet- 로 설정하면 컨테이너 플릿을 생성합니다`CONTAINER`. `ComputeType`
- RegisterCompute- 컨테이너 플릿에 계산을 등록합니다.
- UpdateFleetAttributes- Anywhere 플릿 구성 옵션과 같은 플릿의 변경 가능한 속성을 업데이트합니다.
- UpdateFleetCapacity- 관리형 EC2 플릿 또는 컨테이너 플릿의 용량 설정을 업데이트합니다.
- UpdateRuntimeConfiguration- 컴퓨팅으로 등록된 각 복제 컨테이너 그룹에서 실행할 서버 프로세스를 설명하는 런타임 구성을 업데이트합니다.

리소스 삭제하기

다음은 컨테이너 플릿 리소스를 제거할 수 있는 몇 가지 방법입니다.

- DeleteContainerGroupDefinition- 컨테이너 그룹 정의를 삭제합니다.
- DeleteFleet- 플릿을 삭제합니다.
- DeregisterCompute- 컨테이너 플릿에서 컴퓨팅 리소스를 제거합니다.

Amazon GameLift 컨테이너 플릿 규모 조정

이 문서는 공개 프리뷰 릴리스에 있는 기능을 위한 것입니다. 내용은 변경될 수 있습니다.

게임 호스팅에서 가장 어려운 작업 중 하나는 필요하지 않은 리소스에 비용을 낭비하지 않고 플레이어 수요에 맞게 용량을 확장하는 것입니다. 컨테이너 플릿에서는 플릿 인스턴스를 추가하거나 제거하여 플릿 용량을 확장할 수 있습니다.

새 플릿을 생성하면 Amazon은 플릿의 원하는 용량을 인스턴스 1개로 GameLift 설정하고 플릿의 홈 지역에 인스턴스 하나를 배포합니다. 다중 위치 플릿의 경우 Amazon은 홈 지역 및 각 원격 위치에 인스

턴스 하나를 GameLift 배포합니다. 플릿 상태가 ACTIVE 되면 원하는 용량을 늘려 확장하거나 원하는 용량을 줄여 축소할 수 있습니다.

Amazon GameLift Scaling 기능을 사용하여 용량을 수동으로 변경하거나 플레이어 수요에 따라 자동 크기 조정을 설정할 수 있습니다.

- 목표 추적을 통한 자동 크기 조정을 설정하십시오. 대상 기반 Auto Scaling를 참조하세요.
- 플릿의 용량을 수동으로 변경하십시오. Amazon GameLift 플릿의 수동 용량 설정를 참조하세요.

컨테이너 플릿을 확장할 때는 인스턴스 추가 또는 제거가 플릿의 게임 세션 및 플레이어 호스팅 용량에 어떤 영향을 미치는지 고려하세요.

- 인스턴스당 게임 세션
  - 인스턴스에서 실행되는 각 게임 서버 프로세스는 하나의 게임 세션을 호스팅할 수 있는 용량을 나타냅니다.
  - 다음 공식을 사용하여 컨테이너 플릿 인스턴스에서 동시에 실행되는 게임 세션 수를 계산합니다.

    ```
    [Game sessions per instance] = [# of processes per replica container group] * [# of
     replica container groups per instance]
    ```

    - 레플리카 컨테이너 그룹별 프로세스의 경우 게임 서버 프로세스의 동시 실행 수를 DescribeRuntimeConfiguration호출하고 계산합니다.
    - 인스턴스별 레플리카 컨테이너 그룹의 경우 DescribeFleetAttributes호출하여 값을 가져오십시오. DesiredReplicaContainerGroupPerInstance 이 값이 설정되지 않은 경우 MaxReplicaContainerGroupsPerInstance 값을 사용하십시오.
- 인스턴스당 플레이어 수
  - 각 게임 세션에서 허용할 플레이어 슬롯 수를 결정합니다. 호스팅 솔루션이 게임 세션 배치를 처리하는 방식에 따라 매치메이킹 구성 또는 게임 세션 배치를 시작하기 위한 호출에서 게임 세션당 플레이어를 정의할 수 있습니다.
  - 다음 공식을 사용하여 컨테이너 플릿 인스턴스에서 동시에 게임을 플레이할 수 있는 플레이어 수를 계산하세요.

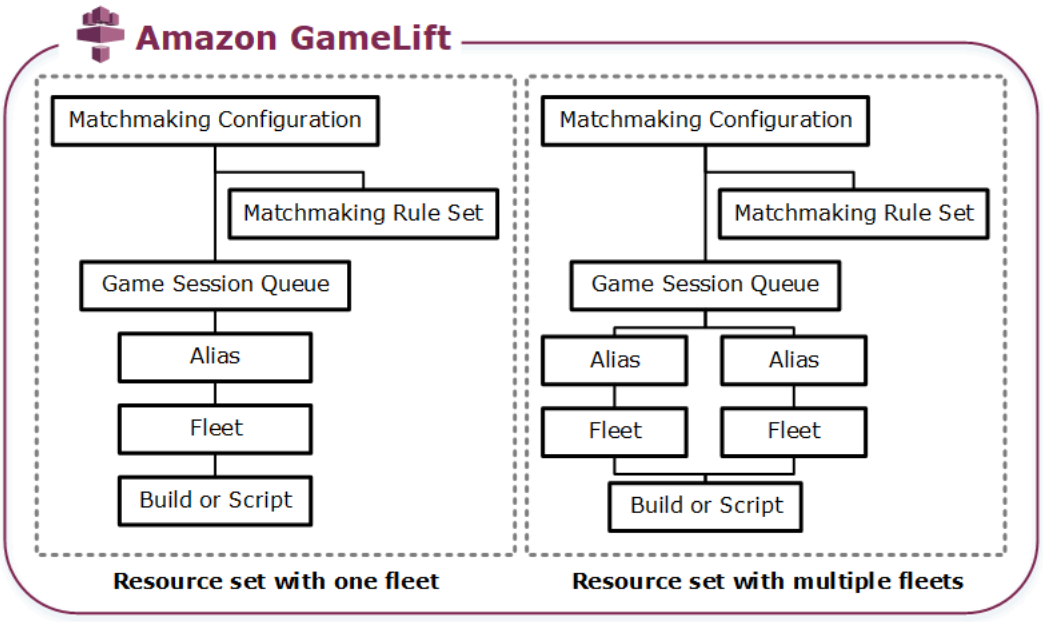
    ```
    [Players per instance] = [# of game sessions per instance] * [# of player slots per
     game session]
    ```

컨테이너 플릿의 현재 총 용량을 구하려면 DescribeFleetCapacity또는 DescribeFleetLocation Capacity 를 호출하여 플릿의 복제 컨테이너 그룹 수를 가져오십시오. 활성 그룹은 현재 게임 세션을 호스팅하고 있는 그룹입니다. 유휴 그룹은 새 게임 세션을 주최할 준비가 되었습니다. 이 값에 복제 컨테이너 그룹당 서버 프로세스 수를 곱하십시오.

Amazon GameLift 호스팅 리소스 관리

이 섹션에서는 게임 서버를 실행하고 플레이어를 위한 게임 세션을 호스팅하도록 Amazon GameLift 관리형 리소스 설정에 대한 자세한 정보를 제공합니다. 리소스를 구성 및 배포하고, 플레이어 수요에 맞게 용량을 확장하고, 게임 세션을 호스팅하는 데 사용할 수 있는 리소스를 찾아야 합니다.

다음 다이어그램은 Amazon GameLift 리소스 객체가 서로 어떻게 관련되는지를 보여 줍니다. 빌드 또는 스크립트를 사용하여 플릿을 생성하고, 플릿에 별칭을 지정하며, 해당 별칭을 사용하여 플릿을 게임 세션 대기열에 추가합니다. FlexMatch 매치메이킹을 사용하는 게임의 경우, 게임 세션 대기열과 매치메이킹 규칙 세트를 사용하여 매치메이킹 구성을 생성합니다.



게임 서버 코드

- 빌드 - Amazon GameLift에서 실행하고 플레이어를 위한 게임 세션을 호스팅하는 사용자 지정으로 빌드된 게임 서버 소프트웨어입니다. 게임 빌드는 특정 운영 체제에서 게임 서버를 실행하고 Amazon GameLift와 통합해야 하는 파일 세트를 나타냅니다. 플릿을 설정하려는 AWS 리전에서 Amazon GameLift로 게임 빌드 파일을 업로드합니다. 자세한 내용은 Amazon GameLift에 사용자 지정 서버 빌드 업로드 섹션을 참조하세요.
- 스크립트 - Realtime 서버에 사용할 구성 및 사용자 지정 게임 로직입니다. JavaScript로 스크립트를 생성하여 게임 클라이언트에 대한 Realtime 서버를 구성하고 플레이어의 게임 세션을 호스팅하도록 사용자 지정 게임 로직을 추가합니다. 자세한 내용은 Amazon GameLift에 Realtime 서버 스크립트 업로드 섹션을 참조하세요.

플릿

게임 서버를 실행하고 플레이어를 위해 게임 세션을 호스팅하는 컴퓨팅 리소스 컬렉션입니다. 플릿을 배포할 수 있는 위치에 대한 자세한 내용은 아마존 GameLift 호스팅 위치을 참조하세요. 플릿 생성에 대한 내용은 Amazon GameLift 플릿 설정 섹션을 참조하세요.

별칭

플레이어가 연결된 플릿을 언제든지 변경하는 데 사용할 수 있는 플릿의 추상적인 식별자입니다. 자세한 내용은 Amazon GameLift 플릿에 별칭 추가 섹션을 참조하세요.

게임 세션 대기열

새 게임 세션에 대한 요청을 수신하고 새 세션을 호스팅할 수 있는 게임 서버를 검색하는 게임 세션 배치 메커니즘입니다. 게임 세션 대기열에 대한 자세한 내용은 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.

Amazon GameLift에 빌드 및 스크립트 업로드

Amazon GameLift를 통해 호스팅하기 위해 멀티플레이어 게임 서버를 배포하기 전에 게임 서버 파일을 업로드해야 합니다. 이 섹션의 항목에서는 사용자 지정 게임 서버 빌드 파일 또는 Realtime 서버 서버 스크립트 파일을 준비하고 업로드하는 방법에 대한 지침을 제공합니다.

주제

- Amazon GameLift에 사용자 지정 서버 빌드 업로드
- Amazon GameLift에 Realtime 서버 스크립트 업로드

Amazon GameLift에 사용자 지정 서버 빌드 업로드

게임 서버를 Amazon GameLift와 통합한 후 빌드 파일을 Amazon GameLift에 업로드합니다. 이 주제에서는 게임의 빌드 파일을 패키징하고, 선택적 빌드 설치 스크립트를 생성한 다음, AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 파일을 업로드하는 방법을 다룹니다.

주제

- 게임 빌드 파일 패키징하기
- Amazon GameLift 빌드 생성
- 빌드 파일 업데이트

- 빌드 설치 스크립트 추가

게임 빌드 파일 패키징하기

구성된 게임 서버를 Amazon GameLift에 업로드하기 전에 게임 빌드 파일을 빌드 디렉터리에 패키징합니다. 이 디렉터리에는 다음을 포함하여 게임 서버를 실행하고 게임 세션을 호스팅하는 데 필요한 모든 구성 요소가 포함되어야 합니다.

- 게임 서버 바이너리 - 게임 서버를 실행하는 데 필요한 바이너리 파일. 같은 플랫폼에서 실행되도록 구축된 경우에 한해, 하나의 빌드에 여러 게임 서버의 바이너리를 포함할 수 있습니다. 지원되는 플랫폼 목록은 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.
- 종속성 - 게임 서버 실행 파일을 실행하는 데 필요한 모든 종속 파일입니다. 예를 들어 자산, 구성 파일, 종속 라이브러리 등이 있습니다.

> **Note**
>
> C++용 Amazon GameLift Server SDK로 만든 게임 빌드(Unreal 플러그인으로 만든 빌드 포함)의 경우, Server SDK를 빌드할 때 사용한 것과 동일한 버전의 OpenSSL용 OpenSSL DLL을 포함합니다. 자세한 내용은 Server SDK README 파일을 참조하세요.

- 설치 스크립트(선택 사항) - Amazon GameLift 호스팅 서버에 게임 빌드를 설치하는 작업을 처리하는 스크립트 파일입니다. 이 파일은 빌드 디렉터리의 루트에 배치합니다. Amazon GameLift는 플릿 생성의 일부로 설치 스크립트를 실행합니다.

다른 AWS 서비스의 리소스에 안전하게 액세스하기 위해 설치 스크립트를 포함하여 빌드에서 애플리케이션을 설정할 수 있습니다. 이를 수행하는 자세한 방법은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

빌드 파일을 패키징한 후에는 게임 서버가 새로 설치된 대상 OS에서 실행될 수 있는지 확인합니다. 이 단계를 통해 필요한 모든 종속성을 패키지에 포함시키고 설치 스크립트가 정확한지 확인합니다.

Amazon GameLift 빌드 생성

빌드를 생성하고 파일을 업로드할 때 몇 가지 옵션이 있습니다.

- 파일 디렉터리에서 빌드 생성. 이 방법은 가장 간단하며 일반적으로 사용되는 선택 사항입니다.
- Amazon Simple Storage Service(S3)의 파일을 사용하여 빌드 생성. 이 선택 사항을 사용하면 Amazon S3에서 빌드 버전을 관리할 수 있습니다.

두 방법을 모두 사용하면 Amazon GameLift가 고유의 빌드 ID 및 기타 메타데이터를 사용하여 새 빌드 리소스를 생성합니다. 빌드는 초기화됨 상태에서 시작됩니다. Amazon GameLift가 게임 서버 파일을 획득한 후 빌드는 준비 완료 상태로 이동합니다.

빌드가 준비되면 새 Amazon GameLift 플릿에 배포할 수 있습니다. 자세한 내용은 Amazon GameLift 관리형 플릿 생성 섹션을 참조하세요. Amazon GameLift는 새 플릿을 설정할 때 빌드 파일을 각 플릿 인스턴스에 다운로드하고 빌드 파일을 설치합니다.

파일 디렉터리에서 빌드 생성

로컬 디렉터리 등 임의의 위치에 저장된 게임 빌드를 생성하려면 `upload-build` AWS CLI 명령을 사용합니다. 이 명령은 Amazon GameLift에 새 빌드 레코드를 생성하고 지정한 위치에서 파일을 업로드합니다.

업로드 요청을 보냅니다. 명령줄 창에 다음 upload-build 명령 및 파라미터를 입력합니다.

```
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 - 게임 서버 빌드의 런타임 환경. OS 값을 지정해야 합니다. 나중에 업데이트할 수 없습니다.
- server-sdk-version - 게임 서버가 통합되는 Amazon GameLift Server SDK 버전입니다. 값을 입력하지 않으면 Amazon GameLift에서 기본값 `4.0.2`를 사용합니다. Server SDK 버전을 잘못 지정하면 Amazon GameLift 서비스에 대한 연결을 설정하도록 `InitSdk`를 호출할 때 게임 서버 빌드가 실패할 수 있습니다.
- build-root - 빌드 파일의 디렉터리 경로입니다.
- name - 새 빌드를 설명하는 이름입니다.
- build-version - 빌드 파일의 버전 세부 정보입니다.
- region - 빌드를 생성하려는 AWS 리전입니다. 플릿을 배포하려고 계획하는 리전에서 빌드를 생성합니다. 여러 리전에서 게임을 배포하는 경우 각 리전에서 빌드를 생성합니다.

> **Note**
>
> aws configure get region을 사용하여 현재 기본 리전을 확인합니다. 기본 리전을 변경하려면 aws configure set region ***region name*** 명령을 사용합니다.

예

```
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
```

업로드 요청에 대한 응답으로 Amazon GameLift는 업로드 진행 상황을 제공합니다. 업로드에 성공하면 Amazon GameLift는 새 빌드 레코드 ID를 반환합니다. 업로드 시간은 게임 파일의 크기와 연결 속도에 따라 달라집니다.

Amazon S3 내 파일을 사용하여 빌드 생성

Amazon S3에 빌드 파일을 저장하고 여기에서 Amazon GameLift에 업로드할 수 있습니다. 빌드를 생성할 때 S3 버킷 위치를 지정하면 Amazon GameLift는 Amazon S3에서 직접 빌드 파일을 검색합니다.

빌드 리소스를 생성하려면

1. 빌드 파일을 Amazon S3에 저장합니다. 패키지된 빌드 파일을 포함하는 .zip 파일을 만들어 AWS 계정의 S3 버킷에 업로드합니다. Amazon GameLift 빌드를 생성할 때 필요하므로 버킷 레이블과 파일 이름을 기록해 둡니다.

2. 빌드 파일에 Amazon GameLift 액세스 권한을 부여합니다. Amazon S3의 게임 빌드 파일에 액세스의 지침에 따라 IAM 역할을 생성합니다. 역할을 생성한 후에 빌드를 생성할 때 필요하므로 새 역할의 Amazon 리소스 이름(ARN)을 기록해 둡니다.
3. 빌드를 생성합니다. Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 새 빌드 레코드를 생성합니다. Amazon GameLift에 사용되는 IAM 권한 예제에 설명된 대로 `PassRole` 권한이 있어야 합니다.

Console

1. Amazon GameLift 콘솔의 탐색 창에서 호스팅, 빌드를 선택합니다.
2. 빌드 페이지에서 빌드 생성을 선택합니다.
3. 빌드 생성 페이지의 빌드 설정에서 다음을 수행하세요.
   a. 이름에 스크립트 이름을 입력합니다.
   b. 버전에 버전을 입력합니다. 빌드의 콘텐츠를 업데이트할 수 있으므로 버전 데이터를 통해 업데이트를 추적할 수 있습니다.
   c. 운영 체제(OS)에 게임 서버 빌드의 OS를 선택합니다. 나중에는 이 값을 업데이트할 수 없습니다.
   d. 게임 서버 빌드에는 Amazon S3에 업로드한 빌드 객체의 S3 URI를 입력하고 객체 버전을 선택합니다. Amazon S3 URI와 객체 버전이 기억나지 않는 경우, S3 찾아보기를 선택하고 빌드 객체를 검색합니다.
   e. IAM 역할에는 Amazon GameLift에서 S3 버킷 및 빌드 객체에 액세스할 수 있도록 생성한 역할을 선택합니다.
4. (선택 사항) 태그에서 키 및 값 쌍을 입력하여 빌드에 태그를 추가합니다.
5. 생성을 선택합니다.

Amazon GameLift는 새 빌드에 ID를 할당하고 지정된.zip 파일을 업로드합니다. 빌드 페이지에서 상태를 포함한 새 빌드를 볼 수 있습니다.

AWS CLI

새로운 빌드를 정의하고 서버 빌드 파일을 업로드하려면 `create-build` 명령을 사용합니다.

1. 명령줄 창을 열고 AWS CLI를 사용할 수 있는 디렉터리로 전환합니다.
2. 다음 create-build 명령을 입력합니다.

```
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 - 새 빌드를 설명하는 이름입니다.
- server-sdk-version - 게임 서버를 Amazon GameLift와 통합하는 데 사용되는 Amazon GameLift Server SDK 버전입니다. 값을 입력하지 않으면 Amazon GameLift에서 기본값 `4.0.2`를 사용합니다.
- operating-system - 게임 서버 빌드의 런타임 환경. OS 값을 지정해야 합니다. 나중에 업데이트할 수 없습니다.
- build-version - 빌드 파일의 버전 세부 정보입니다. 게임 서버의 새 버전이 나올 때마다 새 빌드 리소스가 필요하기 때문에 이 정보가 유용할 수 있습니다.
- storage-location
  - Bucket - 빌드가 포함되어 있는 S3 버킷의 이름입니다. 예: “my_build_files”.
  - Key - 빌드 파일을 포함하는 .zip 파일의 이름입니다. 예: “my_game_build_7.0.1, 7.0.2”.
  - RoleARN - 생성한 IAM 역할에 할당된 ARN입니다. 예: “arn:aws:iam::111122223333:role/GameLiftAccess”. 정책에 대한 예는 Amazon S3의 게임 빌드 파일에 액세스 섹션을 참조하세요.
- region - 플릿을 배포하려고 계획하는 AWS 리전에서 빌드를 생성합니다. 여러 리전에서 게임을 배포하는 경우 각 리전에서 빌드를 생성합니다.

> **Note**
>
> `configure get` 명령을 사용하여 현재 기본 리전을 확인하는 것이 좋습니다. 기본 리전을 변경하려면 `configure set` 명령을 사용합니다.

예

```
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. 새 빌드를 보려면 `describe-build` 명령을 사용합니다.

빌드 파일 업데이트

Amazon GameLift 콘솔 또는 `update-build`AWS CLI 명령을 사용하여 빌드 리소스의 메타데이터를 업데이트할 수 있습니다.

Amazon GameLift 빌드를 생성한 후에는 해당 빌드와 연결된 빌드 파일을 업데이트할 수 없습니다. 각각의 새 파일 세트에 대해 새 Amazon GameLift 빌드를 생성합니다. Amazon GameLift는 `upload-build` 명령을 사용하여 각 요청에 대해 새 빌드 레코드를 자동으로 생성합니다. `create-build` 명령을 사용하여 빌드 파일을 제공하는 경우 다른 이름의 새 빌드 .zip 파일을 Amazon S3로 업로드하고 새 파일을 이름을 참조하여 빌드를 생성합니다.

다음은 업데이트된 빌드를 배포할 때의 팁입니다.

- 필요에 따라 대기열을 사용하고 및 플릿을 스왑합니다. Amazon GameLift를 사용하여 게임 클라이언트를 설정할 때는 플릿 대신 대기열을 지정합니다. 대기열을 사용하면 새 빌드를 포함한 새 플릿을 대기열에 추가하고 기존 플릿을 제거할 수 있습니다. 자세한 내용은 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.
- 별칭을 사용하여 플레이어를 새 게임 빌드로 이전합니다. 게임 클라이언트를 Amazon GameLift와 통합할 때는 플릿 ID 대신 플릿 별칭을 지정합니다. 자세한 내용은 Amazon GameLift 플릿에 별칭 추가 섹션을 참조하세요.
- 자동화된 빌드 업데이트를 설정합니다. Amazon GameLift 배포를 빌드 시스템에 통합하는 방법에 대한 샘플 스크립트와 자세한 내용은 AWS GameTech 블로그의 Automating Deployments to Amazon GameLift를 참조하세요.

빌드 설치 스크립트 추가

게임 빌드의 운영 체제(OS)에 맞게 다음과 같이 설치 스크립트를 생성합니다.

- Windows: `install.bat`이라는 배치 파일을 생성합니다.
- Linux의: `install.sh`라는 쉘 스크립트 파일을 생성합니다.

설치 스크립트를 생성할 때는 다음 사항을 염두합니다.

- 스크립트는 어떤 사용자 입력도 가져올 수 없습니다.
- Amazon GameLift는 빌드를 설치하고 다음 위치의 호스팅 서버에 빌드 패키지의 파일 디렉터리를 다시 생성합니다.
  - Windows 플릿: `C:\game`
  - Linux 플릿: `/local/game`
- Linux 플릿에 대한 설치 프로세스 중에 사용자로 실행을 통해 인스턴스 파일 구조에 대한 액세스를 제한했습니다. 이 사용자는 빌드 파일이 설치된 디렉터리에 대한 전체 권한을 가집니다. 설치 스크립트가 관리자 권한이 필요한 작업을 수행하는 경우 sudo를 사용하여 관리자 액세스를 지정합니다. Windows 플릿용 사용자로 실행을 통해 기본적으로 관리자 권한을 가집니다. 설치 스크립트와 관련해 권한 실패가 발생하면 스크립트에 문제가 있음을 나타내는 이벤트 메시지가 생성됩니다.
- Linux에서 Amazon GameLift는 bash 같은 공통 쉘 인터프리터 언어를 지원합니다. 설치 스크립트 맨 위에 shebang(예: `#!/bin/bash`)을 추가합니다. 선호하는 쉘 명령이 지원되는지 확인하려면 활성 Linux 인스턴스에 원격으로 액세스하여 쉘 프롬프트를 엽니다. 자세한 내용은 Amazon GameLift 플릿 인스턴스에 원격으로 연결 섹션을 참조하세요.
- 설치 스크립트는 VPC 피어링 연결을 사용할 수 없습니다. VPC 피어링 연결은 Amazon GameLift가 플릿 인스턴스에 빌드를 설치하기 전까지는 사용할 수 없습니다.

Example Windows 설치 bash 파일

여기 나온 `install.bat` 파일은 게임 서버에 필요한 Visual C++ 런타임 구성 요소를 설치하고 로그 파일에 결과를 기록하는 방법을 보여줍니다. 스크립트는 루트의 빌드 패키지에 구성 요소 파일을 포함합니다.

```
vcredist_x64.exe /install /quiet /norestart /log c:\game\vcredist_2013_x64.log
```

Example Linux 설치 쉘 스크립트

이 예제 `install.sh` 파일은 설치 스크립트에서 bash를 사용하고 결과를 로그 파일에 기록합니다.

```
#!/bin/bash
echo 'Hello World' > install.log
```

이 예제 `install.sh` 파일은 Amazon CloudWatch 에이전트를 사용하여 시스템 수준 및 사용자 지정 지표를 수집하고 로그 순환을 처리하는 방법을 보여줍니다. Amazon GameLift는 서비스 VPC에서 실

행되므로 사용자를 대신하여 AWS Identity and Access Management(IAM) 역할을 맡을 수 있는 권한을 Amazon GameLift에 부여해야 합니다. Amazon GameLift가 역할을 맡도록 허용하려면 AWS 관리형 정책 `CloudWatchAgentAdminPolicy`를 포함하는 역할을 생성하고 플릿을 생성할 때 해당 역할을 사용합니다.

```
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
```

Amazon GameLift에 Realtime 서버 스크립트 업로드

게임에 Realtime 서버를 배포할 준비가 되면 완성된 Realtime 서버 스크립트 파일을 Amazon GameLift에 업로드합니다. Amazon GameLift 스크립트 리소스를 생성하고 스크립트 파일의 위치를 지정하여 이를 수행합니다. 기존 스크립트 리소스의 새 파일을 업로드하여 이미 배포된 서버 스크립트 파일을 업데이트할 수도 있습니다.

새 스크립트 리소스를 생성하면 Amazon GameLift는 해당 리소스에 고유한 스크립트 ID(예:`script-1111aaaa-22bb-33cc-44dd-5555eeee66ff`)를 할당하고 스크립트 파일의 사본을 업로드합니다. 업로드 시간은 스크립트 파일의 크기와 연결 속도에 따라 달라집니다.

스크립트 리소스를 생성한 후 Amazon GameLift는 새 Realtime 서버 플릿과 함께 스크립트를 배포합니다. Amazon GameLift는 서버 스크립트를 플릿의 각 인스턴스에 설치하고 `/local/game`에 스크립트 파일을 배치합니다.

서버 스크립트와 관련된 플릿 활성화 문제를 해결하려면 Amazon GameLift 플릿 문제 디버깅 섹션을 참조하세요.

스크립트 파일 패키징

서버 스크립트에는 업로드를 위한 단일.zip 파일로 결합된 하나 이상의 파일이 포함될 수 있습니다. .zip 파일에는 스크립트를 실행하는 데 필요한 모든 파일이 포함되어야 합니다.

압축된 스크립트 파일은 로컬 파일 디렉터리나 Amazon Simple Storage Service(S3) 버킷에 저장할 수 있습니다.

로컬 디렉터리에서 스크립트 파일을 업로드합니다.

스크립트 파일을 로컬에 저장한 경우 해당 파일을 Amazon GameLift로 업로드할 수 있습니다. 스크립트 리소스를 생성하려면 Amazon GameLift 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용합니다.

Amazon GameLift console

스크립트 리소스를 생성하려면

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창에서 호스팅, 스크립트를 선택합니다.
3. 스크립트 페이지에서 스크립트 생성을 선택합니다.
4. 스크립트 생성 페이지의 스크립트 설정에서 다음을 수행하세요.
   a. 이름에 스크립트 이름을 입력합니다.
   b. (선택 사항) 버전에 버전 정보를 입력합니다. 스크립트의 내용이 업데이트될 수 있으므로 버전 데이터를 사용하여 업데이트를 추적하면 유용합니다.
   c. 스크립트 소스의 경우.zip 파일 업로드를 선택합니다.
   d. 스크립트 파일의 경우 파일 선택을 선택하고 스크립트가 포함된 .zip 파일을 찾은 다음 해당 파일을 선택합니다.
5. (선택 사항) 태그에서 키 및 값 쌍을 입력하여 스크립트에 태그를 추가합니다.
6. 생성을 선택합니다.

   Amazon GameLift는 새 스크립트에 ID를 할당하고 지정된.zip 파일을 업로드합니다. 스크립트 페이지에서 상태를 비롯한 새 스크립트를 볼 수 있습니다.

AWS CLI

`create-script` AWS CLI 명령을 사용하여 새로운 스크립트를 정의하고 서버 스크립트 파일을 업로드합니다.

스크립트 리소스를 생성하려면

1. .zip 파일을 AWS CLI를 사용할 수 있는 디렉터리에 넣습니다.
2. 명령줄 창을 열고 .zip 파일을 배치한 디렉터리로 전환합니다.
3. 다음 create-script 명령 및 파라미터를 입력합니다. --zip-file 파라미터에서 .zip 파일 이름에 `fileb://` 문자열을 추가해야 합니다. Amazon GameLift가 압축된 콘텐츠를 처리할 수 있도록 파일을 바이너리로 식별합니다.

```
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
```

예

```
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
```

요청에 대한 응답으로 Amazon GameLift가 새로운 스크립트 객체를 반환합니다.

4. 새 스크립트를 보려면 describe-script를 호출합니다.

Amazon S3에서 스크립트 파일 업로드

Amazon S3 버킷에 스크립트 파일을 저장하고 여기에서 Amazon GameLift에 업로드할 수 있습니다. 스크립트를 생성할 때, S3 버킷 위치를 지정하면 Amazon GameLift가 Amazon S3에서 해당 스크립트 파일을 검색합니다.

스크립트 리소스를 생성하려면

1. S3 버킷에 스크립트 파일을 저장합니다. 서버 스크립트 파일이 포함된 .zip 파일을 생성하고 제어하는 AWS 계정의 S3 버킷에 업로드합니다. 객체 URI를 기록해 둡니다. Amazon GameLift 스크립트를 생성할 때 필요합니다.

   > **Note**
   >
   > Amazon GameLift는 이름에 점(.)이 포함된 S3 버킷에서 업로드하는 것을 지원하지 않습니다.

2. 스크립트 파일에 Amazon GameLift 액세스 권한을 부여합니다. Amazon GameLift가 서버 스크립트를 포함하는 S3 버킷에 액세스할 수 있도록 허용하는 AWS Identity and Access Management(IAM) 역할을 생성하려면 Amazon에 대한 IAM 서비스 역할 설정 GameLift의 지침을 수행합니다. 새 역할을 생성한 후에는 스크립트 생성 시 필요한 이름을 기록해 둡니다.
3. 스크립트를 생성합니다. Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 새 스크립트 레코드를 생성합니다. 이 요청을 수행하려면 Amazon GameLift에 사용되는 IAM 권한 예제에 설명된 대로 IAM `PassRole` 권한이 있어야 합니다.

Amazon GameLift console

1. Amazon GameLift 콘솔의 탐색 창에서 호스팅, 스크립트를 선택합니다.
2. 스크립트 페이지에서 스크립트 생성을 선택합니다.
3. 스크립트 생성 페이지의 스크립트 설정에서 다음을 수행하세요.
   a. 이름에 스크립트 이름을 입력합니다.
   b. (선택 사항) 버전에 버전 정보를 입력합니다. 스크립트의 내용이 업데이트될 수 있으므로 버전 데이터를 사용하여 업데이트를 추적하면 유용합니다.
   c. 스크립트 소스의 경우 Amazon S3 URI를 선택합니다.
   d. 게임 서버 빌드에는 Amazon S3에 업로드한 스크립트 객체의 S3 URI를 입력한 다음 객체 버전을 선택합니다. Amazon S3 URI와 객체 버전이 기억나지 않는 경우, S3 찾아보기를 선택한 다음 스크립트 객체를 검색합니다.
4. (선택 사항) 태그에서 키 및 값 쌍을 입력하여 스크립트에 태그를 추가합니다.
5. 생성을 선택합니다.

   Amazon GameLift는 새 스크립트에 ID를 할당하고 지정된.zip 파일을 업로드합니다. 스크립트 페이지에서 상태를 비롯한 새 스크립트를 볼 수 있습니다.

AWS CLI

`create-script` AWS CLI 명령을 사용하여 새로운 스크립트를 정의하고 서버 스크립트 파일을 업로드합니다.

1. 명령줄 창을 열고 AWS CLI를 사용할 수 있는 디렉터리로 전환합니다.
2. 다음 create-script 명령 및 파라미터를 입력합니다. --storage-location 파라미터는 스크립트 파일의 Amazon S3 버킷 위치를 지정합니다.

```
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
```

   예

```
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
```

요청에 대한 응답으로 Amazon GameLift가 새로운 스크립트 객체를 반환합니다.

3. 새 스크립트를 보려면 describe-script를 호출합니다.

스크립트 파일 업데이트

Amazon GameLift 콘솔 또는 update-scriptAWS CLI 명령을 사용하여 스크립트 리소스의 메타데이터를 업데이트할 수 있습니다.

스크립트 리소스의 스크립트 콘텐츠를 업데이트할 수도 있습니다. Amazon GameLift는 업데이트된 스크립트 리소스를 사용하는 모든 플릿 인스턴스에 스크립트 콘텐츠를 배포합니다. 업데이트된 스크립트가 배포되면 인스턴스는 새 게임 세션을 시작할 때 해당 스크립트를 사용합니다. 업데이트 당시 이미 실행 중인 게임 세션은 업데이트된 스크립트를 사용하지 않습니다.

스크립트 파일을 업데이트하려면

- 로컬에 저장된 스크립트 파일의 경우 업데이트된 스크립트 .zip 파일을 업로드하려면 Amazon GameLift 콘솔 또는 update-script 명령을 사용합니다.
- Amazon S3 버킷에 저장된 스크립트 파일의 경우 업데이트된 스크립트 파일을 S3 버킷에 업로드합니다. Amazon GameLift는 정기적으로 업데이트된 스크립트 파일을 확인하고 S3 버킷에서 직접 해당 파일을 검색합니다.

Amazon GameLift 플릿 설정

이 섹션에서는 Amazon GameLift와 함께 사용하기 위한 플릿의 설계, 구축 및 관리에 대한 상세한 정보를 제공합니다. Amazon GameLift 플릿을 사용하여 사용자 지정 게임 서버 및 Realtime 서버를 배포할 수 있습니다.

플릿은 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 또는 물리적 하드웨어 세트로 구성된 호스팅 리소스를 나타냅니다. 플릿의 위치에 따라 플레이어의 게임 세션을 호스팅하기 위해 인스턴스

또는 하드웨어를 배포할 위치가 결정됩니다. 플릿의 크기, 지원할 수 있는 게임 세션 및 플레이어 수는 플릿에 제공하는 인스턴스 수 또는 하드웨어 양에 따라 달라집니다. 수동으로 또는 Auto Scaling을 사용하여 가상 인스턴스를 조정할 수 있습니다.

프로덕션 환경에서 대부분의 게임은 여러 개의 플릿을 사용합니다. 예를 들어 여러 플릿을 사용하여 두 개 이상의 게임 서버 버전을 동시에 실행하거나, 스팟 플릿에 백업 용량을 제공하거나, 중복성을 구축할 수 있습니다.

게임의 요구 사항에 맞게 설계된 플릿을 만드는 방법을 알아보려면 Amazon GameLift 플릿 설계 가이드 섹션부터 시작하세요. 플릿을 실행한 후에는 Amazon GameLift 호스팅 용량 확장, Amazon GameLift 플릿에 별칭 추가, 및 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.

주제

- Amazon GameLift 플릿 설계 가이드
- 새 Amazon GameLift 플릿 생성
- Amazon GameLift 플릿 관리
- Amazon GameLift 플릿에 별칭 추가
- Amazon GameLift 플릿 문제 디버깅
- Amazon GameLift 플릿 인스턴스에 원격으로 연결

Amazon GameLift 플릿 설계 가이드

이 설계 가이드에서는 Amazon GameLift와 함께 사용할 호스팅 리소스 플릿을 생성하는 모범 사례를 다룹니다. 호스팅 리소스의 조합을 선택하고 게임에 적합하게 구성하는 방법을 알아보세요.

주제

- Amazon GameLift 컴퓨팅 리소스 선택
- Amazon GameLift가 게임 서버를 시작하는 방법 관리
- Amazon에서 스팟 인스턴스 사용 GameLift

Amazon GameLift 컴퓨팅 리소스 선택

게임 서버를 배포하고 플레이어를 위한 게임 세션을 호스팅하기 위해 Amazon은 인스턴스라고 하는 Amazon Elastic Compute Cloud (Amazon EC2) 리소스 또는 물리적 하드웨어를 GameLift 사용합니다. 인스턴스를 사용하여 새로운 플릿을 설정할 때 필요한 인스턴스 유형과 게임 서버 프로세스를 실행하

는 방법을 결정합니다. 관리형 EC2 플릿이 활성 상태이고 게임 세션을 호스팅할 준비가 되면 필요에 따라 인스턴스를 추가하거나 제거하여 플레이어 수요를 수용할 수 있습니다.

Amazon GameLift 게임 서버를 두 가지 컴퓨팅 유형의 조합으로 배포할 수 있습니다.

- 관리형 EC2 - 관리형 EC2 플릿은 Amazon EC2 인스턴스를 사용하여 게임 서버를 호스팅합니다. Amazon은 인스턴스를 GameLift 관리하고 게임 호스팅에 따른 하드웨어 및 소프트웨어 관리 부담을 없애줍니다.
- Amazon GameLift Anywhere — Amazon GameLift Anywhere 플릿은 기존 인프라를 사용하여 게임 서버를 호스팅하고 Amazon은 매치메이킹과 대기열을 GameLift 관리합니다.

플릿의 컴퓨팅 리소스를 선택할 때는 다음 요소를 고려해야 합니다.

- 사용 가능한 하드웨어
- 플릿 위치
- 온디맨드 인스턴스 및 스팟 인스턴스 비교
- 운영 체제
- 인스턴스 타입
- Service quotas

사용 가능한 하드웨어

구현 시 기존 인프라를 고려합니다. 게임을 GameLift Amazon으로 마이그레이션하는 동안에도 인프라를 계속 사용할 수 있습니다. Amazon을 사용하면 Amazon GameLift Anywhere GameLift 관리형 EC2 인스턴스와 함께 자체 인프라를 사용할 수 있습니다. 또한 기존 인프라를 사용하여 지원되는 Amazon GameLift 위치보다 가까운 곳에서 플레이어와 게임을 호스팅할 수 있습니다. Amazon GameLift Anywhere 플릿 설정에 대한 자세한 내용은 을 참조하십시오아마존 GameLift Anywhere 플릿 생성.

플릿 위치

게임 서버를 배포하려는 지리적 위치를 고려합니다. 인스턴스 유형 가용성은 AWS 리전 및 로컬 영역에 따라 다릅니다.

다중 위치 플릿의 경우 인스턴스 가용성 및 할당량은 플릿의 홈 리전과 선택된 원격 위치의 조합에 따라 달라집니다. 플릿 위치에 대한 자세한 내용은 아마존 GameLift 호스팅 위치 섹션을 참조하세요.

Amazon GameLift Anywhere 플릿의 경우 물리적 하드웨어의 위치를 결정합니다. 사용자 지정 위치에 대한 자세한 내용은 아마존 GameLift Anywhere 섹션을 참조하세요.

온디맨드 인스턴스 및 스팟 인스턴스 비교

Amazon EC2 온디맨드 인스턴스 및 스팟 인스턴스는 동일한 하드웨어와 성능을 제공하지만 가용성과 비용이 다릅니다.

온디맨드 인스턴스

필요할 때 언제든지 온디맨드 인스턴스를 획득하고 원하는 만큼 유지할 수 있습니다. 온디맨드 인스턴스는 고정 비용입니다. 즉, 사용한 시간에 따라 비용을 지불하며 장기 약정은 없습니다.

스팟 인스턴스

스팟 인스턴스는 AWS 미사용 컴퓨팅 용량을 활용하여 온디맨드 인스턴스에 대한 비용 효율적인 대안을 제공할 수 있습니다. 스팟 인스턴스 가격은 각 위치의 각 인스턴스 유형에 대한 수요와 공급에 따라 변동됩니다. AWS 용량을 다시 확보해야 할 때마다 스팟 인스턴스가 중단될 수 있습니다. GameLift Amazon은 대기열과 FleetiQ 알고리즘을 사용하여 스팟 인스턴스를 중단할지 결정하고 인스턴스를 재활용 상태로 전환합니다. AWS 그러면 인스턴스에 활성 게임 세션이 없는 경우 Amazon은 인스턴스를 GameLift 교체하려고 합니다.

스팟 인스턴스를 사용하는 방법에 대한 자세한 정보는 Amazon에서 스팟 인스턴스 사용 GameLift 섹션을 참조하세요.

운영 체제

Amazon GameLift 인스턴스는 마이크로소프트 윈도우 또는 아마존 리눅스에서 실행되는 게임 서버 빌드를 지원합니다. GameLiftAmazon에 게임 빌드를 업로드할 때 게임의 운영 체제를 지정하십시오. Amazon EC2 플릿을 생성하여 게임 빌드를 배포하면 Amazon은 해당 빌드의 운영 체제로 인스턴스를 GameLift 자동으로 설정합니다. 지원되는 게임 서버 운영 체제에 대한 자세한 내용은 아마존을 통한 개발 지원 GameLift 섹션을 참조하세요.

Amazon GameLift Anywhere 플릿을 사용하는 경우 하드웨어가 지원하는 모든 운영 체제를 사용할 수 있습니다. Amazon GameLift Anywhere 플릿을 사용하려면 게임 빌드를 하드웨어에 배포하고 Amazon을 사용하여 리소스를 한 곳에서 GameLift 관리해야 합니다.

인스턴스 타입

Amazon EC2 플릿의 인스턴스 유형에 따라 인스턴스에 사용할 하드웨어 종류가 결정됩니다. 다른 인스턴스 유형은 컴퓨팅 성능, 메모리, 스토리지 및 네트워킹 기능의 다양한 조합을 제공합니다.

게임에 사용할 수 있는 인스턴스 유형을 선택할 때는 다음 사항을 고려합니다.

- 게임 서버의 컴퓨팅 아키텍처는 x64 또는 Arm (AWS Graviton) 입니다.

> **Note**
>
> Graviton Arm 인스턴스를 사용하려면 리눅스 OS에 Amazon GameLift 서버를 빌드해야 합니다. C++ 및 C#에는 Server SDK 5.1.1 이상이 필요합니다. Go에는 Server SDK 5.0 이상이 필요합니다. 이러한 인스턴스는 아마존 리눅스 2023 (AL2023) 또는 아마존 리눅스 2 (AL2)에서의 모노 설치를 out-of-the-box 지원하지 않습니다.

- 게임 서버 빌드의 컴퓨팅, 메모리 및 스토리지 요구 사항입니다.
- 인스턴스당 실행하기로 결정한 서버 프로세스의 수입니다.

더 큰 인스턴스 유형을 사용하면 각 인스턴스에서 여러 서버 프로세스를 실행할 수 있습니다. 이렇게 하면 플레이어 수요를 충족하는 데 필요한 인스턴스 수를 줄일 수 있습니다.

자세한 내용:

- 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 인스턴스 유형을 참조하세요.
- 인스턴스당 여러 프로세스를 실행하는 방법에 대해서는 Amazon GameLift가 게임 서버를 시작하는 방법 관리 섹션을 참조하세요.

Service quotas

Amazon의 기본 서비스 할당량과 현재 Amazon GameLift 서비스 할당량을 보려면 다음을 수행하십시오 AWS 계정.

- GameLiftAmazon에 대한 일반적인 서비스 할당량 정보는 의 Amazon GameLift 엔드포인트 및 할당량을 참조하십시오. AWS 일반 참조
- 위치별로 계정에 사용할 수 있는 인스턴스 유형 목록을 보려면 Amazon GameLift 콘솔의 서비스 할당량 페이지를 여십시오. 또한 이 페이지에는 각 위치의 각 인스턴스 유형에 대한 계정의 현재 사용량도 표시됩니다.
- 지역별 인스턴스 유형에 대한 계정의 현재 할당량 목록을 보려면 () 명령을 실행하십시오. AWS Command Line Interface AWS CLI`describe-ec2-instance-limits` 이 명령은 기본 리전(또는 지정한 다른 리전)에 있는 활성 인스턴스 수를 반환합니다.

게임 출시를 준비하면서 Amazon GameLift 콘솔에서 출시 설문지를 작성하십시오. Amazon GameLift 팀은 출시 설문지를 사용하여 게임의 올바른 할당량 및 한도를 결정합니다.

Amazon GameLift가 게임 서버를 시작하는 방법 관리

인스턴스당 여러 게임 서버 프로세스를 실행하도록 관리형 EC2 플릿의 런타임 구성을 설정할 수 있습니다. 이렇게 하면 호스팅 리소스가 더 효율적으로 사용됩니다.

플릿의 여러 프로세스 관리 방식

Amazon GameLift는 플릿의 런타임 구성을 사용하여 각 인스턴스에서 실행할 프로세스의 유형과 수를 결정합니다. 런타임 구성에는 게임 서버 실행 파일 하나를 나타내는 서버 프로세스 구성이 하나 이상 포함되어 있습니다. 게임과 관련된 다른 유형의 프로세스를 실행하기 위해 추가 서버 프로세스 구성을 정의할 수 있습니다. 각 서버 프로세스 구성에는 다음 정보가 포함됩니다.

- 게임 빌드에서 실행 파일의 파일 이름 및 경로
- (선택 사항) 시작 시 프로세스에 전달할 파라미터
- 동시에 실행할 프로세스 수

플릿의 인스턴스가 활성화되면 런타임 구성에 정의된 서버 프로세스의 집합이 시작됩니다. 여러 프로세스를 사용하는 경우 Amazon GameLift는 각 프로세스의 시작을 시차를 두고 진행합니다. 서버 프로세스의 수명은 제한되어 있습니다. 작업이 끝나면 Amazon GameLift는 런타임 구성에 정의된 서버 프로세스의 수와 유형을 유지하기 위해 새로운 프로세스를 시작합니다.

언제든 서버 프로세스 구성을 추가, 변경 또는 삭제하여 런타임 구성을 변경할 수 있습니다. 각 인스턴스는 플릿의 런타임 구성에 대한 업데이트를 정기적으로 확인하여 변경 사항을 구현합니다. Amazon GameLift가 런타임 구성 변경을 채택하는 방식은 다음과 같습니다.

1. 인스턴스는 Amazon GameLift에 최신 버전의 런타임 구성을 위한 요청을 전송합니다.
2. 인스턴스는 활성 프로세스를 최신 런타임 구성과 비교한 후, 다음을 수행합니다.
   - 업데이트된 런타임 구성에서 서버 프로세스 유형을 제거하면, 이 유형의 활성 서버 프로세스는 종료될 때까지 계속 실행됩니다. 인스턴스는 이러한 서버 프로세스를 대체하지 않습니다.
   - 업데이트된 런타임 구성에서 서버 프로세스 유형에 대한 동시 프로세스 수를 줄이면, 이 유형의 초과 서버 프로세스는 종료될 때까지 계속 실행됩니다. 인스턴스는 이러한 초과 서버 프로세스를 대체하지 않습니다.
   - 업데이트된 런타임 구성에서 새 서버 프로세스 유형을 추가하거나 기존 유형에 대한 동시 프로세스를 증가시키면, 인스턴스는 최대 Amazon GameLift를 기준으로 새 서버 프로세스를 시작합니다. 이 경우 기존 프로세스가 종료되면 인스턴스가 새 서버 프로세스를 시작합니다.

여러 프로세스에 대한 플릿 최적화

플릿에서 여러 프로세스를 사용하려면 다음을 수행합니다.

- 플릿에 배포하려는 게임 서버 실행 파일이 포함된 빌드를 생성한 다음 Amazon GameLift에 빌드를 업로드합니다. 빌드의 모든 게임 서버는 동일한 플랫폼에서 실행되어야 하며 Amazon GameLift Server SDK를 사용해야 합니다.
- 하나 이상의 서버 프로세스 구성과 여러 개의 동시 프로세스로 이루어진 런타임 구성을 생성합니다.
- 게임 클라이언트를 AWS SDK 버전 2016-08-04 이상과 통합합니다.

플릿 성능을 최적화하려면 다음을 수행하는 것이 좋습니다.

- 서버 프로세스 종료 시나리오를 처리하여 Amazon GameLift가 프로세스를 효율적으로 재활용할 수 있도록 합니다. 예:
  - 서버 API `ProcessEnding()`을 호출하는 게임 서버 코드에 종료 절차를 추가합니다.
  - 게임 서버 코드에 `OnProcessTerminate()` 콜백 함수를 구현하여 Amazon GameLift의 종료 요청을 처리합니다.
- Amazon GameLift가 비정상 서버 프로세스가 종료하고 다시 시작하는지 확인합니다. 게임 서버 코드에 `OnHealthCheck()` 콜백 함수를 구현하여 상태를 Amazon GameLift에 다시 보고합니다. Amazon GameLift는 3개의 연속된 보고서에서 비정상으로 보고된 서버 프로세스를 자동으로 종료합니다. `OnHealthCheck()`를 구현하지 않으면, 프로세스가 통신에 응답하지 않는 한 Amazon GameLift는 서버 프로세스를 정상이라고 가정합니다.

인스턴스당 프로세스 수 선택

인스턴스에서 실행할 동시 프로세스 수를 결정할 때 다음과 같이 유념해야 합니다.

- Amazon GameLift는 각 인스턴스에 지정할 수 있는 최대 동시 프로세스 수를 제한합니다. 플릿의 서버 프로세스 구성에 대한 모든 동시 프로세스의 합계는 이 할당량을 초과할 수 없습니다.
- 허용 가능한 성능 수준을 유지하려면 Amazon EC2 인스턴스 유형으로 동시에 실행할 수 있는 프로세스 수를 제한할 수 있습니다. 게임에 대해 다양한 구성을 테스트하여 선호하는 인스턴스 유형에서 올바른 프로세스 수를 찾습니다.
- Amazon GameLift는 구성된 전체 프로세스 수보다 많은 동시 프로세스를 실행하지 않습니다. 즉, 이전 런타임 구성에서 새 구성으로의 전환이 점진적으로 이루어질 수 있습니다.

Amazon에서 스팟 인스턴스 사용 GameLift

Amazon GameLift 관리형 EC2 플릿을 설정할 때 스팟 인스턴스, 온디맨드 인스턴스 또는 이 둘을 조합하여 사용할 수 있습니다. Amazon이 스팟 인스턴스를 GameLift 사용하는 방법에 대해 자세히 알아보십시오온디맨드 인스턴스 및 스팟 인스턴스 비교. 스팟 플릿을 사용하려면 게임 통합에 이 페이지에 나열된 조정이 필요합니다.

FlexMatch 매치메이킹에 사용하고 계신가요? 매치메이킹 배치를 위해 기존 게임 세션 대기열에 스팟 플릿을 추가할 수 있습니다.

1. 스팟 인스턴스용 게임 세션 대기열을 설계합니다.

   대기열을 사용하여 게임 세션 배치를 관리하는 것이 가장 좋은 방법이지만 스팟 인스턴스를 사용할 때는 필수 사항입니다. 대기열을 설계하려면 다음 사항을 고려해야 합니다.

   - 위치 - 플레이어 경험을 최적화하려면 플레이어와 지리적으로 가까운 위치를 선택합니다.
   - 인스턴스 유형 - 게임 서버의 하드웨어 요구 사항과 선택한 위치의 인스턴스 가용성을 고려합니다.

   스팟 가용성 및 복원력을 최적화하는 대기열을 사용해 보려면 자습서: 스팟 인스턴스용 게임 세션 대기열 설치 섹션을 참조하세요.
2. 스팟 최적화 대기열에 플릿을 생성합니다.

   대기열 설계에 따라 플릿을 생성하여 원하는 위치 및 인스턴스 유형에 게임 서버를 배포합니다. 새 플릿의 생성 및 구성에 대한 자세한 내용은 Amazon GameLift 관리형 플릿 생성 섹션을 참조하세요.
3. 게임 세션 대기열을 생성합니다.

   플릿 목적지를 추가하고, 게임 세션 배치 프로세스를 구성하며, 배치 우선 순위를 정의합니다. 새 대기열의 생성 및 구성에 대한 자세한 내용은 게임 세션 대기열 생성 섹션을 참조하세요.
4. 대기열을 사용하도록 게임 클라이언트 서비스를 업데이트합니다.

   게임 클라이언트가 대기열을 사용하여 리소스를 요청하면 대기열이 중단될 가능성이 높은 리소스를 피하고 정의된 우선 순위와 일치하는 위치를 선택합니다. 게임 클라이언트에서 게임 세션 배치를 구현하는 방법은 게임 세션 만들기 섹션을 참조하세요.
5. 게임 서버가 스팟 중단을 업데이트합니다.

AWS 용량을 다시 확보해야 하는 경우 2분 알림으로 스팟 인스턴스를 중단할 수 있습니다. 방해 요소를 처리하도록 게임 서버를 설정하여 플레이어에게 미치는 영향을 최소화합니다.

스팟 인스턴스를 AWS 회수하기 전에 종료 알림을 보냅니다. Amazon은 Amazon Server SDK 콜백 함수를 호출하여 영향을 받는 모든 GameLift 서버 프로세스에 알림을 GameLift 전달합니다. `onProcessTerminate()` 이 콜백을 구현하여 게임 세션을 종료하거나 게임 세션과 플레이어를 새 인스턴스로 이동합니다. `onProcessTerminate()` 구현에 대한 자세한 내용은 서버 프로세스 종료 알림에 응답 섹션을 참조하세요.

> **Note**
>
> AWS 인스턴스를 회수하기 전에 알림을 제공하기 위해 모든 노력을 기울이지만 경고가 도착하기 전에 스팟 인스턴스를 AWS 회수할 수도 있습니다. 게임 서버를 준비하여 예기치 않은 중단을 처리합니다.

6. 스팟 플릿 및 대기열의 성능을 검토합니다.

   Amazon GameLift 콘솔 또는 Amazon에서 Amazon GameLift 지표를 보고 성능을 CloudWatch 검토할 수 있습니다. Amazon GameLift 지표에 대한 자세한 내용은 을 참조하십시오Amazon CloudWatch를 사용한 Amazon GameLift 모니터링. 주요 지표는 다음과 같습니다.

   - 중단율 - `InstanceInterruptions` 및 `GameSessionInterruptions` 지표를 사용하여 인스턴스 및 게임 세션에 대한 스팟 관련 중단의 횟수와 빈도를 추적합니다. 가 회수한 게임 세션의 상태는 `TERMINATED` 이고 상태 이유는 입니다. AWS `INTERRUPTED`
   - 대기열 효율성 - 배치 성공률, 평균 대기 시간 및 대기열 깊이를 추적하여 스팟 플릿 사용이 대기열 성능에 영향을 미치지 않는지 확인합니다.
   - 플릿 사용 - 인스턴스, 게임 세션, 플레이어 세션에 대한 데이터를 모니터링합니다. 온디맨드 플릿의 사용량은 중단 상황이 발생하지 않도록 대기열이 스팟 플릿에 배치되는 것을 피하고 있다는 지표가 될 수 있습니다.

새 Amazon GameLift 플릿 생성

새 플릿을 만들고 사용자 지정 게임 서버 빌드나 호스팅용 Realtime 서버를 배포합니다. Amazon GameLift에 업로드한 모든 게임 빌드 또는 스크립트 리소스를 배포할 수 있습니다.

주제

- Amazon GameLift 플릿 생성 작동 방식

- Amazon GameLift 관리형 플릿 생성
- 아마존 GameLift Anywhere 플릿 생성

Amazon GameLift 플릿 생성 작동 방식

새 플릿을 생성하면 Amazon GameLift에서 각 플릿 위치에 하나의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스가 있는 플릿을 생성하는 워크플로를 시작합니다. Amazon GameLift가 워크플로의 각 단계를 완료하면 플릿에서 이벤트가 발생하고 Amazon GameLift가 플릿 상태를 업데이트합니다. Amazon GameLift 콘솔을 사용하거나 Amazon GameLift API 작업 DescribeFleetEvents를 호출하여 모든 이벤트를 추적할 수 있습니다. 또한 DescribeFleetLocationAttributes를 사용하여 개별 위치의 상태를 추적할 수 있습니다.

EC2 플릿 생성 워크플로:

- Amazon GameLift는 플릿의 홈 리전과 플릿에 정의된 각 원격 위치에 플릿 리소스를 생성합니다.
- Amazon GameLift는 원하는 용량을 1개의 인스턴스로 설정합니다.
- Amazon GameLift는 플릿 및 위치 상태를 신규로 설정합니다.
- Amazon GameLift는 플릿 이벤트 로그에 이벤트를 기록하기 시작합니다.
- Amazon GameLift는 요청된 컴퓨팅 리소스를 각 플릿 위치의 새 인스턴스 하나에 할당합니다.
- Amazon GameLift는 게임 서버 파일을 각 인스턴스에 다운로드하고 플릿 상태를 다운로드 중으로 설정합니다.
- Amazon GameLift는 각 인스턴스에서 다운로드된 게임 서버 파일을 검증하여 다운로드 중에 오류가 발생하지 않았는지 확인합니다. Amazon GameLift는 플릿 상태를 검증 중으로 설정합니다.
- Amazon GameLift는 각 인스턴스에 게임 서버를 구축하고 플릿 상태를 빌드 중으로 설정합니다.
- Amazon GameLift에서 플릿의 런타임 구성 지침에 따라 각 인스턴스에서 서버 프로세스 실행을 시작합니다. 인스턴스당 여러 동시 실행 서버 프로세스를 실행하도록 플릿을 구성한 경우 Amazon GameLift는 프로세스 시작 시간을 몇 초씩 지연시킵니다. 각 프로세스가 온라인 상태가 되면 Amazon GameLift에 준비 상태를 다시 보고합니다. Amazon GameLift는 플릿 상태를 활성화 중으로 설정합니다.
- Amazon GameLift는 서버가 보고서 준비를 처리함에 따라 플릿 상태와 위치 상태를 활성으로 설정합니다.

Amazon GameLift Anywhere fleet creation

- Amazon GameLift는 플릿 리소스를 생성합니다. 플릿의 홈 리전 및 플릿에 정의된 각 사용자 지정 위치의 경우 Amazon GameLift는 플릿 및 위치 상태를 신규로 설정합니다.
- Amazon GameLift는 플릿 이벤트 로그에 이벤트를 기록하기 시작합니다.
- 플릿의 한 서버 프로세스가 Amazon GameLift에 준비가 완료되었음을 알리면 Amazon GameLift는 플릿 상태와 위치 상태를 활성으로 설정합니다. 다른 플릿 위치에서 서버가 보고서 준비를 처리하므로 Amazon GameLift는 각 플릿 위치 상태를 활성으로 설정합니다.

플릿 생성 문제에 도움을 받으려면 Amazon GameLift 플릿 문제 디버깅 섹션을 참조하세요.

Amazon GameLift 관리형 플릿 생성

Amazon GameLift 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 관리형 플릿을 생성합니다.

새 관리형 EC2 플릿을 만든 후에는 Amazon GameLift가 플릿을 배포하고, 게임 서버를 설치하며, 시작하기 때문에 플릿 상태가 여러 단계를 거치게 됩니다. 플릿이 `ACTIVE` 상태가 되면 플릿은 게임 세션을 호스팅할 준비가 된 것입니다. 플릿 생성 문제에 도움을 받으려면 Amazon GameLift 플릿 문제 디버깅 단원을 참조하십시오.

Console

관리형 EC2 플릿을 생성하려면

1. Amazon GameLift 콘솔의 탐색 창에서 플릿을 선택합니다.
2. 플릿 페이지에서 플릿 생성을 선택합니다.
3. 관리형 EC2를 선택합니다.
4. 플릿 세부 정보 페이지에서 다음을 수행합니다.
    a. 이름에 플릿 이름을 입력합니다. 플릿 이름에 플릿 유형(스팟 또는 온디맨드)을 포함하는 것이 좋습니다. 이렇게 하면 플릿 목록을 볼 때 플릿 유형을 훨씬 쉽게 식별할 수 있습니다.
    b. 설명에는 플릿에 대한 간략한 설명을 입력합니다.
    c. 바이너리 형식에는 빌드 또는 스크립트를 선택하여 Amazon GameLift가 이 플릿에 배포하는 게임 서버 유형을 정의합니다.
    d. 업로드된 스크립트 또는 빌드의 드롭다운 목록에서 스크립트 또는 빌드를 선택합니다.
5. (선택 사항) 추가 세부 정보에서 다음을 수행합니다.

a. 인스턴스 역할에는 게임 빌드의 애플리케이션이 계정의 다른 AWS 리소스에 액세스할 수 있도록 승인하는 IAM 역할을 지정합니다. 자세한 내용은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요. 인스턴스 역할을 사용하여 플릿을 생성하려면 계정에 IAM `PassRole` 권한이 있어야 합니다. 자세한 내용은 Amazon GameLift에 사용되는 IAM 권한 예제 섹션을 참조하세요.

   CloudWatch 에이전트와 같이 서버 실행 파일이 아닌 애플리케이션을 승인하려면 공유 자격 증명 옵션을 활성화합니다.

   플릿 생성 후에는 이 설정을 업데이트할 수 없습니다.

b. 인증 생성에는 플릿에 대한 Amazon GameLift TLS 인증서 생성을 선택합니다. 연결할 때 플릿 TLS 인증서를 사용하여 게임 클라이언트에서 게임 서버를 인증하고 모든 클라이언트/서버 통신을 암호화할 수 있습니다. TLS 활성화 플릿에 있는 각 인스턴스에 대해 Amazon GameLift는 인증서를 사용하여 새 DNS 항목도 생성합니다. 다음 리소스를 사용하여 게임에 대한 인증 및 암호화를 설정합니다.

c. 지표 그룹에는 신규 또는 기존 플릿 지표 그룹의 이름을 입력합니다. 동일한 지표 그룹에 추가하여 복수 플릿에 대한 지표를 집계할 수 있습니다.

   플릿 생성 후에는 메트릭 그룹을 업데이트할 수 없습니다.

6. 다음을 선택합니다.

7. 위치 선택 페이지에서 인스턴스를 배포할 추가 원격 위치를 하나 이상 선택합니다. 홈 리전은 콘솔에 액세스하는 리전에 따라 자동으로 선택됩니다. 위치를 추가로 선택하면 플릿 인스턴스도 해당 위치에 배포됩니다.

> **Important**
>
> 기본적으로 활성화되지 않은 리전을 사용하려면 AWS 계정에서 활성화합니다.
>
> - 2022년 2월 28일 이전에 생성한 리전 중 활성화되지 않은 리전의 플릿은 영향을 받지 않습니다.
> - 새 다중 위치 플릿을 생성하거나 기존의 다중 위치 플릿을 업데이트하려면 먼저 사용하려는 리전을 모두 활성화해야 합니다.
>
> 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 AWS 리전 관리를 참조하세요.

8. 다음을 선택합니다.
9. 인스턴스 세부 정보 정의 페이지에서

   a. 이 플릿의 온디맨드 또는 스팟 인스턴스를 선택합니다. 플릿 유형에 대한 자세한 내용은 온디맨드 인스턴스 및 스팟 인스턴스 비교 섹션을 참조하세요.

   b. 아키텍처 필터링 메뉴에서 x64 또는 Arm을 선택합니다.

   > **Note**
   >
   > Graviton Arm 인스턴스에는 Linux OS 기반 Amazon GameLift 서버 빌드가 필요합니다. C++ 및 C#에는 Server SDK 5.1.1 이상이 필요합니다. Go에는 Server SDK 5.0 이상이 필요합니다. 이러한 인스턴스는 Amazon Linux 2023(AL2023) 또는 Amazon Linux 2(AL2)에서 모노 설치에 대한 기본 지원을 제공하지 않습니다.

   Amazon EC2 Arm 아키텍처에 대한 자세한 내용은 AWS Graviton 프로세서 및 Amazon EC2 인스턴스 유형을 참조하세요.

   Amazon GameLift에서 지원하는 인스턴스 유형에 대한 자세한 내용은 CreateFleet() 요청 파라미터 아래의 `EC2InstanceType` 값을 참조하세요.
10. 목록에서 Amazon EC2 인스턴스 유형을 선택합니다. 인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 인스턴스 타입 섹션을 참조하세요. 플릿을 생성한 후에는 인스턴스 유형을 변경할 수 없습니다.
11. 다음을 선택합니다.
12. 런타임 구성 페이지의 런타임 구성에서 다음을 수행합니다.

    a. 시작 경로에는 빌드나 스크립트에 게임 실행 파일 경로를 입력합니다. Windows 인스턴스에서 게임 서버는 `C:\game` 경로에 빌드됩니다. Linux 인스턴스에서 게임 서버는 `/local/game`에 빌드됩니다. 예: **`C:\game\MyGame\server.exe`**, **`/local/game/MyGame/server.exe`**, 또는 **`MyRealtimeLaunchScript.js`**.

    b. (선택 사항) 시작 파라미터에는 게임 실행 파일에 전달할 정보를 명령줄 파라미터 세트로 입력합니다. 예: **`+sv_port 33435 +start_lobby`**.

    c. 동시 프로세스에는 플릿의 각 인스턴스에서 동시에 실행할 서버 프로세스 수를 선택합니다. 동시 서버 프로세스 수에 대한 Amazon GameLift 제한을 검토합니다.

인스턴스당 동시 서버 프로세스 한도는 모든 구성에 대한 총 동시 프로세스 수에 적용됩니다. 한도를 초과하도록 플릿을 구성하면 플릿이 활성화되지 않습니다.

13. 게임 세션 활성화의 경우 이 플릿의 인스턴스에서 새 게임 세션을 활성화하기 위한 제한을 제공합니다.

    a. 최대 동시 게임 세션 활성화에는 동시에 활성화할 수 있는 인스턴스의 게임 세션 수를 입력합니다. 이 제한은 여러 개의 새 게임 세션을 시작하면 인스턴스에서 실행 중인 다른 게임 세션의 성능에 영향을 미칠 수 있으므로 유용합니다.

    b. 새 활성화 제한 시간에는 세션이 활성화될 때까지 기다려야 하는 시간을 입력합니다. 게임 세션이 제한 시간 이전의 `ACTIVE` 상태로 이동하지 않으면 Amazon GameLift는 게임 세션 활성화를 종료합니다.

14. (선택 사항) EC2 포트 설정에서 다음을 수행합니다.

    a. 포트 설정 추가를 클릭하여 이 플릿에서 배포된 서버 프로세스에 연결되는 인바운드 트래픽에 대한 액세스 권한을 정의합니다.

    b. 유형에는 사용자 지정 TCP 또는 사용자 지정 UDP를 선택합니다.

    c. 포트 범위에는 인바운드 연결을 허용하는 포트 번호 범위를 입력합니다. 포트 범위는 `nnnnn[-nnnnn]` 형식으로 1026~60000의 값을 사용해야 합니다. 예: **`1500`** 또는 **`1500-20000`**.

    d. IP 주소 범위에는 IP 주소의 범위를 입력합니다. CIDR 표기법을 사용합니다. 예: **`0.0.0.0/0`** (이 예는 연결을 시도하는 모든 사람에게 액세스를 허용합니다.)

15. (선택 사항) 게임 세션 리소스 설정에서 다음을 수행합니다.

    a. 게임 크기 조정 보호 정책의 경우 크기 조정 보호를 켜거나 끕니다. Amazon GameLift는 활성 게임 세션을 호스팅하고 있는 경우, 스케일 다운 이벤트 중에 보호를 적용한 인스턴스를 종료하지 않습니다.

    b. 리소스 생성 한도에는 정책 기간 동안 플레이어가 생성할 수 있는 최대 게임 세션 수를 입력합니다.

16. 다음을 선택합니다.

17. (선택 사항) 키 및 값 쌍을 입력하여 빌드에 태그를 추가합니다. 플릿 생성 검토를 계속하려면 다음을 선택합니다.

18. 생성을 선택합니다. Amazon GameLift가 새 플릿에 ID를 할당하고 플릿 활성화 프로세스를 시작합니다. 플릿 페이지에서 새 플릿의 상태를 볼 수 있습니다.

플릿 상태와 상관없이 플릿의 메타데이터 및 구성을 언제든 업데이트할 수 있습니다. 자세한 내용은 Amazon GameLift 플릿 관리 섹션을 참조하세요. 플릿이 활성 상태가 된 후에 플릿 용량을 업데이트할 수 있습니다. 자세한 내용은 Amazon GameLift 호스팅 용량 확장 섹션을 참조하세요. 원격 위치를 추가하거나 제거할 수도 있습니다.

AWS CLI

AWS CLI로 플릿을 생성하려면 명령줄 창을 열고 `create-fleet` 명령을 사용합니다. `create-fleet` 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 `create-fleet` 섹션을 참조하세요.

아래 표시된 예제 `create-fleet` 요청은 다음과 같은 특징을 가진 새로운 플릿을 만듭니다.

- 플릿은 선택한 게임 빌드에 적절한 운영 체제와 함께 c5.large 온디맨드 인스턴스를 사용합니다.
- 지정된 게임 서버 빌드를 배포하며, 다음 위치에서 준비 상태여야 합니다.
  - us-west-2(홈 리전)
  - sa-east-1(원격 위치)
- TLS 인증서 생성이 활성화됩니다.
- 플릿의 각 인스턴스는 10개의 동일한 게임 서버 프로세스를 동시에 실행하므로 각 인스턴스가 최대 10개의 게임 세션을 동시에 호스팅할 수 있습니다.
- 각 인스턴스에서 Amazon GameLift는 두 개의 새 게임 세션만 동시에 활성화할 수 있습니다. 또한 300초 이내에 플레이어를 호스팅할 준비가 되지 않은 경우 모든 활성 게임 세션을 종료합니다.
- 이 플릿의 인스턴스에서 호스팅되는 모든 게임 세션에는 게임 세션 보호가 설정되어 있습니다.
- 각 플레이어는 15분 내에 3개의 새 게임 세션을 만들 수 있습니다.
- 이 플릿에서 호스팅되는 각 게임 세션에는 지정된 IP 주소 및 포트 범위에 속하는 연결 포인트가 있습니다.
- Amazon GameLift는 이 플릿에 대한 지표를 `EMEAfleets` 지표 그룹에 추가합니다. 이 예제에서 이 지표 그룹은 EMEA 리전의 모든 플릿에 대한 지표를 결합합니다.

```
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"
```

플릿 생성 요청에 성공하면 Amazon GameLift는 사용자가 요청한 구성 설정과 새 플릿 ID를 포함하는 플릿 속성 세트를 반환합니다. 그러면 Amazon GameLift가 플릿 활성화 프로세스를 시작하고 플릿 상태와 위치 상태를 신규로 설정합니다. 이러한 CLI 명령을 사용하여 플릿의 상태를 추적하고 다른 플릿 정보를 확인할 수 있습니다.

- 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

이러한 명령을 사용하여 필요에 따라 플릿의 용량 및 기타 구성 설정을 변경할 수 있습니다.

- update-fleet-attributes
- update-fleet-capacity
- update-fleet-port-settings
- update-runtime-configuration
- create-fleet-locations
- delete-fleet-locations

아마존 GameLift Anywhere 플릿 생성

GameLift Amazon을 사용하여 사용자 환경의 하드웨어를 Amazon GameLift 게임 호스팅에 통합하십시오. Amazon은 하나의 Anywhere 플릿으로 하드웨어를 GameLift Amazon에 GameLift Anywhere 등록합니다. 매치메이커와 게임 세션 대기열에서 Anywhere 및 관리형 EC2 플릿을 통합하여 매치메이킹과 게임 배치를 관리할 수 있습니다.

GameLift AnywhereAmazon에서 게임 서버를 테스트하는 방법에 대한 자세한 내용은 을 참조하십시오Amazon GameLift Anywhere 플릿을 사용하여 통합 테스트.

시작하려면 아마존을 통한 개발 지원 GameLift 버전 5 이상에서 Amazon GameLift Anywhere 플릿 사용에 대한 다음 개념을 검토하십시오.

사용자 지정 위치

Amazon GameLift Anywhere 플릿은 사용자 지정 위치를 사용하여 인프라의 물리적 위치를 나타냅니다.

디바이스 등록

Amazon GameLift Anywhere 플릿이 컴퓨팅 리소스와 통신하려면 먼저 디바이스를 등록하십시오. 작업을 사용하여 Amazon GameLift AWS SDK에서 디바이스 등록을 완료할 수 있습니다 `RegisterCompute`. 이 작업은 디바이스의 IP 주소를 사용하여 디바이스를 플릿 위치와 연결하고 Amazon과 GameLift 통신합니다.

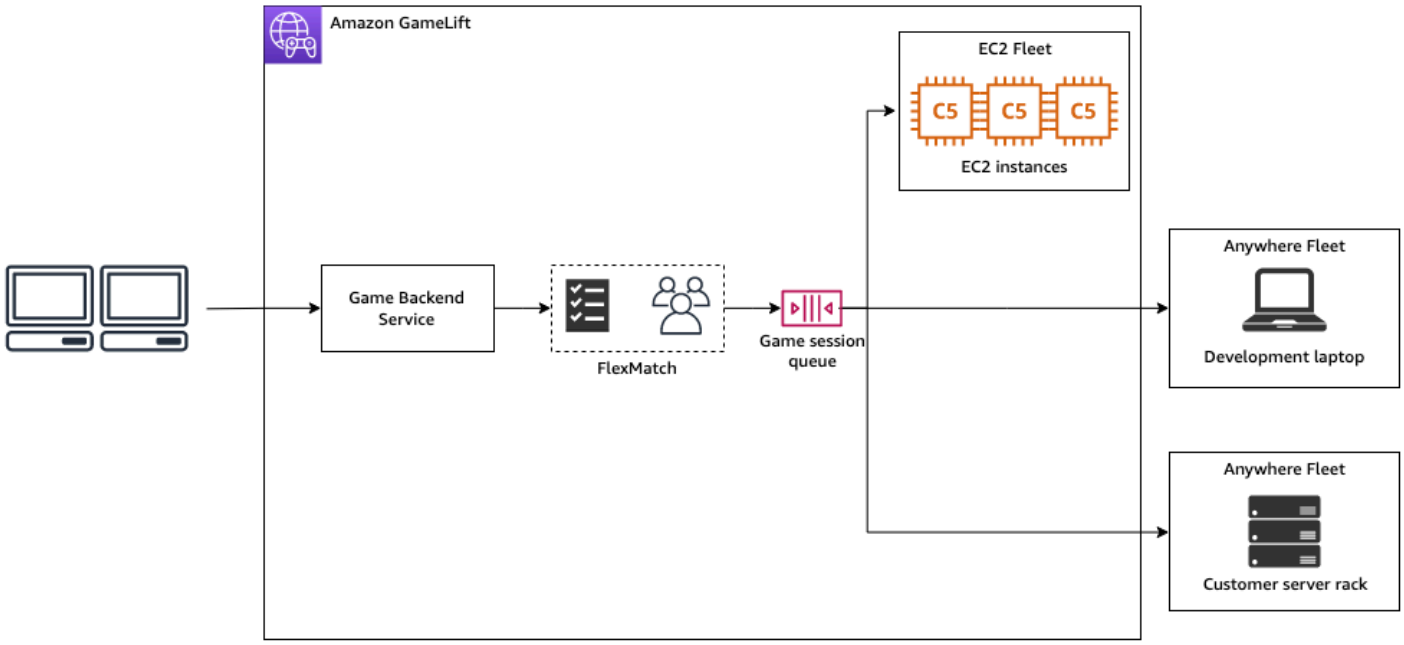
인증 토큰

컴퓨팅에서 게임 서버를 초기화할 때 Amazon GameLift Server SDK는 인증 토큰을 사용하여 Amazon에 게임 서버를 인증합니다. GameLift 인증 토큰 만료 시간까지 동일한 컴퓨팅의 모든 게임 서버에 동일한 인증 토큰을 재사용할 수 있습니다. 인증 토큰을 검색하려면 () 명령을 호출합니다. `get-compute-auth-token` AWS Command Line Interface AWS CLI필요에 따라 토큰을 각 게임 서버에 전달합니다.

게임 세션

컴퓨팅의 각 게임 세션은 컴퓨팅을 플릿 위치에 등록할 때 생성된 것과 동일한 인증 토큰을 사용합니다.

다음 다이어그램은 FlexMatch 매치메이킹과 여러 플릿을 사용하는 게임 세션 대기열을 보여줍니다. 플릿에는 C5 인스턴스가 있는 EC2 플릿, 개발용 랩톱이 있는 Anywhere 플릿, 고객 호스팅 서버 랙이 있는 Anywhere 플릿을 포함합니다.



주제

- 사용자 지정 위치 생성
- 플릿 만들기
- 컴퓨팅 등록
- 서버 프로세스 실행
- 게임 세션 생성
- 관리형 EC2로 마이그레이션

사용자 지정 위치 생성

컴퓨팅 리소스에서 게임 호스팅을 시작하려면 컴퓨터가 있는 위치를 설명하는 사용자 지정 위치를 만듭니다.

Console

사용자 지정 위치를 생성하려면

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 위치를 선택합니다.
3. 위치 페이지에서 위치 생성을 선택합니다.
4. 위치 생성 대화 상자에서 다음 작업을 수행하세요.

a. 위치 이름을 입력합니다. 여기에는 Amazon이 게임을 여러 개 실행하기 위해 GameLift 사용하는 하드웨어의 위치가 레이블로 Anywhere 표시됩니다. Amazon은 사용자 지정 위치의 이름을 custom-로 GameLift 추가합니다.

b. (선택) 태그를 키-값 페어로 사용자 지정 위치에 추가합니다. 추가하려는 각 추가 태그에 대해 새 태그 추가를 선택합니다.

c. 생성을 선택합니다.

AWS CLI

`create-location` 명령을 사용하여 사용자 지정 위치를 생성합니다. `location-name`레이블에는 Amazon이 게임을 여러 개 실행하기 위해 GameLift 사용하는 하드웨어의 위치가 Anywhere 표시됩니다. 사용자 지정 위치를 생성할 때는 위치 이름은 `custom-`으로 시작해야 합니다.

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

출력

```
{
    "Location": {
        "LocationName": "custom-location-1",
        "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-
location-1"
    }
}
```

플릿 만들기

Amazon GameLift 콘솔 또는 를 사용하여 AWS CLI Anywhere 플릿을 생성합니다.

새 Anywhere 플릿을 생성한 후 플릿의 상태가 `NEW`에서 `ACTIVE`로 이동합니다. 플릿이 `ACTIVE` 상태가 되면 플릿은 게임 세션을 호스팅할 준비가 된 것입니다. 플릿 생성 문제에 도움을 받으려면 Amazon GameLift 플릿 문제 디버깅 섹션을 참조하세요.

Console

Anywhere 플릿을 생성하려면

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿 생성을 선택합니다.
4. 컴퓨팅 유형 단계에서 Anywhere를 선택하고 다음을 선택합니다.
5. 플릿 세부 정보 단계에서 세부 정보를 정의한 후 다음을 선택합니다.
6. 사용자 지정 위치 단계에서 생성한 사용자 지정 위치를 선택한 후 다음을 선택합니다. Amazon은 GameLift 자동으로 집을 AWS 리전 플릿을 생성할 지역으로 선택합니다. 홈 리전을 사용하여 리소스에 액세스하고 사용할 수 있습니다.
7. 나머지 플릿 생성 단계를 완료한 다음 제출을 선택하여 Anywhere 플릿을 생성합니다.

AWS CLI

`create-fleet` 명령을 사용하여 Anywhere 플릿을 생성합니다. `locations`에 사용자 지정 위치를 포함합니다. Amazon은 거주 지역 및 사용자가 제공한 사용자 지정 위치에 플릿을 GameLift 생성합니다. 다음 예제에서는 *`FleetName`* 및 *`custom-location-1`*을 고유한 정보로 바꿉니다. `custom-location-1` 변수는 사용자 지정 위치 생성 단계에서 생성한 위치의 이름입니다.

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

출력 예시

```
{
    "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"
    }
}
```

컴퓨팅 등록

생성한 플릿에 컴퓨팅 리소스를 등록하려면 register-compute 명령을 사용합니다. *fleet-id*를 이전 단계에서 fleet-id 반환된 값이나 콘솔의 플릿 세부 정보 페이지에 있는 플릿 ARN으로 교체합니다. *compute-name*을 컴퓨팅 리소스의 IP 주소 *ip-address*로 바꿉니다.

> **Note**
>
> register-compute및 get-compute-auth-token 명령 모두 게임 서버와 분리된 스크립트 또는 프로세스 관리자에서 호출하는 것이 좋습니다.

```
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
```

출력 예시

```
{
    "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"
    }
}
```

서버 프로세스 실행

1. 생성한 플릿에서 컴퓨팅 리소스의 인증 토큰을 가져옵니다.

   게임 서버는 인증 토큰을 사용하여 GameLift Amazon에서 인증합니다. 각 인증 토큰에는 만료 시간이 있습니다. 계속해서 컴퓨팅 리소스를 사용하여 게임 서버를 호스팅하려면 만료 전에 새 인증 토큰을 검색합니다.

   > **Note**
   >
   > GameLift Amazon은 `register-compute` 및 `get-compute-auth-token` 명령을 모두 게임 서버와 분리된 스크립트 또는 프로세스 관리자에서 호출할 것을 권장합니다.

   다음 예제에서는 `fleet-id`를 이전 단계에서 생성한 플릿의 ARN 또는 플릿 ID로 교체합니다. `compute-name`을 이전 단계에서 `register-compute` 명령을 사용하여 생성한 컴퓨팅 이름으로 바꿉니다.

   ```
   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
   ```

   출력 예제:

   ```
   {
       "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. 게임 서버 실행 파일의 인스턴스를 실행합니다.

   게임 서버를 실행하려면 `InitSDK()`를 호출하고 이를 서버 파라미터에 전달하여 게임 서버를 초기화합니다. 자세한 정보는 ServerParameters을 참조하세요.

   Server SDK 입력:

```
//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. 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 게임 서버에서 Amazon으로 전화를 `ProcessReady()` 겁니다 GameLift. 프로세스 파라미터에 대한 자세한 내용은 ProcessParameters 섹션을 참조하세요.

```
// 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);
```

게임 세션 생성

1. 서버 프로세스가 onStartGameSession( ) 메시지에 ActivateGameSession( )으로 응답하도록 게임 서버에 로직을 추가합니다. 이 작업에는 매개변수가 없지만 서버가 게임 세션 생성 메시지를 수신하고 GameLift 수락했음을 Amazon에 확인합니다.

```
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. 게임 클라이언트 백엔드 서비스에서 start-matchmaking, start-game-session-placement, 또는 create-game-session 명령을 사용하여 게임 세션을 시작합니다.

```
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
```

출력 예제:

```
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,
  ...
}
```

Amazon은 등록된 서버 프로세스로 onStartGameSession( ) 메시지를 GameLift 보냅니다. 메시지에는 게임 속성, 게임 세션 데이터, 매치메이커 데이터 및 게임 세션에 대한 추가 정보와 함께 이전 단계의 GameSession 객체가 포함됩니다.

3. 게임 세션이 완료되면 게임 서버 프로세스를 종료합니다.

   Server SDK 입력:

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
   Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

4. `ProcessReady(processParams)`를 호출하여 다른 게임 서버 프로세스를 시작합니다.

관리형 EC2로 마이그레이션

게임 서버를 개발하고 제작 준비를 마치면 Amazon에서 하드웨어를 GameLift 관리하도록 할 수 있습니다. 관리형 EC2 플릿으로 마이그레이션하려면 Amazon에 빌드를 GameLift 업로드하고 관리형 EC2 플릿을 생성하십시오. 빌드 업로드 및 플릿 설정에 대한 자세한 내용은 Amazon GameLift에 사용자 지정 서버 빌드 업로드 및 Amazon GameLift 관리형 플릿 생성 섹션을 참조하세요.

Amazon GameLift 플릿 관리

Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 플릿 설정을 업데이트하거나, 원격 위치를 변경 또는 플릿을 삭제할 수 있습니다.

플릿 구성 업데이트

Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 변경 가능한 플릿 속성, 포트 설정 및 런타임 구성을 업데이트할 수 있습니다. 크기 조정 제한을 변경하려면 Amazon GameLift로 플릿 용량 Auto Scaling 섹션을 참조하세요.

Amazon GameLift console

1. Amazon GameLift 콘솔의 탐색 창에서 플릿을 선택합니다.
2. 업데이트할 플릿을 선택합니다. `ACTIVE` 상태에 있는 플릿만 편집할 수 있습니다.
3. 플릿 세부 정보 페이지의 다음 섹션 중 하나에서 편집을 선택합니다.
   - 플릿 설정
     - 이름과 설명처럼 플릿 속성을 변경합니다.
     - Amazon CloudWatch가 여러 플릿에 대해 집계된 Amazon GameLift 지표를 추적하는 데 사용하는 지표 그룹을 추가하거나 제거합니다.

    - 리소스 생성 제한 설정을 업데이트합니다.
    - 게임 세션 보호를 켜고 끕니다.
  - 런타임 구성 - 런타임 구성의 다음 설정을 변경하고 런타임 구성을 추가하거나 제거할 수 있습니다.
    - 게임 서버의 시작 경로를 변경합니다.
    - 선택 사항인 시작 파라미터를 추가, 제거 또는 변경합니다.
    - 게임 서버가 실행하는 동시 프로세스 수를 변경합니다.
  - 게임 세션 활성화 - 최대 동시 게임 세션 활성화 및 새 활성화 제한 시간을 업데이트하여 서버 프로세스의 실행 및 게임 세션 호스팅 방식을 변경합니다.
  - EC2 포트 설정 - 플릿에 대한 인바운드 액세스를 허용하는 IP 주소 및 포트 범위를 업데이트합니다.
4. 그런 다음 확인을 선택해 변경 사항을 저장합니다.

AWS CLI

다음 AWS CLI 명령을 사용하여 플릿을 업데이트합니다.

- update-fleet-attributes
- update-fleet-port-settings
- update-runtime-configuration

플릿 위치 업데이트

Amazon GameLift 콘솔 또는 AWS CLI를 사용하여 플릿의 원격 위치를 추가하거나 제거할 수 있습니다. 플릿의 홈 리전은 변경할 수 없습니다.

Amazon GameLift console

1. Amazon GameLift 콘솔의 탐색 창에서 플릿을 선택합니다.
2. 업데이트할 플릿을 선택합니다. ACTIVE 상태에 있는 플릿만 편집할 수 있습니다.
3. 플릿 세부 정보 페이지에서 위치 탭을 선택하여 플릿의 위치를 확인합니다.
4. 새 원격 위치를 추가하려면 추가를 선택하고 인스턴스를 배포할 위치를 선택합니다. 플릿의 인스턴스 유형을 사용할 수 없는 인스턴스는 이 목록에 포함되지 않습니다.

5. 새 위치를 선택한 상태에서 추가를 선택합니다. Amazon GameLift는 상태가 NEW로 설정된 새 위치를 목록에 추가합니다. 그러면 Amazon GameLift는 추가된 각 위치에서 인스턴스를 프로비저닝하고 게임 세션을 호스팅할 준비를 시작합니다.
6. 플릿에서 기존 원격 위치를 제거하려면 확인란을 사용하여 나열된 위치를 하나 이상 선택합니다.
7. 플릿을 하나 이상 선택한 상태에서 제거를 선택합니다. 제거된 위치는 상태가 DELETING으로 설정된 상태로 목록에 남아 있습니다. 그러면 Amazon GameLift는 제거된 위치에서 활동을 종료하는 프로세스를 시작합니다. 게임 세션을 호스팅하는 활성 인스턴스가 있는 경우 Amazon GameLift는 게임 서버 종료 프로세스를 사용하여 게임 세션을 정상적으로 종료하고, 게임 서버와 인스턴스를 종료합니다.

AWS CLI

다음 AWS CLI 명령을 사용하여 다음 플릿 위치를 업데이트합니다.

- create-fleet-locations
- delete-fleet-locations

플릿 삭제

플릿이 더 이상 필요 없는 경우 삭제할 수 있습니다. 플릿을 삭제하면 관련된 게임 세션과 플레이어 세션과 관련된 모든 데이터 및 수집 지표 데이터가 영구적으로 제거됩니다. 대신에, 플릿 보관, Auto Scaling 비활성화 및 수동으로 인스턴스 0로 플릿을 조정합니다.

> **Note**
> 플릿에 VPC 피어링 연결이 있는 경우에는 먼저 CreateVpcPeeringAuthorization을 호출하여 권한 부여를 요청합니다. Amazon GameLift는 플릿을 삭제하는 동안 VPC 피어링 연결을 삭제합니다.

Amazon GameLift 콘솔 또는 AWS CLI 도구를 사용하여 플릿을 삭제할 수 있습니다.

Amazon GameLift console

1. Amazon GameLift 콘솔의 탐색 창에서 플릿을 선택합니다.

2. 삭제하려는 플릿을 선택합니다. ACTIVE 또는 ERROR 상태인 플릿만 삭제할 수 있습니다.
3. 삭제를 선택합니다.
4. 플릿 삭제 대화 상자에 **delete**을 입력하여 삭제를 확인합니다.
5. 삭제를 선택합니다.

AWS CLI

다음 AWS CLI 명령을 사용하여 다음 플릿을 업데이트합니다.

- delete-fleet

Amazon GameLift 플릿에 별칭 추가

Amazon GameLift 별칭은 플릿 지정을 추상화하는 데 사용됩니다. 플릿 지정은 플레이어를 위한 새 게임 세션을 생성할 때 사용 가능한 리소스를 검색할 GameLift 위치를 Amazon에 알려줍니다. 특정 플릿 ID 대신 별칭을 사용하면 별칭의 대상 위치를 변경하여 플레이어 트래픽을 한 대상에서 다른 대상으로 원활하게 전환합니다.

두 종류의 별칭 라우팅 전략이 있습니다.

- 단순 - 플레이어 트래픽을 지정된 플릿 ID로 라우팅합니다. 언제든 별칭에 대한 플릿 ID를 업데이트할 수 있습니다.
- 터미널 - 메시지를 클라이언트로 다시 전달합니다. 예를 들어 out-of-date 클라이언트를 사용하는 플레이어를 업그레이드를 받을 수 있는 위치로 안내할 수 있습니다.

플릿의 수명은 유한하며, 게임 수명 동안 플릿을 전환할 몇 가지 이유가 있습니다. 플릿의 게임 서버 빌드를 업데이트하거나 기존 플릿에서 특정 컴퓨팅 리소스 속성을 변경할 수 없습니다. 대신 변경 사항을 적용한 새로운 플릿을 생성한 다음 플레이어를 새로운 플릿으로 전환합니다. 별칭이 있는 경우 플릿 전환은 게임에 최소한의 영향을 미치며 플레이어에게는 보이지 않습니다.

별칭은 대기열을 사용하지 않는 게임에서 유용합니다. 대기열에서 플릿을 전환하는 것은 새로운 플릿을 만들고, 대기열에 추가하고, 플레이어에게 보이지 않는 오래된 플릿을 제거하는 간단한 문제입니다. 반대로 대기열을 사용하지 않는 게임 클라이언트는 Amazon GameLift 서비스와 통신할 때 사용할 플릿을 지정해야 합니다. 별칭이 없는 경우 플릿 전환 시 게임 코드를 업데이트하고 업데이트된 게임 클라이언트를 플레이어에게 배포해야 합니다.

별칭이 가리키는 fleet-id를 업데이트하면 최대 2분의 전환 기간이 있으며, 이 경우 별칭에 대한 게임 세션이 이전 플릿으로 넘어갈 수 있습니다.

새 별칭 생성

별칭은 여기에 설명된 대로 Amazon GameLift 콘솔을 사용하거나 AWS create-alias라는 CLI 명령을 사용하여 생성할 수 있습니다.

1. Amazon GameLift 콘솔의 탐색 창에서 Aliases를 선택합니다.
2. 별칭 페이지에서 별칭 생성을 선택합니다. 별칭 이름에 플릿 유형을 포함하는 것이 좋습니다. 이렇게 하면 별칭 목록을 볼 때 플릿 유형을 훨씬 쉽게 식별할 수 있습니다.
3. 별칭 생성 페이지의 별칭 세부 정보에서 다음을 수행합니다.
   a. 이름에는 별칭 이름을 입력합니다.
   b. 설명에는 식별하기 위한 간단한 설명을 입력합니다.
   c. 단순 또는 터미널 라우팅 유형을 선택합니다.
4. (선택) 태그에서 키 및 값 쌍을 입력하여 별칭에 태그를 추가합니다.
5. 생성을 선택합니다.

별칭 편집

Amazon GameLift 콘솔을 사용하거나 AWS CLI 명령 update-alias를 사용하여 별칭을 편집할 수 있습니다.

1. Amazon GameLift 콘솔의 탐색 창에서 Aliases를 선택합니다.
2. 별칭 페이지에서 편집할 별칭을 선택합니다.
3. 별칭 페이지에서 편집을 선택합니다.
4. 별칭 편집 페이지에서 다음을 편집할 수 있습니다.
   - 별칭 이름 - 별칭의 표시 이름.
   - 설명 - 별칭에 대한 간략한 설명.
   - 유형 - 플레이어 트래픽 라우팅 전략. 단순을 선택하여 연결된 플릿을 변경하거나 터미널을 선택하여 종료 메시지를 편집합니다.
5. 변경 사항 저장을 선택합니다.

Amazon GameLift 플릿 문제 디버깅

이 주제에서는 Amazon GameLift 관리형 호스팅 솔루션의 플릿 구성 문제에 대한 지침을 제공합니다. 추가적인 문제 해결을 위해 플릿이 활성화되면 플릿 인스턴스에 원격으로 액세스할 수 있습니다. Amazon GameLift 플릿 인스턴스에 원격으로 연결 섹션을 참조하세요.

플릿 생성 문제

플릿이 생성되면 Amazon GameLift 서비스는 각 플릿 위치에 새 인스턴스를 배포하고 게임 서버를 실행할 수 있도록 준비하는 워크플로를 시작합니다. 자세한 설명은 Amazon GameLift 플릿 생성 작동 방식 섹션을 참조하세요. 플릿은 활성 상태에 도달할 때까지 게임 세션과 플레이어를 호스팅할 수 없습니다. 이 섹션에서는 플릿이 활성화되지 못하게 하는 가장 일반적인 문제에 대해 설명합니다.

다운로드 및 검증

이 단계에서는 추출된 빌드 파일에 문제가 있거나 설치 스크립트가 실행되지 않거나 혹은 런타임 구성에 지정된 실행 파일이 빌드 파일에 포함되지 않아 플릿 생성이 실패할 수 있습니다. Amazon GameLift는 이러한 각각의 문제와 관련된 로그를 제공합니다.

만약 로그가 이 문제를 발견하지 못한다면 문제는 내부 서비스 오류일 가능성이 있습니다. 이 경우, 플릿을 다시 생성해보십시오. 문제가 지속된다면 (파일이 손상되었을 수도 있으니) 게임 빌드의 업로드를 다시 고려하십시오. Amazon GameLift 지원에 문의하거나 포럼에 질문을 게시해도 됩니다.

빌드

빌드 단계 동안의 실패 문제는 대부분 게임 빌드 파일 및/혹은 설치 스크립트 문제입니다. Amazon GameLift에 업로드할 때 게임 빌드 파일을 적절한 운영 체제가 실행 중인 시스템에 설치할 수 있는지 확인하십시오. 기존 개발 환경이 아닌 건전한 OS 설치를 사용했는지 확인하십시오.

활성화하는 중

대부분의 일반 플릿 생성 문제는 활성화 중 단계 동안 발생합니다. 이 단계에서는 게임 서버 실행 가능성, 런타임 구성 설정, Server SDK를 사용하여 Amazon GameLift 서비스와 상호 작용하는 게임 서버 기능을 비롯해 여러 요소에 대한 테스트가 진행됩니다. 플릿 활성화 단계에서 발생하는 공통적인 문제는 다음과 같습니다.

서버 프로세스를 시작하지 못합니다.

먼저, 플릿의 런타임 구성에서 실행 경로와 선택적 시작 파라미터를 올바르게 설정했는지 확인합니다. 플릿 세부 정보 페이지의 세부 정보 섹션을 사용하거나 AWS CLI 명령 describe-runtime-

configuration을 호출하여 플릿의 현재 런타임 구성을 확인할 수 있습니다. 런타임 구성이 올바르게 보인다면 게임 빌드 파일 및/또는 설치 스크립트 문제를 확인합니다.

서버 프로세스가 시작되지만 플릿을 활성화할 수 없습니다.

서버 프로세스가 시작되고 계속 성공적으로 실행되지만 플릿이 활성 상태로 변경되지 않는 경우, 서버 프로세스가 Amazon GameLift에 게임 세션을 호스팅할 준비가 되었음을 알리는 데 실패했기 때문일 가능성이 높습니다. 게임 서버가 Server API 작업 `ProcessReady()`를 올바르게 호출하고 있는지 확인합니다(서버 프로세스 초기화 참조).

VPC 피어링 연결 요청 실패

VPC 피어링 연결로 생성되는 플릿은(새 플릿으로 VPC 피어링을 설정하려면 참조) 이 활성화 단계에서 VPC 피어링이 실행됩니다. 어떤 이유로든 VPC 피어링에 실패할 경우 새 플릿이 활성 상태로 바뀌지 못합니다. describe-vpc-peering-connections를 호출하여 피어링 요청이 성공했는지 여부를 추적할 수 있습니다. 유효한 VPC 피어링 권한 부여가 존재하는지 확인해야 합니다(describe-vpc-peering-authorizations). 권한 부여는 24시간 동안만 유효하기 때문입니다.

서버 프로세스 문제

서버 프로세스가 시작되었지만 곧 실패했거나 상태 불량을 보고합니다.

게임 빌드와 관련된 문제 외에, 인스턴스에서 동시에 너무 많은 서버 프로세스를 실행하려고 할 경우에도 이런 결과가 발생할 수 있습니다. 동시에 실행 가능한 최적 프로세스 수는 인스턴스 유형과 게임 서버의 리소스 요구 사항에 따라 다릅니다. 성능이 개선되는지 보려면 플릿의 런타임 구성에 설정되어 있는 동시 프로세스 수를 줄여보십시오. Amazon GameLift 콘솔(플릿의 용량 할당 설정 편집)을 사용하거나 AWS CLI 명령 update-runtime-configuration을 호출하여 플릿의 런타임 구성을 변경할 수 있습니다.

플릿 삭제 문제

최대 인스턴스 카운트 때문에 플릿을 종료할 수 없습니다.

이 오류 메시지는 삭제되어 허용되면 안 되는 플릿의 인스턴스가 여전히 활성 상태임을 나타냅니다. 먼저 플릿의 활성 인스턴스를 0으로 축소해야 합니다. 플릿의 인스턴스 카운트를 “0”으로 직접 설정한 후 스케일 다운이 적용될 때까지 기다립니다. 수동 설정에 방해가 되므로 자동 조정(auto-scaling) 기능을 해제해야 합니다.

VPC 작업이 허용되지 않습니다.

이 문제는 특히 VPC 피어링 연결을 생성한 플릿에만 적용됩니다(Amazon GameLift용 VPC 피어링 참조). 이 시나리오는 플릿 삭제 프로세스에 플릿의 VPC 및 모든 VPC 피어링 연결 삭제도 포함되기 때문에 발생합니다. 먼저 Amazon GameLift Service API CreateVpcPeeringAuthorization()를 호출하거나 AWS CLI 명령 `create-vpc-peering-authorization`을 사용하여 승인을 받아야 합니다. 권한을 받은 후에 플릿을 삭제할 수 있습니다.

Realtime 서버 플릿 문제

좀비 게임 세션: 게임을 시작하고 실행하지만 절대로 끝나지 않습니다.

이 문제가 다음 시나리오 중 하나처럼 관찰될 수 있습니다.

- 플릿의 Realtime 서버에서 스크립트 업데이트를 픽업하지 않습니다.
- 플릿이 최대 용량에 빠르게 도달하고, 플레이어 활동(예: 새로운 게임 세션 요청)이 감소해도 축소되지 않습니다.

대개 Realtime 스크립트에서 `processEnding` 호출에 실패한 것이 원인입니다. 플릿이 활성화되고 게임 세션이 시작되더라도 이를 중지할 방법이 없습니다. 따라서 게임 세션을 실행 중인 Realtime 서버가 비워지지 않아 새로운 세션을 시작하지 못하고 새로운 Realtime 서버가 확보될 때만 새로운 게임 세션이 시작될 수 있습니다. 그리고 Realtime 스크립트 업데이트는 이미 실행 중인 게임 세션에 영향을 주지 않습니다.

이를 방지하기 위해서는 `processEnding` 호출을 시작하는 메커니즘이 스크립트에 있어야 합니다. Realtime 서버 스크립트 예제에서 설명한 것처럼 한 가지 방법은 일정 시간 동안 플레이어가 연결되지 않을 경우 스크립트에서 현재 게임 세션을 종료하도록 유휴 세션 제한 시간을 프로그래밍하는 것입니다.

그러나 이 시나리오가 발생할 경우 Realtime 서버의 중단을 막는 두 가지 차선책이 있습니다. 그 방법은 Realtime 서버 프로세스 또는 기본 플릿 인스턴스를 다시 시작하게 하는 것입니다. 이 경우 GameLift가 게임 세션을 자동으로 종료합니다. Realtime 서버가 비워지면 실시간 스크립트 최신 버전을 사용하여 새로운 게임 세션을 시작할 수 있습니다.

이 문제의 정도에 따라 두 가지 방법으로 해결할 수 있습니다.

- 전체 플릿을 축소합니다. 이 방법은 간단하면서 광범위한 영향을 줍니다. 플릿을 제로 인스턴스로 숙소하고, 플릿이 완전히 축소될 때까지 기다렸다가 다시 확장합니다. 그러면 기존 게임 세션이 모두 지워져 가장 최근에 업데이트된 Realtime 스크립트로 새로 시작할 수 있습니다.

- 인스턴스에 원격으로 액세스하여 프로세스를 다시 시작합니다. 이 방법은 해결할 프로세스가 적을 경우에 좋습니다. 이미 인스턴스에 로그인한 경우에는(예: 테일 로그 또는 디버깅을 위해) 이 방법이 가장 빠른 방법일 수 있습니다. Amazon GameLift 플릿 인스턴스에 원격으로 연결 섹션을 참조하세요.

Realtime 스크립트에서 `processEnding`을 호출하는 방법을 포함시키지 않을 경우 플릿이 활성화되어 게임 세션이 시작되어도 두 가지 어려운 상황이 발생할 수 있습니다. 첫째, 실행 중인 게임 세션이 끝나지 않습니다. 따라서 이 게임 세션을 실행 중인 서버 스포세스가 결코 비워지지 않아 새 게임 세션을 시작하지 못합니다. 둘째, Realtime 서버가 스크립트 업데이트를 픽업하지 않습니다.

Amazon GameLift 플릿 인스턴스에 원격으로 연결

활성 Amazon GameLift 관리형 EC2 플릿의 모든 인스턴스에 연결할 수 있습니다. 인스턴스에 액세스하는 일반적인 이유는 다음과 같습니다.

- 게임 서버 통합 관련 문제 해결
- 런타임 구성 및 기타 플릿별 설정을 미세 조정합니다.
- 로그 추적과 같은 실시간 게임 서버 활동을 확인할 수 있습니다.
- 실제 플레이어 트래픽을 사용하여 벤치마킹 도구를 실행하세요.
- 게임 세션 또는 서버 프로세스와 관련된 특정 문제를 조사하세요.

인스턴스에 연결할 때는 다음과 같은 잠재적 문제를 고려하세요.

- 활성 플릿의 인스턴스에 연결할 수 있습니다. 비활성 플릿, 즉 활성화 중이거나 오류 상태에 있는 비활성 플릿은 짧은 기간 동안 액세스할 수 있을 수 있습니다. 플릿 활성화 문제에 대한 도움이 필요하면 Amazon GameLift 플릿 문제 디버깅 섹션을 참조하세요.
- 활성 인스턴스에 연결해도 인스턴스의 호스팅 활동에는 영향을 주지 않습니다. 인스턴스는 런타임 구성에 따라 서버 프로세스를 계속 시작하고 중지합니다. 게임 세션을 활성화하고 호스팅합니다. 축소 이벤트 또는 기타 이벤트에 대한 응답으로 종료될 수 있습니다.
- 인스턴스의 파일 또는 설정을 변경하면 인스턴스의 활성 게임 세션과 연결된 플레이어에 영향을 미칠 수 있습니다.

다음 지침은 AWS 명령줄 인터페이스 (CLI) 를 사용하여 인스턴스에 원격으로 연결하는 방법을 설명합니다. Amazon GameLift 서비스 API 참조에 설명된 대로 AWS SDK를 사용하여 프로그래밍 방식으로 호출할 수도 있습니다.

인스턴스 데이터 수집

다음 정보를 수집하세요.

- 연결하려는 인스턴스의 ID. 인스턴스 ID 또는 ARN을 사용할 수 있습니다.
- 인스턴스에서 사용 중인 Amazon GameLift 서버 SDK 버전. 서버 SDK는 인스턴스에서 실행되는 게임 빌드와 통합됩니다.

인스턴스 데이터를 검색하려면

다음 단계에서는 연결하려는 인스턴스의 관리형 EC2 플릿 ID가 있다고 가정합니다.

1. 컴퓨팅 이름을 가져오세요.

   관리형 EC2 플릿의 list-compute를 호출하여 플릿 내 모든 활성 컴퓨팅 목록을 가져옵니다. 단일 위치 플릿의 경우 플릿 ID 또는 ARN을 지정합니다. 여러 위치에 있는 플릿의 경우 플릿 ID 또는 ARN과 위치를 지정합니다. 관리형 EC2 플릿의 경우 컴퓨팅은 EC2 인스턴스이고 반환된 속성은 `ComputeName` 인스턴스 ID입니다. 예:

   요청

   ```
   aws gamelift list-compute \
     --fleet-id  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
     --location ""sa-east-1"
   ```

   응답

   ```
   {
     "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. 서버 SDK 버전을 찾으십시오.

   서버 SDK 버전은 빌드 리소스의 속성입니다.

   a. 플릿 describe-fleet-attributesID로 호출하여 플릿의 빌드 ID 및 ARN을 가져옵니다.

   b. 빌드 ID 또는 ARN을 사용하여 describe-build를 호출하여 빌드의 서버 SDK 버전을 가져옵니다.

      예:

      요청

```
aws gamelift describe-fleet-attributes /
  --fleet-ids  "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

      응답

```
{
  "FleetAttributes": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeType": "EC2",
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",
      . . .
    }
  ]
}
```

      요청

```
aws gamelift describe-build /
  --build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```

      응답

```
"Build": {
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "Name": "My_Game_Server_Build_One",
  "OperatingSystem": "AMAZON_LINUX_2",
  "ServerSdkVersion": "5.1.1",
  . . .
}
```

인스턴스에 연결 (서버 SDK 5)

연결하려는 인스턴스가 서버 SDK 버전 5.x의 게임 빌드를 실행하는 경우 다음 지침에 따라 Amazon EC2 Systems Manager (SSM) 를 사용하여 인스턴스에 연결합니다. Windows나 Linux를 실행 중인 원격 인스턴스에 액세스할 수 있습니다.

1. 인스턴스에 대한 액세스 자격 증명을 요청합니다. 연결하려는 인스턴스의 컴퓨팅 이름과 플릿 ID가 있으면 를 호출하십시오. get-compute-access 성공하면 Amazon은 인스턴스 액세스를 위한 임시 자격 증명 세트를 GameLift 반환합니다. 예:

   요청

   ```
   aws gamelift get-compute-access \
   --compute-name i-11111111a222b333c \
   --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa
   --region us-west-2
   ```

   응답

   ```
   {
     "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. 액세스 자격 증명을 내보냅니다. 선택적으로 자격 증명을 환경 변수로 내보내고 이를 사용하여 기본 사용자에 대한 AWS CLI를 구성할 수 있습니다. 자세한 내용은 사용 AWS Command Line Interface설명서의 AWS CLI를 구성하기 위한 환경 변수를 참조하십시오.

```
export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
```

3. 플릿 인스턴스에 연결합니다. 연결하려는 인스턴스로 SSM 세션을 시작합니다. 인스턴스의 AWS 지역 또는 위치를 포함하세요. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 세션 시작 (AWSCLI) 을 참조하십시오. 1단계에서 획득한 자격 증명을 사용하십시오. 예:

```
aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2
```

인스턴스에 연결 (서버 SDK 4.x 이하)

연결하려는 인스턴스가 서버 SDK 버전 4 또는 이전 버전의 게임 빌드를 실행하는 경우 다음 안내를 따르세요. Windows 또는 Linux를 실행하는 인스턴스에 연결할 수 있습니다. 원격 데스크톱 프로토콜 (RDP) 클라이언트를 사용하여 Windows 인스턴스에 연결합니다. SSH 클라이언트를 사용하여 Linux 인스턴스에 연결합니다.

1. 인스턴스에 대한 액세스 자격 증명을 요청합니다. 인스턴스 ID가 있는 경우 명령을 get-instance-access사용하여 액세스 자격 증명을 요청합니다. 성공하면 Amazon은 인스턴스의 운영 체제, IP 주소 및 자격 증명 세트 (사용자 이름 및 비밀 키) 를 GameLift 반환합니다. 자격 증명 형식은 인스턴스 운영 체제에 따라 다릅니다. 다음 지침을 이용해 RDP나 SSH에 대한 자격 증명을 가져옵니다.

   - Windows 인스턴스 - Windows 인스턴스에 연결을 시도하면 RDP가 사용자 이름과 암호를 요구합니다. `get-instance-access` 요청이 단순한 문자열로 이러한 값을 반환하므로 이 반환된 값을 그대로 사용할 수 있습니다. 예제 자격 증명:

```
"Credentials": {
    "Secret": "aA1bBB2cCCd3EEE",
    "UserName": "gl-user-remote"
}
```

- Linux 인스턴스 - Linux 인스턴스에 연결을 시도하면 SSH가 사용자 이름과 프라이빗 키를 요구합니다. Amazon은 RSA 프라이빗 키를 GameLift 발급하여 단일 문자열로 반환합니다. 이때 줄 바꿈 문자 (\n) 는 줄 바꿈을 나타냅니다. 프라이빗 키를 사용할 수 있게 하려면 (1) 문자열을 파일로 변환하고 (2) 새 .pem 파일에 대한 권한을 설정하는 단계를 수행하십시오. 예제 자격 증명이 반환되었습니다.

```
"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"
}
```

AWSCLI를 사용하는 경우 요청에 --query 및 --output 매개변수를 포함하여 .pem 파일을 자동으로 생성할 수 있습니다. `get-instance-access`

.pem 파일에 대한 권한을 설정하려면 다음 명령을 실행합니다.

```
$ chmod 400 MyPrivateKey.pem
```

2. 원격 연결을 위한 포트를 엽니다. 플릿 구성에서 승인된 모든 포트를 통해 Amazon GameLift 플릿의 인스턴스에 액세스할 수 있습니다. describe-fleet-port-settings 명령을 사용하여 플릿의 포트 설정을 볼 수 있습니다.

   모범 사례로, 필요할 때만 원격 액세스용 포트를 열고 완료되면 닫을 것을 권장합니다. 플릿을 생성한 후 활성화되기 전에는 포트 설정을 업데이트할 수 없습니다. 문제가 발생하면 포트 설정을 연 상태로 플릿을 다시 생성하십시오.

   update-fleet-port-settings 명령을 사용하여 원격 연결을 위한 포트 설정을 추가합니다(예: SSH는 22, RDP는 3389). IP 범위 값에는 연결에 사용할 장치의 IP 주소를 지정합니다(CIDR 형식으로 변환). 예제

   ```
   $ 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"
   ```

   다음 예제에서는 Windows 플릿에서 포트 3389를 엽니다.

   ```
   $ 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. 원격 연결 클라이언트를 엽니다. Windows용 원격 데스크톱 또는 Linux 인스턴스용 SSH를 사용합니다. IP 주소, 포트 설정, 액세스 자격 증명을 사용하여 인스턴스에 연결합니다.

   SSH 예:

   ```
   ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
   ```

원격 인스턴스의 파일 보기

인스턴스에 원격으로 연결하려면 완전한 사용자 및 관리자 액세스가 필요합니다. 즉 게임 호스팅에 오류와 장애를 야기할 수도 있다는 뜻입니다. 인스턴스가 활성 플레이어가 있는 게임을 호스팅하는 경우 게임 세션이 중단되고 플레이어가 삭제되거나 게임 종료 프로세스가 중단되고 저장된 게임 데이터 및 로그에 오류가 발생할 위험이 있습니다.

호스팅 인스턴스에서 다음 리소스를 찾아보세요.

- 게임 빌드 파일. 이 파일은 Amazon에 업로드한 게임 GameLift 빌드입니다. 여기에는 하나 이상의 게임 서버 실행 파일, 자산 및 종속성이 포함됩니다. 게임 빌드 파일은 다음과 같은 루트 디렉터리에 있습니다. game
  - Windows: c:\game
  - Linux: /local/game
- 게임 로그 파일. 지정한 디렉터리 경로의 game 루트 디렉터리에서 게임 서버가 생성하는 로그 파일을 찾을 수 있습니다.
- 아마존 GameLift 호스팅 리소스. 루트 Whitewater 디렉터리에는 Amazon GameLift 서비스에서 게임 호스팅 활동을 관리하는 데 사용하는 파일이 들어 있습니다. 어떤 이유로든 이러한 파일을 수정하지 마십시오.
- 런타임 구성. 개별 인스턴스의 런타임 구성에는 액세스하지 마십시오. 런타임 구성 속성을 변경하려면 플릿의 런타임 구성을 업데이트하십시오 (AWSSDK 작업 UpdateRuntimeConfiguration또는 참조 AWS CLI update-runtime-configuration).
- 플릿 데이터. JSON 파일에는 인스턴스가 속한 플릿에 대한 정보가 포함되어 있으며, 이는 인스턴스에서 실행되는 서버 프로세스에서 사용할 수 있습니다. JSON 파일은 다음 위치에 있습니다.
  - Windows: C:\GameMetadata\gamelift-metadata.json
  - Linux: /local/gamemetadata/gamelift-metadata.json
- TLS 인증서. 인스턴스가 TLS 인증서 생성이 활성화된 플릿에 있는 경우 다음 위치에서 인증서, 인증서 체인, 개인 키, 루트 인증서를 비롯한 인증서 파일을 찾으십시오.
  - Windows: c:\\GameMetadata\Certificates
  - Linux: /local/gamemetadata/certificates/

Amazon GameLift 호스팅 용량 확장

인스턴스 단위로 측정한 호스팅 용량은 Amazon GameLift가 동시에 호스팅할 수 있는 게임 세션 수와 해당 게임 세션이 수용할 수 있는 동시 플레이어 수를 나타냅니다. 게임 호스팅에서 가장 어려운 작업 중 하나는 필요하지 않은 리소스에 비용을 낭비하지 않고 플레이어 수요에 맞게 용량을 확장하는 것입니다. 자세한 내용은 플릿 용량 조정 섹션을 참조하세요.

용량은 플릿 위치 수준에서 조정됩니다. 모든 플릿에는 플릿의 홈 AWS 리전이 하나 이상 있습니다. 용량을 확인하거나 조정할 때 정보는 플릿의 홈 리전 및 추가 원격 위치를 포함하여 위치별로 나열됩니다.

유지할 인스턴스 수를 수동으로 설정하거나 플레이어 수요 변화에 따라 용량을 동적으로 조정하도록 Auto Scaling을 설정할 수 있습니다. 대상 기반 Auto Scaling을 켜서 시작하는 것이 좋습니다. 대상 기

반 Auto Scaling의 목표는 현재 플레이어를 수용할 수 있는 충분한 호스팅 리소스를 유지하면서 예상치 못한 플레이어 수요 급증에 대비할 수 있는 추가 리소스를 유지하는 것입니다. 대부분의 게임에서 대상 기반 Auto Scaling은 매우 효과적인 조정 솔루션을 제공합니다.

이 섹션의 항목에서는 다음 작업에 대한 상세한 도움을 제공합니다.

- 용량 조정의 최소 및 최대 제한 조정
- 수동으로 용량 수준 설정
- 대상 기반 Auto Scaling 사용
- 규칙 기반 Auto Scaling 관리(고급 기능)
- Auto Scaling을 일시적으로 비활성화

대부분의 플릿 조정 작업은 Amazon GameLift 콘솔을 사용하여 수행할 수 있습니다. Amazon GameLift Service API와 함께 AWS SDK 또는 AWS Command Line Interface(AWS CLI)를 사용할 수도 있습니다.

콘솔에서 플릿 용량을 관리하려면

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿의 세부 정보 페이지를 열기 위해 활성 플릿의 이름을 클릭합니다.
4. 크기 조정 탭을 선택합니다. 이 탭에서는 다음을 수행할 수 있습니다.
    - 전체 플릿에 대한 과거 조정 지표를 볼 수 있습니다.
    - 크기 조정 제한 및 현재 용량 설정을 포함하여 각 플릿 위치의 용량 설정을 보고 업데이트할 수 있습니다.
    - 대상 기반 Auto Scaling을 업데이트하고, 전체 플릿에 적용된 규칙 기반 Auto Scaling 정책을 확인하며, 각 위치에 대한 Auto Scaling 활동을 일시 중단할 수 있습니다.

주제

- Amazon GameLift의 용량 제한 설정
- Amazon GameLift 플릿의 수동 용량 설정
- Amazon GameLift로 플릿 용량 Auto Scaling

Amazon GameLift의 용량 제한 설정

Amazon GameLift 플릿 위치의 호스팅 용량을 수동 또는 Auto Scaling을 통해 확장할 때는 해당 위치의 크기 조정 제한을 고려합니다. 모든 플릿 위치에는 위치 용량에 대한 허용 범위를 정의하는 최소 및 최대 제한이 있습니다. 플릿 위치의 기본 제한 값은 최소는 0개 인스턴스, 최대는 1개 인스턴스입니다. 플릿 위치를 조정하려면 우선 제한을 조정합니다.

Auto Scaling을 사용하는 경우 최대 제한에 따라 Amazon GameLift는 플레이어 수요에 맞춰 플릿 위치를 확대할 수 있지만, DDOS 공격과 같은 호스팅 비용 폭주를 방지할 수 있습니다. 용량이 최대 제한 값에 가까워졌을 때 이를 알리도록 Amazon CloudWatch 경보를 설정하면 상황을 평가해 필요에 따라 수동으로 조정을 할 수 있습니다. (결제 경보를 생성하여 AWS 비용을 모니터링할 수도 있습니다.) 최소 제한은 플레이어 수요가 적은 경우에도 호스팅 가용성을 유지하는 데 유용합니다.

Amazon GameLift 콘솔에서나 AWS Command Line Interface(AWS CLI)를 사용하여 플릿 위치에 대한 용량 제한을 설정할 수 있습니다.

용량 제한을 설정하려면

Console

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿의 세부 정보 페이지를 열기 위해 활성 플릿의 이름을 클릭합니다.
4. 크기 조정 탭의 크기 조정 용량 아래에서 플릿 위치를 선택한 다음 편집을 선택합니다.
5. 크기 조정 용량 편집 대화 상자에서 최소 크기, 원하는 인스턴스, 최대 크기에 대한 인스턴스 수를 설정합니다.
6. 확인을 선택합니다.

AWS CLI

1. 현재 용량 설정을 확인합니다. 명령줄 창에서 용량 변경을 원하는 플릿 ID와 함께 describe-fleet-location-capacity 명령을 사용합니다. 이 명령은 해당 위치에 대한 현재 용량 설정이 포함된 FleetCapacity 객체를 반환합니다. 새 인스턴스 제한 값이 현재 원하는 인스턴스에 대한 설정을 수용할 수 있는지 확인합니다.

```
aws gamelift describe-fleet-location-capacity \
    --fleet-id <fleet identifier> \
```

```
    --location <location name>
```

2. 제한에 대한 설정을 업데이트합니다. 명령줄 창에 다음 파라미터와 함께 update-fleet-capacity 명령을 사용합니다. 동일한 명령으로 두 인스턴스 제한 값과 원하는 인스턴스 수를 조정할 수 있습니다.

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
--desired-instances <fleet capacity goal>
```

예제:

```
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
```

요청이 성공하면 Amazon GameLift이 플릿 ID를 반환합니다. 새 `max-size` 또는 `min-size` 값이 현재 `desired-instances` 설정과 충돌하는 경우 Amazon GameLift는 오류를 반환합니다.

Amazon GameLift 플릿의 수동 용량 설정

새 플릿을 생성하면 Amazon GameLift는 자동으로 원하는 인스턴스를 각 플릿 위치에서 하나의 인스턴스로 설정합니다. 그런 다음 Amazon GameLift는 각 위치에 새 인스턴스를 하나씩 배포합니다. 플릿 용량을 변경하려면 대상 기반 Auto Scaling 정책을 추가하거나 위치에 원하는 인스턴스 수를 수동으로 설정할 수 있습니다. 자세한 내용은 플릿 용량 조정 섹션을 참조하세요.

Auto Scaling이 필요하지 않거나 용량을 지정된 수준으로 유지해야 하는 경우 플릿의 용량을 수동으로 설정하는 것이 유용할 수 있습니다. 수동 용량 설정은 대상 기반 Auto Scaling 정책을 사용하지 않는 경우에만 작동합니다. 대상 기반 Auto Scaling 정책이 있는 경우, 그 즉시 조정 규칙에 따라 원하는 용량으로 재설정됩니다.

Amazon GameLift 콘솔이나 AWS Command Line Interface(AWS CLI)를 사용하여 용량을 수동으로 설정할 수 있습니다. 플릿은 활성화 상태이어야 합니다.

Auto Scaling 일시 중지

각 플릿 위치에 대한 모든 Auto Scaling 활동을 일시 중단할 수 있습니다. Auto Scaling이 일시 중단되면 수동으로 변경하지 않는 한, 플릿 위치에 원하는 인스턴스 수가 동일하게 유지됩니다. 특정 위치에 대한 Auto Scaling을 일시 중단하면 플릿의 현재 정책과 향후 정의할 수 있는 모든 정책에 영향을 미칩니다.

수동으로 플릿 용량을 설정하려면

Console

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿의 세부 정보 페이지를 열기 위해 활성 플릿의 이름을 클릭합니다.
4. 크기 조정 탭의 일시 중지된 Auto Scaling 위치에서 Auto Scaling을 일시 중단하려는 각 위치를 선택한 다음 일시 중지를 선택합니다.
5. 크기 조정 용량에서 수동으로 설정하려는 위치를 선택한 다음 편집을 선택합니다.
6. 크기 조정 용량 편집 대화 상자에서 원하는 인스턴스의 기본 값을 설정한 다음 확인을 선택합니다. 이를 통해 활성 상태로 유지하고 게임 세션 호스팅에 대한 준비를 할 인스턴스 수를 Amazon GameLift에 알려줍니다.

   Amazon GameLift는 추가 인스턴스를 배포하거나 불필요한 인스턴스를 종료하여 변화에 대응합니다. Amazon GameLift에서 이 프로세스를 완료하면 해당 위치의 활성 인스턴스 수가 업데이트된 원하는 인스턴스 값과 일치하도록 변경됩니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.

AWS CLI

1. 현재 용량 설정을 확인합니다. 명령줄 창에서 용량 변경을 원하는 플릿 ID와 함께 describe-fleet-location-capacity 명령을 사용합니다. 이 명령은 해당 위치에 대한 현재 용량 설정이 포함된 FleetCapacity 객체를 반환합니다. 인스턴스 제한 값이 원하는 새 인스턴스에 대한 설정을 수용할 수 있는지 확인합니다.

   ```
   aws gamelift describe-fleet-location-capacity \
       --fleet-id <fleet identifier> \
       --location <location name>
   ```

2. 원하는 용량으로 업데이트합니다. update-fleet-capacity 명령과 함께 플릿 ID, 위치, 원하는 인스턴스의 새 값을 사용합니다. 이 값이 현재 제한 값 범위 밖일 경우 동일한 명령에 제한 값을 조정할 수 있습니다.

```
--fleet-id <fleet identifier>
--location <location name>
--desired-instances <fleet capacity as an integer>
--max-size <maximum capacity>    [Optional]
--min-size <minimum capacity>    [Optional]
```

예제:

```
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
```

요청이 성공하면 Amazon GameLift이 플릿 ID를 반환합니다. 새 원하는 인스턴스 설정이 최소/최대 제한 밖일 경우 Amazon GameLift는 오류를 반환합니다.

Amazon GameLift로 플릿 용량 Auto Scaling

Amazon GameLift의 Auto Scaling을 사용하여 게임 서버상의 활동에 따라 동적으로 플릿 용량을 조정합니다. 플레이어가 도착해 게임 세션을 시작하면 Auto Scaling으로 더 많은 인스턴스를 추가하고, 플레이어의 수요가 줄어들면 Auto Scaling을 통해 불필요한 인스턴스를 종료할 수 있습니다. Auto Scaling은 호스팅 리소스와 비용을 최소화하면서, 원활하고 빠른 플레이어 경험을 제공할 수 있는 효과적인 방법입니다.

Auto Scaling을 사용하려면 Amazon GameLift에 스케일 업 또는 스케일 다운 시기를 알려주는 조정 정책을 생성해야 합니다. 대상 기반과 규칙 기반이라는 두 가지 조정 정책이 있습니다. 목표 기반 접근 방식인 대상 추적은 완벽한 솔루션입니다. 가장 간단하고 효과적인 옵션으로 사용하는 것이 좋습니다. Auto Scaling 의사 결정 프로세스의 각 요소를 정의해야 하는 규칙 기반 조정 정책은 특정 문제를 다루는 데 유용합니다. 이 솔루션은 대상 기반 Auto Scaling을 보완할 때 가장 좋습니다.

Amazon GameLift 콘솔, AWS Command Line Interface(AWS CLI), 또는 AWS SDK를 사용하여 대상 기반 Auto Scaling을 관리할 수 있습니다. 규칙 기반 Auto Scaling은 콘솔에서 규칙 기반 조정 정책을 확인할 수 있는 경우에도 AWS CLI 또는 AWS SDK 중 하나만 사용하여 관리할 수 있습니다.

주제

대상 기반 Auto Scaling

Amazon GameLift의 대상 기반 Auto Scaling은 플릿 지표 `PercentAvailableGameSessions`에 따라 용량 수준을 조정합니다. 이 지표는 갑작스러운 플레이어 수요 증가에 대한 플릿의 사용 가능한 버퍼를 나타냅니다.

용량 버퍼를 유지해야 하는 기본적인 이유는 플레이어 대기 시간 때문입니다. 게임 세션 슬롯이 준비되어 대기하고 있는 경우, 새 플레이어의 게임 세션 참여에는 몇 초 정도 소요됩니다. 리소스를 사용할 수 없는 경우, 플레이어는 기존 게임 세션이 끝나거나, 새 리소스를 사용할 수 있을 때까지 대기해야 합니다. 새 인스턴스 및 서버 프로세스 시작에 몇 분이 소요될 수 있습니다.

대상 기반 Auto Scaling을 설정하는 경우 플릿에서 유지하려는 버퍼 크기만 지정합니다. `PercentAvailableGameSessions`이 사용 가능한 리소스의 백분율을 측정하기 때문에 실제 버퍼 크기는 전체 플릿 용량의 백분율입니다. Amazon GameLift는 대상 버퍼 크기를 유지하기 위해 인스턴스를 추가하거나 제거합니다. 버퍼 크기가 클 수록 대기 시간이 최소화되지만, 동시에 사용하지 않을 수도 있는 추가 리소스에 대한 요금을 지불해야 합니다. 플레이어가 대기 시간에 대한 참을성이 크다면 버퍼 크기를 작게 설정해 비용을 낮출 수 있습니다.

대상 기반 Auto Scaling을 설정하려면

Console

1. Amazon GameLift 콘솔을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿의 세부 정보 페이지를 열기 위해 활성 플릿의 이름을 클릭합니다.
4. 크기 조정 탭을 선택합니다. 탭에 플릿의 과거 조정 지표가 표시되며, 여기에는 현재 조정 설정을 조정할 수 있는 컨트롤이 포함되어 있습니다.
5. 크기 조정 용량에서 최소 크기 및 최대 크기 제한이 플릿에 적합한지 확인합니다. Auto Scaling이 활성화되어 있는 경우, 이 두 제한 값 사이에서 용량이 조정됩니다.

6. 대상 기반 자동 크기 조정 정책에서 편집을 선택합니다.
7. 대상 기반 자동 크기 조정 정책 편집 대화 상자에서 사용 가능한 게임 세션 비율(%) 대해 유지하려는 비율을 설정한 다음 확인을 선택합니다. 설정을 확인한 후 Amazon GameLift는 대상 기반 자동 크기 조정 정책 하에 새로운 대상 기반 정책을 추가합니다.

AWS CLI

1. 용량 제한을 설정합니다. update-fleet-capacity 명령을 사용하여 제한 값을 설정합니다. 자세한 내용은 Amazon GameLift의 용량 제한 설정 섹션을 참조하세요.
2. 새 정책 생성. 명령줄 창을 열고 put-scaling-policy 명령과 함께 사용자의 정책 파라미터 설정을 사용합니다. 기존 정책을 업데이트하려면 정책 이름을 지정하고, 완전히 업데이트한 정책 버전을 제공합니다.

```
--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>
```

예제:

```
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"
```

규칙 기반 정책을 사용한 Auto Scaling

Amazon GameLift의 규칙 기반의 조정 정책을 사용하면 플레이어의 활동에 맞춰 플릿 용량의 Auto Scaling을 세부적으로 제어할 수 있습니다. 각 정책에서 여러 플릿 지표 중 하나와 조정을 연결시키고, 트리거 지점을 식별하고, 응답할 스케일 업 또는 스케일 다운 이벤트를 사용자 지정할 수 있습니다. 규칙 기반의 정책은 특정 상황을 처리하기 위해 대상 기반 조정 방식을 보완하는 데 유용합니다.

규칙 기반 정책은 “플릿 지표가 일정 시간 동안 임계 값에 도달하거나 이를 초과하면 플릿 용량을 일정 수치만큼 변경합니다.”로 명시할 수 있습니다. 이번 주제에서는 정책 설명을 만들기 위해 사용하는 구문을 설명하고, 규칙 기반 정책 생성 및 관리에 도움을 주는 정보를 제공합니다.

규칙 기반 정책 관리

Amazon GameLift 서비스 API의 AWS SDK 또는 AWS Command Line Interface(AWS CLI)을 사용하여 규칙 기반 정책을 생성, 업데이트 또는 삭제합니다. Amazon GameLift 콘솔에서 모든 활성 정책을 확인할 수 있습니다.

특정 플릿에 대한 모든 조정 정책을 일시적으로 중지하려면 AWS CLI 명령인 stop-fleet-actions를 사용합니다.

규칙 기반 조정 정책(AWS CLI)을 생성 또는 업데이트하려면

1. 용량 제한을 설정합니다. update-fleet-capacity 명령을 사용하여 두 제한 값 중 하나 또는 둘 모두를 설정합니다. 자세한 내용은 Amazon GameLift의 용량 제한 설정 섹션을 참조하세요.
2. 새 정책 생성. 명령줄 창을 열고 put-scaling-policy 명령과 함께 사용자의 정책 파라미터 설정을 사용합니다. 기존 정책을 업데이트하려면 정책 이름을 지정하고, 완전히 업데이트한 정책 버전을 제공합니다.

```
--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>
```

예제:

```
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
```

AWS CLI를 사용하여 규칙 기반 조정 정책을 삭제하려면

- 명령줄 창을 열고 delete-scaling-policy 명령과 함께 플릿 ID와 정책 이름을 사용합니다.

  예제:

```
aws gamelift delete-scaling-policy \
    --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --name "Scale up when AGS<50"
```

Auto Scaling 규칙 구문

규칙 기반 조정 정책 설명을 구성하려면 다음과 같이 여섯 가지 주요 변수를 지정합니다.

*<metric name>*이 *<comparison operator>* 동안 *<threshold value>*로 유지되면 *<evaluation period>*을 이용하여 플릿 용량을 *<adjustment type>*로/만큼 *<adjustment value>* 변경합니다.

예를 들어, 이 정책 설명은 플릿의 추가 용량이 50개의 새 게임 세션 처리에 필요한 용량보다 적을 때마다 스케일 업 이벤트를 시작합니다.

`AvailableGameSessions`가 `less than 50` 동안 `10 minutes`로 유지되면 `ChangeInCapacity`를 이용하여 플릿 용량을 `1 instances`만큼 변경합니다.

지표 이름

조정 이벤트를 시작하려면 플릿에 특정적인 다음 지표 중 하나에 Auto Scaling 정책을 연결합니다. 지표에 대한 자세한 설명은 Amazon GameLift 플릿 지표 섹션을 참조하세요.

- 게임 세션 활성화
- 활성 게임 세션
- 사용 가능한 게임 세션
- 사용 가능한 게임 세션 백분율
- 활성 인스턴스
- 사용 가능한 플레이어 세션
- 현재 플레이어 세션
- 유휴 인스턴스

- 유휴 인스턴스 백분율

플릿이 게임 세션 대기열에 있는 경우 다음 지표를 사용할 수 있습니다.

- 대기열 깊이 - 이 플릿이 사용 가능한 최상의 호스팅 위치인 대기 중인 게임 세션 요청의 수.
- 대기 시간 - 플릿별 대기 시간. 충족해야 할 가장 오래 대기 중인 게임 세션 요청의 대기 시간입니다. 플릿의 대기 시간은 가장 오래된 현재 요청의 대기열 체류 시간과 동일합니다.

비교 연산자

임계값을 기준으로 지표 데이터를 비교하는 방법을 Amazon GameLift에 제공합니다. 유효한 비교 연산자로는 초과(>), 미만(<), 이상(>=) 및 이하(<=) 등이 있습니다.

임계값

지정 지표값이 임계값에 도달하거나 초과할 때 조정 이벤트를 시작합니다. 이 값은 항상 양의 정수입니다.

평가 기간

지표가 전체 평가 기간의 임계값에 도달하거나 이를 초과해야 조정 이벤트가 시작됩니다. 평가 기간은 연속적입니다. 지표가 임계값보다 작아지면 평가 기간이 처음부터 다시 시작됩니다.

조정 유형 및 값

이 변수 세트는 조정 이벤트가 시작될 때 Amazon GameLift가 플릿 용량을 조정하는 방식을 지정합니다. 다음 세 가지 조정 유형 중에서 선택할 수 있습니다.

- 용량 변경 - 현재 용량을 지정된 인스턴스 수만큼 늘리거나 줄입니다. 플릿에 추가하거나 플릿에서 제거할 인스턴스 수로 조정 값을 설정합니다. 양수 값은 인스턴스를 추가하고, 음수 값은 인스턴스를 제거합니다. 예를 들어, “-10”은 플릿의 총 크기에 상관 없이 플릿을 10개 인스턴스씩 축소시킵니다.
- 용량의 백분율 변경 - 현재 용량을 지정된 백분율만큼 늘리거나 줄입니다. 플릿 용량에 늘리거나 줄이기를 원하는 만큼의 백분율로 조정 값을 설정합니다. 양수 값은 인스턴스를 추가하고, 음수 값은 인스턴스를 제거합니다. 예를 들어, 50개 인스턴스의 플릿에서 백분율 변경 값이 “20”이면 플릿에 10개 인스턴스가 추가됩니다.
- 정확한 용량 - 현재 용량을 특정 값으로 늘리거나 줄입니다. 플릿에 유지할 정확한 인스턴스 수로 조정 값을 설정합니다.

규칙 기반 Auto Scaling에 대한 팁

다음 제안은 규칙 기반 정책을 사용한 Auto Scaling을 최대한 활용하는 데 도움이 됩니다.

다수의 정책 적용

한 플릿에 여러 Auto Scaling 정책을 동시에 적용할 수 있습니다. 가장 일반적인 시나리오는 대상 기반 정책으로 대부분의 조정 필요 사항을 관리하고, 규칙 기반 정책으로 엣지 사례를 처리하는 것입니다. 여러 정책을 사용할 수도 있습니다.

여러 정책이 있는 경우에는 각 정책이 독립적으로 작동합니다. 조정 이벤트의 순서를 제어하는 방법은 없습니다. 예를 들어 확장을 관리하는 정책이 여럿인 경우, 플레이어 활동으로 인해 여러 조정 이벤트가 동시에 시작될 수 있습니다. 서로 동시에 시작하는 정책을 피하세요. 예를 들어, 서로의 임계값을 넘어선 용량을 설정하는 스케일 업 및 스케일 다운 정책을 생성한 경우, 무한 루프가 발생할 수 있습니다.

최대 및 최소 용량 설정

각 플릿의 최대 및 최소 용량에 제한이 있습니다. 이 기능은 Auto Scaling을 사용할 때 중요합니다. Auto Scaling은 이 범위에서 벗어나는 값으로 용량을 설정하지 않습니다. 기본적으로 새롭게 생성된 플릿의 최소 용량은 0, 최대 용량은 1입니다. Auto Scaling 정책이 의도한 대로 용량에 적용되도록 하려면 최대 값을 늘립니다.

플릿 용량은 플릿의 인스턴스 유형 및 AWS 계정의 Service Quotas에 의해서도 제한됩니다. 이러한 제한 사항 및 계정 할당량에서 벗어나는 최소 및 최대 용량을 설정할 수 없습니다.

용량 변경 후 측정치 추적

Amazon GameLift는 Auto Scaling 정책에 따라 용량을 변경하고 10분을 대기한 후 동일한 정책의 트리거에 반응합니다. 이렇게 대기하면서 Amazon GameLift는 새 인스턴스 추가, 게임 서버 시작, 플레이어 연결, 새 인스턴스로부터 데이터 수집 시작 등을 수행합니다. 그 동안 Amazon GameLift는 지표를 기준으로 정책을 평가하고 정책 평가 기간을 추적합니다. 정책 평가 기간은 조정 이벤트가 발생하면 다시 시작됩니다. 즉, 대기 시간이 끝난 직후 조정 정책이 다른 조정 이벤트를 시작할 수 있습니다.

서로 다른 Auto Scaling 정책에서 시작한 조정 이벤트 사이에는 대기 시간이 없습니다.

게임 세션 배치에 대한 Amazon GameLift 대기열 설정

게임 세션 대기열은 새 게임 세션 요청을 처리하고 이를 호스팅할 수 있는 게임 서버를 찾기 위한 기본 메커니즘입니다. 대기열은 게임 개발자와 플레이어에게 상당한 이점을 제공합니다. 다음이 포함됩니다.

- 대기열은 최상의 배치를 제공합니다. 게임 세션 배치 요청을 처리할 때 대기열은 Amazon GameLift 알고리즘을 사용하여 정의된 기본 설정 세트를 기반으로 대기열 위치의 우선 순위를 지정합니다.

- 저렴한 스팟 플릿에서 게임을 호스팅할 수 있습니다. 대기열을 사용하면 호스팅 비용이 크게 절감되는 AWS 스팟 플릿의 사용을 최적화할 수 있습니다. 기본적으로 대기열은 항상 스팟 플릿에서 새 게임 세션을 배치하려고 합니다.
- 수요가 많을 때는 새 게임을 더 빨리 배치할 수 있습니다. 대기열에서는 여러 위치를 배치할 수 있습니다. 즉, 선호하는 배치 위치를 사용할 수 없는 경우 항상 대체 용량을 확보할 수 있습니다.
- 게임 가용성을 보다 신속하게 복원할 수 있습니다. 중단 사태가 발생할 수 있습니다. 다중 위치 대기열을 사용하면 감속이나 정전으로 인해 플레이어의 게임 액세스에 영향이 미치지 않습니다.
- 추가적인 플릿 용량을 보다 효율적으로 사용할 수 있습니다. 예상치 못한 플레이어 수요의 급증을 처리하려면 대기열이 추가 호스팅 용량에 신속하게 액세스할 수 있어야 합니다. 대기열에 있는 플릿 위치는 서로 백업 용량을 제공합니다. 플레이어 수요에 따라 위치를 늘리거나 줄일 수 있습니다.
- 게임 세션 배치 및 대기열 성능에 대한 지표를 확인할 수 있습니다. Amazon GameLift는 배치 성패 여부에 대한 통계, 대기열의 요청 수, 대기열에서 요청이 기다리는 평균 시간 같은 대기열 지표를 출력합니다. Amazon GameLift 콘솔 또는 CloudWatch에서 이러한 지표를 볼 수 있습니다.

대기열을 시작하려면 게임 세션 대기열 설계 섹션을 참조하세요.

게임 세션 대기열 설계

이 주제에서는 지연 시간을 최소화하면서 플레이어 경험을 제공하고 호스팅 리소스를 효율적으로 사용하는 대기열을 설계하는 방법에 대해 설명합니다. 게임 세션 대기열 및 작동 방식에 대한 자세한 내용은 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.

이러한 Amazon GameLift 기능에는 다음과 같은 대기열이 필요합니다.

- FlexMatch를 사용한 매치메이킹
- Amazon에서 스팟 인스턴스 사용 GameLift

대기열의 범위 정의

게임의 플레이어 모집단에는 함께 플레이해서는 안 되는 플레이어 그룹이 있을 수 있습니다. 예를 들어 두 가지 언어로 게임을 게시하는 경우 각 언어에 자체 게임 서버가 있어야 합니다.

플레이어 인구를 대상으로 게임 세션 배치를 설정하려면 각 플레이어 세그먼트에 대해 별도의 대기열을 생성합니다. 각 대기열의 범위를 지정하여 플레이어를 올바른 게임 서버에 배치합니다. 대기열의 범위를 지정하는 몇 가지 일반적인 방법은 다음과 같습니다.

- 지리적 위치별. 여러 지역에 게임 서버를 배포할 때는 각 위치에 플레이어를 위한 대기열을 만들어 플레이어 지연 시간을 줄일 수 있습니다.
- 빌드 또는 스크립트 변형별. 게임 서버의 변형이 두 개 이상이면 동일한 게임 세션에서 플레이할 수 없는 플레이어 그룹을 지원하는 것일 수 있습니다. 예를 들어 게임 서버 빌드나 스크립트는 다양한 언어 또는 디바이스 유형을 지원할 수 있습니다.
- 이벤트 유형별. 토너먼트 또는 기타 특별 이벤트 참가자의 게임을 관리하기 위해 특별 대기열을 만들 수 있습니다.

플레이어 대기 정책 생성

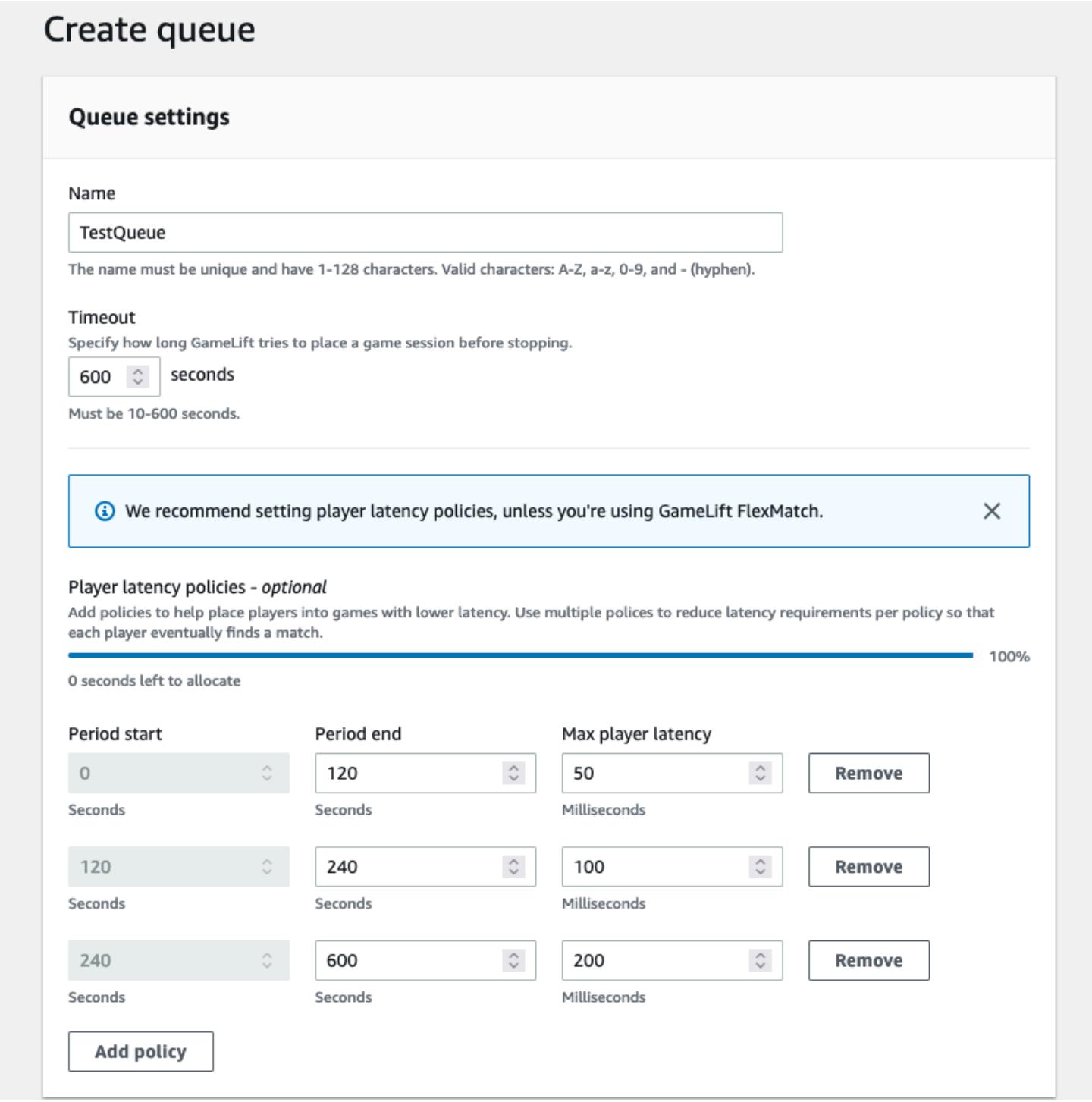
배치 요청에 플레이어 지연 시간 데이터가 포함된 경우 알고리즘은 모든 플레이어의 평균 지연 시간이 가장 낮은 위치에서 게임 세션을 찾습니다. 평균 플레이어 지연 시간을 기준으로 게임 세션을 배치하면 Amazon GameLift에서 지연 시간이 긴 게임에 대부분의 플레이어를 배치하지 못합니다. 하지만 Amazon GameLift는 여전히 플레이어에게 극심한 지연 시간을 안겨줍니다. 이러한 플레이어를 수용하려면 플레이어 지연 시간 정책을 생성해야 합니다.

플레이어 지연 시간 정책을 통해 Amazon GameLift는 요청 대상 플레이어가 최대값을 초과하는 지연 시간을 경험할 수 있는 위치에 요청된 게임 세션을 배치하지 못하게 됩니다. 플레이어 지연 시간 정책은 Amazon GameLift가 게임 세션 요청을 지연 시간이 더 긴 플레이어와 매칭하지 못하게 할 수도 있습니다.

> **Tip**
>
> 지연 시간 관련 규칙(예: 그룹 내 모든 플레이어에 대해 유사한 지연 시간을 요구하는 경우)을 관리하기 위해 Amazon GameLift FlexMatch를 사용하여 지연 시간 기반 매치메이킹 규칙을 생성할 수 있습니다.

제한 시간이 5분이고 플레이어 지연 시간 정책이 다음과 같은 대기열을 예로 들어 보겠습니다.

1. 모든 플레이어 지연 시간이 50밀리초 미만인 위치를 검색하는 데 120초를 소요합니다.
2. 모든 플레이어 지연 시간이 100밀리초 미만인 위치를 검색하는 데 120초를 소요합니다.
3. 모든 플레이어 지연 시간이 200밀리초 미만인 위치를 검색하는 데 남은 대기열 제한 시간을 소요합니다.



다중 위치 대기열 빌드

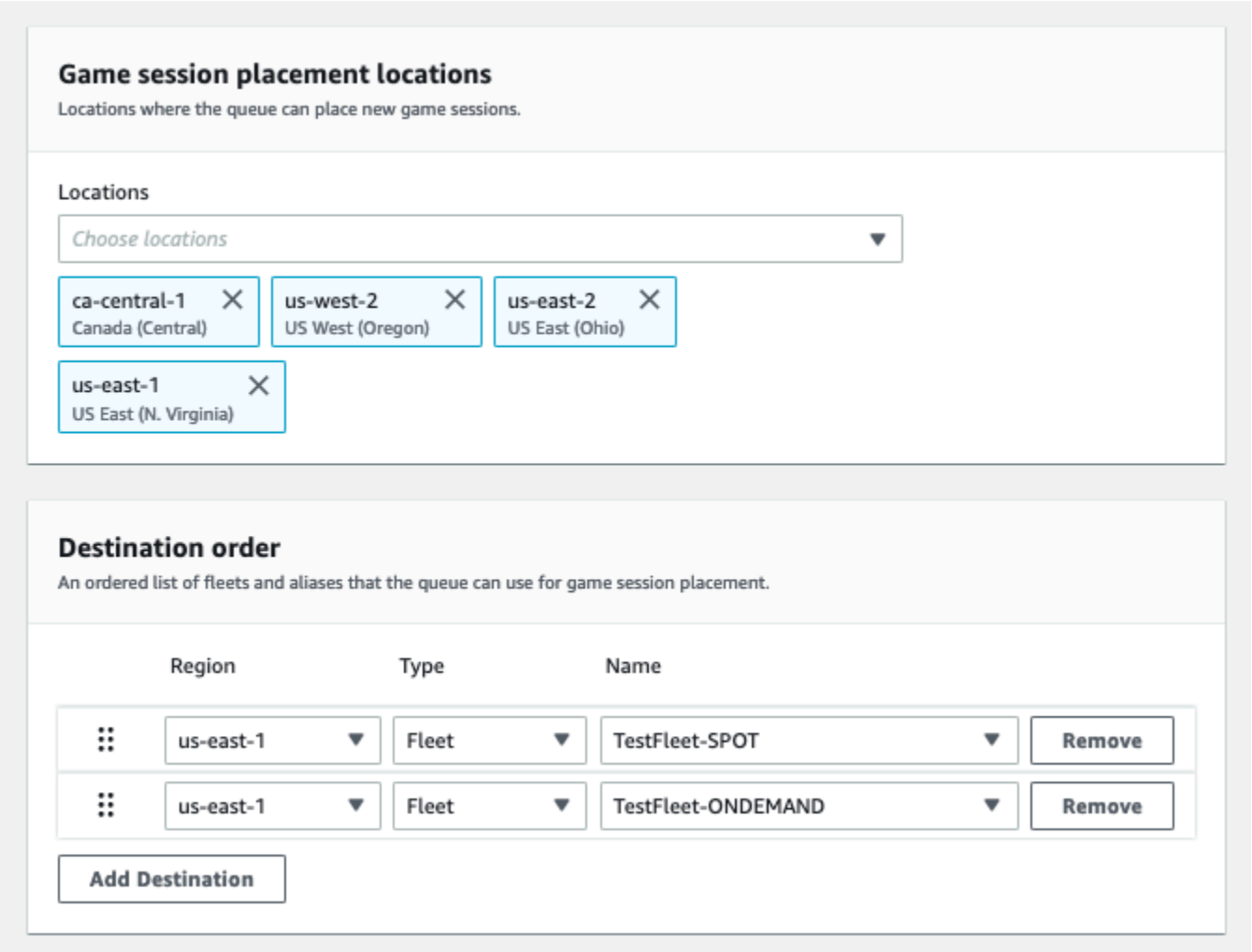
모든 대기열에 다중 위치 설계를 사용하는 것이 좋습니다. 이 설계를 통해 배치 속도와 호스팅 복원성을 개선할 수 있습니다. 플레이어 지연 시간 데이터를 사용하여 플레이어를 지연 시간을 최소화하면서 게임 세션에 참여시키려면 다중 위치 설계가 필요합니다. 스팟 인스턴스 플릿을 사용하는 다중 위치 대기열을 구축하는 경우 자습서: 스팟 인스턴스용 게임 세션 대기열 설치의 지침을 따릅니다.

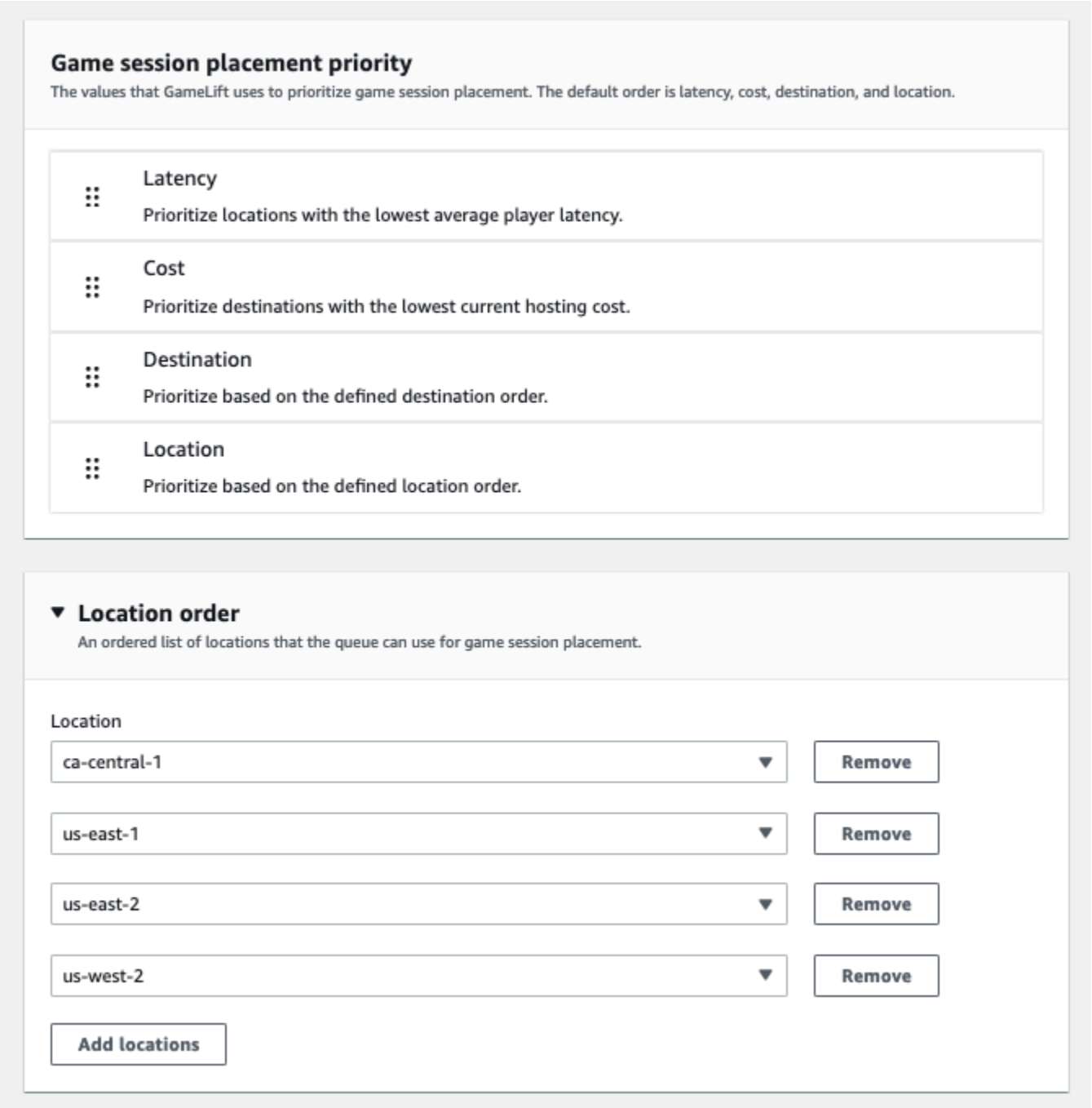
다중 위치 대기열을 생성하는 한 가지 방법은 대기열에 다중 위치 플릿을 추가하는 것입니다. 이렇게 하면 대기열에서 플릿의 어느 위치에나 게임 세션을 배치할 수 있습니다. 중복성을 위해 구성이나 홈 위치가 다른 플릿을 추가할 수도 있습니다. 다중 위치 스팟 인스턴스 플릿을 사용하는 경우 모범 사례를 따르고 동일한 위치의 온디맨드 인스턴스 플릿을 포함합니다.

다음 예는 기본 다중 위치 대기열을 설계하는 프로세스에 대해 간략하게 설명합니다. 이 예에서는 스팟 인스턴스 플릿 하나와 온디맨드 인스턴스 플릿 하나, 이렇게 두 플릿을 사용합니다. 각 플릿에는 다음 AWS 리전의 배치 위치가 있습니다. `us-east-1`, `us-east-2`, `ca-central-1`, `us-west-2`.

다중 위치 플릿이 포함된 기본 다중 위치 대기열을 생성하려면

1. 대기열을 생성할 위치를 선택합니다. 클라이언트 서비스를 배포한 위치 근처에 대기열을 배치하여 요청 지연 시간을 최소화할 수 있습니다. 이 예제에서는 `us-east-1`에서 대기열을 만듭니다.
2. 새 대기열을 생성하고 대기열 대상으로 다중 위치 플릿을 추가합니다. 대상 순서는 Amazon GameLift가 게임 세션을 배치하는 방법에 대해 결정합니다. 이 예에서는 스팟 인스턴스 플릿을 먼저 나열하고 온디맨드 인스턴스 플릿을 두 번째로 나열합니다.
3. 대기열의 게임 세션 배치 우선 순위를 정의합니다. 이 순서는 대기열이 사용 가능한 게임 서버를 가장 먼저 검색하는 위치를 결정합니다. 이 예제에서는 기본 우선 순위 순서를 사용합니다.
4. 위치 순서를 정의합니다. 위치 순서를 정의하지 않는 경우 Amazon GameLift는 위치를 알파벳 순서대로 사용합니다.


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



게임 세션 배치 우선순위

Amazon GameLift는 FleetIQ 알고리즘을 사용하여 정렬된 기준 세트를 기반으로 새 게임 세션을 배치할 위치를 결정합니다. 기본 우선 순위 순서를 사용하거나 순서를 사용자 지정할 수 있습니다.

기본 우선 순위 순서

플레이어 지연 시간 데이터가 포함된 배치 요청의 경우 FleetIQ는 다음과 같은 기본 순서로 게임 세션 배치 기준의 우선 순위를 지정합니다.

1. 지연 시간 - 요청에 포함된 모든 플레이어의 가장 짧은 평균 지연 시간입니다.
2. 비용 - 다중 위치에서 지연 시간이 같을 경우 가장 저렴한 호스팅 비용입니다. 호스팅 비용은 주로 인스턴스 유형과 위치의 조합을 기준으로 합니다.
3. 대상 - 다중 위치에서 지연 시간과 비용이 동일한 경우 대상 순서입니다. FleetIQ는 대기열 구성에 나열되는 순서에 따라 대상의 우선 순위를 지정합니다.
4. 대상 - 다중 위치에서 지연 시간, 비용, 대상이 동일한 경우 위치 순서입니다. FleetIQ는 대기열 구성에 나열되는 순서에 따라 위치의 우선 순위를 지정합니다.

사용자 지정 우선 순위 순서

Amazon GameLift 콘솔에서 대기열의 우선 순위 순서를 사용자 지정하려면 우선 순위 값을 원하는 위치로 드래그합니다. AWS Command Line Interface(AWS CLI)를 사용하여 대기열의 우선 순위 순서를 사용자 지정하려면 create-game-session-queue 명령을 --priority-configuration 옵션과 함께 사용합니다. 이 명령을 사용하여 새 대기열을 만들거나 기존 대기열을 업데이트할 수 있습니다.

FleetIQ 알고리즘은 명시적으로 언급되지 않은 모든 기준을 기본 순서에 따라 목록 끝에 추가합니다. 우선 순위 구성에 위치 기준을 포함하는 경우 순서가 지정된 위치 목록도 제공해야 합니다.

필요에 따라 여러 대기열 설계

게임과 플레이어에 따라 게임 세션 대기열을 두 개 이상 만들어야 할 수도 있습니다. 게임 클라이언트 서비스가 새로운 게임 세션을 요청하면 어떤 게임 세션 대기열을 사용하는지 지정합니다. 대기열을 여러 개 사용할지 여부를 결정하는 데 도움이 되려면 다음을 고려합니다.

- 게임 서버의 변형. 게임 서버의 각 변형에 대해 별도의 대기열을 만들 수 있습니다. 대기열에 있는 모든 플릿은 호환 가능한 게임 서버를 배포해야 합니다. 대기열을 사용하여 게임에 참가하는 플레이어는 해당 대기열의 모든 게임 서버에서 플레이할 수 있어야 하기 때문입니다.
- 다양한 플레이어 그룹. Amazon GameLift가 플레이어 그룹을 기반으로 게임 세션을 배치하는 방법을 사용자 지정할 수 있습니다. 예를 들어 특별한 인스턴스 유형이나 런타임 구성이 필요한 특정 게임 모드에 맞게 사용자 지정된 대기열이 필요할 수 있습니다. 또는 토너먼트나 기타 이벤트의 배치를 관리하기 위한 특수 대기열을 원할 수도 있습니다.
- 게임 세션 대기열 지표. 게임 세션 배치 지표를 수집하려는 방식에 따라 대기열을 설정할 수 있습니다. 자세한 내용은 Amazon GameLift 대기열 지표 섹션을 참조하세요.

대기열 지표 평가

자체 대기열이 잘 수행되고 있는지 지표를 통해 평가하십시오. Amazon GameLift 콘솔 또는 Amazon CloudWatch에서 대기열과 관련된 지표를 볼 수 있습니다. 대기열 지표의 목록 및 설명은 Amazon GameLift 대기열 지표 섹션을 참조하세요.

대기열 지표는 다음에 대한 인사이트를 제공할 수 있습니다.

- 전체 대기열 성능 - 대기열 지표는 대기열이 배치 요청에 얼마나 성공적으로 응답하는지를 나타냅니다. 또한 이러한 지표는 배치가 실패하는 시기와 이유를 식별하는 데도 도움이 될 수 있습니다. 플릿을 수동으로 크기 조정한 대기열의 경우 `AverageWaitTime` 및 `QueueDepth` 지표를 통해 대기열 용량을 조정해야 하는 시기를 표시할 수 있습니다.
- FleetIQ 알고리즘 성능 - FleetIQ 알고리즘을 사용하는 배치 요청의 경우, 지표는 알고리즘이 이상적인 게임 세션 배치를 찾는 빈도를 보여줍니다. 플레이어 지연 시간이 가장 짧은 리소스 또는 비용이 가장 저렴한 리소스를 우선적으로 사용하여 배치할 수 있습니다. Amazon GameLift가 이상적인 배치를 찾지 못하는 일반적인 이유를 식별하는 오류 지표도 있습니다. 지표에 대한 자세한 정보는 Amazon CloudWatch를 사용한 Amazon GameLift 모니터링 섹션을 참조하세요.
- 위치 관련 배치 - 다중 위치 대기열의 경우 지표는 위치별 성공적인 배치를 보여줍니다. FleetIQ 알고리즘을 사용하는 대기열의 경우, 이 데이터는 플레이어 활동이 발생하는 곳에 대한 유용한 인사이트를 제공합니다.

FleetIQ 알고리즘 성능의 지표를 평가할 때는 다음 팁을 고려합니다.

- 대기열이 이상적인 배치를 찾는 속도를 추적하려면 가장 짧은 지연 시간과 최저 비용을 위해 `PlacementsSucceeded` 지표를 FleetIQ 지표와 함께 사용합니다.
- 대기열이 이상적인 배치를 찾는 속도를 높이려면 다음 오류 지표를 검토합니다.
  - `FirstChoiceOutOfCapacity`값이 높으면 대기열 플릿에 맞게 용량 크기를 조정합니다.
  - `FirstChoiceNotViable` 오류 지표가 높으면 스팟 인스턴스 플릿을 살펴봅니다. 특정 인스턴스 유형의 간섭 속도가 매우 높을 경우 스팟 인스턴스 플릿은 사용 가능하지 않은 상태로 여겨집니다. 이 문제를 해결하려면 대기열을 바꿔 다른 인스턴스 유형을 통해 스팟 인스턴스 플릿을 사용합니다. 위치마다 인스턴스 유형이 다른 스팟 인스턴스 플릿을 포함하는 것이 좋습니다.

Amazon GameLift의 게임 세션 대기열에 대한 모범 사례

다음은 게임 세션 배치를 위한 효과적인 게임 세션 대기열을 구축하는 데 도움이 되는 몇 가지 모범 사례입니다.

모든 플릿 유형의 대기열에 대한 모범 사례

대기열에는 새 게임 세션이 배치될 수 있는 플릿 대상 목록이 포함되어 있습니다. 각 플릿에는 여러 지리적 위치에 인스턴스를 배포할 수 있습니다. 배치를 선택할 때 대기열은 플릿과 플릿 위치의 조합을 선택합니다. 배치를 선택할 때 사용할 대기열의 우선 순위 세트를 제공합니다.

다음과 같은 지침과 모범 사례를 고려하세요.

- 플레이어가 커버하는 위치에 플릿을 추가합니다. 사용 가능한 모든 위치에 플릿과 별칭을 추가할 수 있습니다. 보고된 플레이어 지연 시간을 기준으로 배치를 하려면 위치가 중요합니다.
- 모든 플릿에 별칭을 사용합니다. 대기열에 있는 각 플릿에 별칭을 할당하고, 대기열에 대상을 설정할 때 별칭 이름을 사용합니다.
- 모든 플릿에 동일하거나 유사한 게임 빌드나 스크립트를 사용합니다. 대기열을 사용하면 플레이어가 대기열에 있는 모든 플릿의 게임 세션에 참가할 수 있습니다. 플레이어는 모든 플릿의 모든 게임 세션에서 플레이할 수 있어야 합니다.
- 최소 두 군데의 위치에서 플릿을 생성합니다. 게임 서버를 최소 한 곳 이상의 다른 위치에 호스팅하면, 리전 중단으로 플레이어에게 미치는 영향을 완화할 수 있습니다. 백업 플릿을 계속 축소할 수 있으며, 사용량이 증가하면 Auto Scaling을 사용하여 용량을 늘릴 수 있습니다.
- 게임 세션 배치의 우선 순위를 지정합니다. 대기열은 대상 목록 순서를 비롯한 여러 요소를 기반으로 배치 선택의 우선 순위를 정합니다.
- 클라이언트 서비스와 동일한 위치에 대기열을 생성합니다. 클라이언트 서비스 근처의 위치에 대기열을 배치하면 통신 지연 시간을 최소화할 수 있습니다.
- 여러 위치에 있는 플릿을 사용합니다. 대기열 필터 구성을 사용하면 대기열이 게임 세션을 지정된 위치에 배치하지 않도록 할 수 있습니다. 홈 위치가 다른 여러 위치에 있는 플릿을 두 개 이상 사용하면 리전 중단 시 게임 배치에 미치는 영향을 완화할 수 있습니다.
- 모든 플릿에 동일한 TLS 인증서 설정을 사용합니다. 플릿의 게임 세션에 연결하는 게임 클라이언트에는 호환되는 통신 프로토콜이 있어야 합니다.

스팟 플릿의 대기열에 대한 모범 사례

대기열에 스팟 플릿이 포함된 경우 복원력이 높은 대기열을 설정합니다. 이렇게 하면 스팟 플릿을 통해 비용을 절감하는 동시에 게임 세션 중단으로 인한 영향을 최소화할 수 있습니다. 스팟 플릿과 함께 사용할 플릿 및 게임 세션 대기열을 올바르게 구축하는 데 도움이 필요한 경우 자습서: 스팟 인스턴스용 게임 세션 대기열 설치 섹션을 참조하세요. 스팟 인스턴스에 대한 자세한 내용은 Amazon에서 스팟 인스턴스 사용 GameLift 섹션을 참조하세요.

이전 섹션의 일반적인 모범 사례 외에도 다음과 같은 스팟별 모범 사례를 고려해 봅니다.

- 각 위치에 온디맨드 플릿을 하나 이상 생성합니다. 온디맨드 플릿은 플레이어를 위한 백업 게임 서버를 제공합니다. 필요할 때까지 백업 플릿을 축소하고, 스팟 플릿을 사용할 수 없을 때는 Auto Scaling을 사용하여 온디맨드 용량을 늘릴 수 있습니다.
- 한 위치의 여러 스팟 플릿에서 서로 다른 인스턴스 유형을 선택합니다. 한 스팟 인스턴스 유형을 일시적으로 사용할 수 없게 되는 경우 중단은 해당 위치의 한 스팟 플릿에만 영향을 미칩니다. 모범 사례로는 널리 사용 가능한 인스턴스 유형을 선택하고 동일한 패밀리의 인스턴스 유형(예: m5.large, m5.xlarge, m5.2xlarge) 을 사용하는 것입니다. Amazon GameLift 콘솔을 사용하여 인스턴스 유형에 대한 요금 데이터 기록을 볼 수 있습니다.

게임 세션 대기열 생성

대기열은 여러 플릿 및 리전에서 사용 가능한 최적의 호스팅 리소스로 새 게임 세션을 배치하는 데 사용됩니다. 게임에서 대기열을 빌드하는 방법에 대한 자세한 내용은 게임 세션 대기열 설계 섹션을 참조하세요.

게임 클라이언트에서는 배치 요청을 사용하여 대기열에서 새로운 게임 세션이 시작됩니다. 게임 세션 만들기 섹션에서 게임 세션 배치에 대한 자세한 내용을 알아봅니다.

대기열의 대기열 대상을 업데이트할 때 짧은 전환 기간(최대 30초)이 발생하며, 이 기간 동안 대기열 대상에 배치된 게임 세션이 이전 플릿으로 넘어갈 수 있습니다.

Console

1. Amazon GameLift 콘솔의 탐색 창에서 대기열을 선택합니다.
2. 대기열 페이지에서 대기열 생성을 선택합니다.
3. 대기열 생성 페이지의 대기열 설정에서 다음을 수행하세요.

   a. 이름에는 고유한 이름을 입력합니다.

   b. 제한 시간에는 Amazon GameLift가 중지하기 전에 게임 세션을 배치하도록 하려는 시간을 입력합니다. Amazon GameLift는 요청 시간이 될 때까지 모든 플릿에서 사용 가능한 리소스를 검색합니다.

   c. (선택 사항) 플레이어 지연 시간 정책에는 Amazon GameLift가 정의된 최대 지연 시간 내에서 리소스를 찾는 시간을 입력합니다. 정책을 추가하여 최대 지연 시간을 점진적으로 완화시킵니다. 정책을 더 추가하려면 정책 추가를 선택합니다.
4. 게임 세션 배치 위치에서 대기열에 포함할 위치를 선택합니다. 기본적으로 모든 위치가 포함됩니다. 대기열에 있는 모든 플릿에는 동일한 인증서 구성이 있어야 합니다. 모든 플릿이 대기열을 사용하는 게임 클라이언트와 호환되는 게임 빌드를 실행하고 있어야 합니다.

5. 대상 순서에서 하나 이상의 대상을 대기열에 추가합니다.

   a. 대상 추가를 선택합니다.

   b. 대상이 있는 위치를 선택합니다.

   c. 대상 유형을 선택합니다.

   d. 결과로 얻는 플릿 또는 별칭 이름 목록에서 추가하려는 것을 하나 선택합니다.

   e. 대상이 여러 개 있는 경우 대상 왼쪽에 있는 점 6개 아이콘을 드래그하여 기본 순서를 설정합니다. Amazon GameLift는 새 게임 세션 배치에 사용할 수 있는 리소스를 대상에서 검색할 때 이 순서를 사용합니다.

6. 게임 세션 배치 우선순위의 경우 지연 시간, 비용, 대상 및 위치 값을 추가하고 드래그하여 Amazon GameLift가 대기열에 있는 플릿의 우선 순위를 지정하는 방법을 정의합니다. 플릿의 우선 순위 지정에 대한 자세한 내용은 게임 세션 배치 우선순위 섹션을 참조하세요.

7. 위치 순서에 위치를 추가하고 대기열이 사용해야 하는 우선 순위로 드래그합니다. 위치가 게임 세션 배치의 마지막 우선 순위인 경우 Amazon GameLift는 위치를 타이브레이커로 사용합니다.

8. (선택 사항) 이벤트 알림 설정에서 다음을 수행합니다.

   a. 배치 관련 이벤트 알림을 수신할 SNS 주제를 선택하거나 생성합니다. 이벤트 알림에 대한 자세한 내용은 게임 세션 배치의 이벤트 알림 설정 섹션을 참조하세요.

   b. 이 대기열에서 생성된 이벤트에 추가할 사용자 지정 이벤트 데이터를 추가합니다.

9. (선택 사항) 태그를 추가합니다. 태그 지정에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.

10. 생성을 선택합니다.

AWS CLI

Example 대기열 생성

다음 예제에서는 다음과 같은 구성으로 게임 세션 대기열을 생성합니다.

- 5분 제한 시간
- 2개의 플릿 대상
- `us-east-1`, `us-east-2`, `us-west-2`, `ca-central-1`의 위치만 허용하도록 필터링합니다.
- 비용을 기준으로 대상의 우선 순위를 정한 다음 정의된 순서대로 위치를 지정합니다.

```
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**
>
> 플릿 또는 별칭 ID로 describe-fleet-attributes 또는 describe-alias를 호출하여 플릿 및 별칭 ARN 값을 가져올 수 있습니다.

`create-game-session-queue` 요청이 성공하면 Amazon GameLift가 새 대기열 구성과 함께 GameSessionQueue 객체를 반환합니다. 이제 StartGameSessionPlacement를 사용하여 대기열에 요청을 제출할 수 있습니다.

Example 플레이어 지연 시간 정책이 포함된 대기열 생성

다음 예제에서는 다음과 같은 구성으로 게임 세션 대기열을 생성합니다.

- 10분 제한 시간
- 3개의 플릿 대상
- 플레이어 지연 시간 정책 세트

```
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" \
```

`create-game-session-queue` 요청이 성공하면 Amazon GameLift가 새 대기열 구성과 함께 GameSessionQueue 객체를 반환합니다.

게임 세션 배치의 이벤트 알림 설정

이벤트 알림을 사용하여 개별 배치 요청 상태를 모니터링할 수 있습니다. 배치 활동이 많은 모든 게임에 이벤트 알림을 설정하는 것이 좋습니다.

이벤트 알림을 설정하는 데는 두 가지 옵션이 있습니다.

- Amazon GameLift는 대기열을 사용하여 Amazon Simple Notification Service(SNS) 주제에 이벤트 알림을 게시하도록 합니다.
- 자동으로 게시된 Amazon EventBridge 이벤트 및 이벤트 관리 도구 모음을 사용합니다.

Amazon GameLift에서 내보낸 게임 세션 배치 이벤트 목록에 대해서는 게임 세션 배치 이벤트 섹션을 참조하세요.

SNS 주제 설정

Amazon GameLift가 게임 세션 대기열에서 생성된 모든 이벤트를 주제에 게시하도록 하려면 알림 대상 필드를 주제로 설정합니다.

Amazon GameLift 이벤트 알림에 대한 SNS 주제를 설정하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. SNS 주제 페이지에서 주제 생성을 선택하고 지침에 따라 주제를 생성합니다.
3. 액세스 정책에서 다음을 수행합니다.

    a. 고급 메서드를 선택합니다.

    b. JSON 객체의 굵게 표시된 다음 섹션을 기존 정책에 추가합니다.

```
{
```

```
  "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. (선택 사항) 리소스 정책에 조건을 추가하여 주제에 추가 액세스 제어를 추가합니다.

4. 주제 생성을 선택합니다.
5. SNS 주제를 생성한 후에는 대기열 생성 중에 대기열에 추가하거나 기존 대기열을 편집하여 추가합니다.

SNS 주제에 서버 측 암호화 설정

서버 측 암호화(SSE)를 사용하면 암호화된 주제에서 민감한 데이터를 저장할 수 있습니다. SSE는 AWS Key Management Service(AWS KMS)에서 관리되는 키를 사용하여 Amazon SNS 주제의 메시지 내용을 보호합니다. Amazon SNS를 포함한 서버 측 암호화에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 저장 시 암호화를 참조하세요.

SNS 주제에 서버 측 암호화를 설정하려면 다음 주제를 검토합니다.

- AWS Key Management Service 개발자 가이드의 키 생성
- Amazon Simple Notification Service 개발자 가이드의 주제 SSE 활성화

KMS 키를 생성할 때는 다음 KMS 키 정책을 사용합니다.

```
{
    "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"
        }
    }
```

```
}
```

EventBridge 설정

Amazon GameLift는 모든 게임 세션 배치 이벤트를 EventBridge에 자동으로 게시합니다. EventBridge를 사용하면 이벤트를 처리 대상으로 라우팅하도록 규칙을 설정할 수 있습니다. 예를 들어 게임 세션에 연결하기 전에 수행해야 하는 작업을 처리하는 AWS Lambda 함수에 `PlacementFulfilled` 이벤트를 라우팅하도록 규칙을 설정할 수 있습니다. Eventbridge에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge란 무엇입니까?를 참조하세요.

다음은 Amazon GameLift 대기열에 사용할 수 있는 EventBridge 규칙의 몇 가지 예입니다.

모든 Amazon GameLift 대기열의 이벤트와 일치

```
{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ]
}
```

특정 대기열의 이벤트와 일치

```
{
    "source": [
        "aws.gamelift"
    ],
    "detail-type": [
        "GameLift Queue Placement Event"
    ],
    "resources": [
        "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
    ]
}
```

자습서: 스팟 인스턴스용 게임 세션 대기열 설치

소개

이 자습서에서는 저렴한 스팟 플릿에 배포된 게임의 게임 세션 배치를 설정하는 방법에 대해 설명합니다. 스팟 플릿을 사용하면 플레이어의 게임 서버 가용성을 지속적으로 유지하기 위한 추가 단계가 필요합니다.

수강 대상

이 자습서는 스팟 플릿을 사용하여 사용자 지정 게임 서버 또는 Realtime 서버를 호스팅하려는 게임 개발자를 위한 것입니다.

배울 내용

- 게임 세션 대기열에서 서비스를 제공하는 플레이어 그룹을 정의합니다.
- 게임 세션 대기열의 범위를 지원하는 플릿 인프라를 구축합니다.
- 각 플릿에 별칭을 할당하여 플릿 ID를 추상화합니다.
- 대기열을 만들고, 플릿을 추가하며, Amazon GameLift가 게임 세션을 배치하는 위치의 우선 순위를 지정합니다.
- 플레이어 지연 시간 정책을 추가하여 지연 시간 문제를 최소화하는 데 도움이 됩니다.

필수 조건

게임 세션 배치를 위한 플릿과 대기열을 생성하기 전에, 먼저 다음 작업을 완료해야 합니다.

- Amazon GameLift 작동 방식 섹션을 검토합니다.
- Amazon GameLift에 게임 서버를 통합합니다.
- 게임 서버 빌드를 업로드하거나 Realtime 스크립트를 Amazon GameLift에 업로드합니다.
- 플릿 구성을 계획합니다.

1단계: 대기열 범위 정의

이 자습서에서는 하나의 게임 서버 빌드 변형이 있는 게임의 대기열을 설계합니다. 출시 시점에 아시아 태평양(서울) 및 아시아 태평양(싱가포르)의 두 위치에서 게임을 출시할 예정입니다. 이러한 위치는 서로 가깝기 때문에 플레이어에게 지연 시간은 문제가 되지 않습니다.

이 예제에서는 플레이어 세그먼트가 하나 있는데, 이는 대기열을 하나 생성한다는 뜻입니다. 앞으로 북미에서 게임을 출시하면 북미 플레이어만 이용할 수 있는 두 번째 대기열을 만들 수 있습니다.

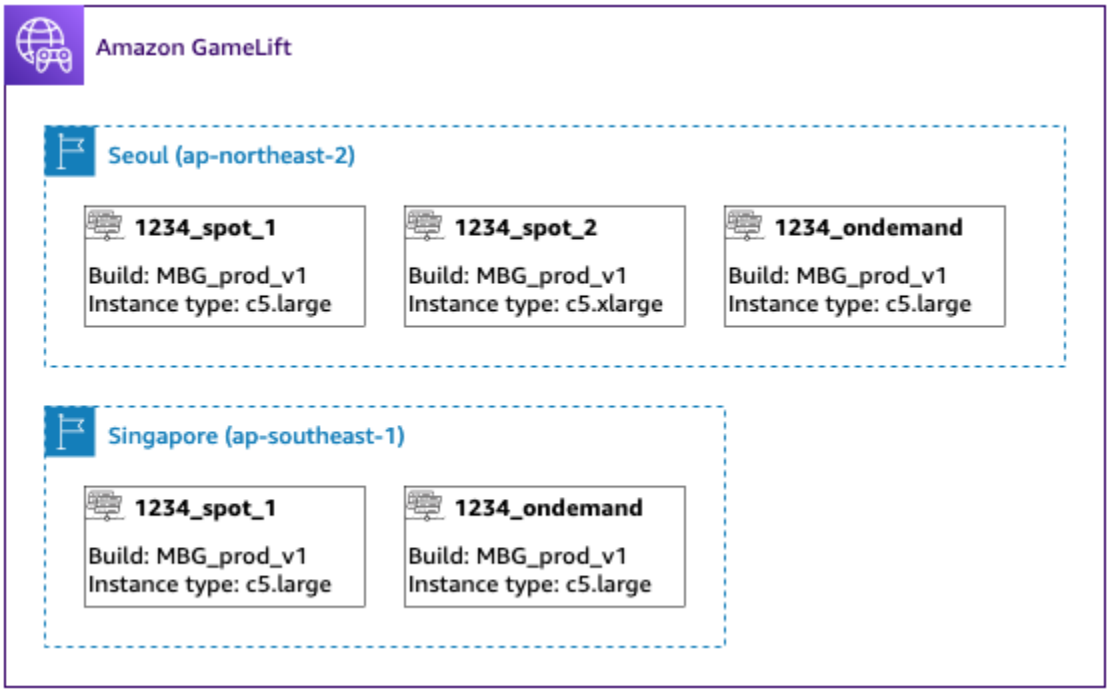
자세한 내용은 대기열의 범위 정의 섹션을 참조하세요.

2단계: 스팟 플릿 인프라 생성

1단계: 대기열 범위 정의에서 정의한 범위에 맞는 게임 서버 빌드 또는 스크립트를 사용하여 위치에 플릿을 생성합니다.

이 자습서에서는 각 위치에 하나 이상의 스팟 플릿과 하나의 온디맨드 플릿이 있는 두 개의 위치 인프라를 생성합니다. 모든 플릿은 동일한 게임 서버 빌드를 배포합니다. 또한 서울 위치에는 플레이어 트래픽이 더 많을 것으로 예상되므로 여기에 스팟 플릿을 더 추가합니다.

다음 다이어그램은 ap-northeast-2(서울) 위치에 3개의 플릿이 있고 ap-southeast-1(싱가포르) 위치에 2개의 플릿이 있는 스팟 플릿 인프라의 예를 보여줍니다. 두 플릿의 모든 인스턴스는 MBG_prod_V1 빌드를 사용하고 있습니다. ap-northeast-2의 플릿에는 다음 플릿 구성을 포함합니다. 인스턴스 유형이 c5.large인 플릿 1234_spot_1, 인스턴스 유형이 c5.xlarge인 플릿 1234_spot_2, 인스턴스 유형이 c5.large인 플릿 1234_ondemand. ap-southeast-1의 플릿에는 다음 플릿 구성을 포함합니다. 인스턴스 유형이 c5.large인 플릿 1234_spot_1, 인스턴스 유형이 c5.large인 플릿 1234_ondemand.

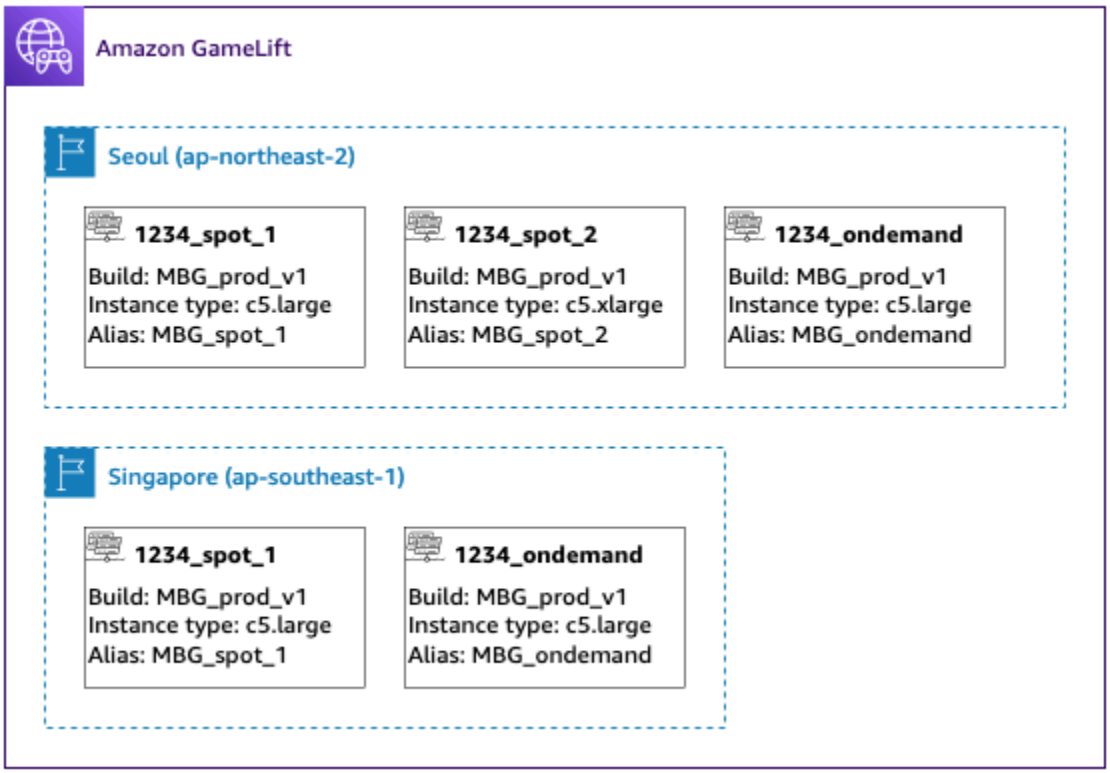


3단계: 각 플릿에 별칭 지정

인프라의 각 플릿에 대해 새 별칭을 생성합니다. 별칭은 플릿 ID를 추상화하여 주기적인 플릿 교체를 효율적으로 만듭니다. 별칭을 만드는 방법에 대한 자세한 내용은 Amazon GameLift 플릿에 별칭 추가 섹션을 참조하세요.

플릿 인프라에는 다섯 개의 플릿이 있으므로 라우팅 전략을 사용하여 다섯 개의 별칭을 생성합니다. 아시아 태평양(서울) 위치에 3개의 별칭이 필요하고 아시아 태평양(싱가포르) 위치에 2개의 별칭이 필요합니다.

다음 다이어그램은 2단계에서 설명한 스팟 플릿 인프라를 각 플릿에 추가된 별칭과 함께 보여줍니다. 플릿 1234_spot_1에는 MBG_spot_1이라는 별칭이 있고, 플릿 1234_spot_2에는 MBG_spot_2라는 별칭이 있으며, 플릿 1234_ondemand에는 MBG_ondemand라는 별칭이 있습니다.



자세한 내용은 다중 위치 대기열 빌드 섹션을 참조하세요.

4단계: 대상이 포함된 대기열 생성

게임 세션 대기열을 생성하고 플릿 대상을 추가합니다. 대기열을 생성에 대한 자세한 내용은 게임 세션 대기열 생성 섹션을 참조하세요.

대기열을 생성할 때:

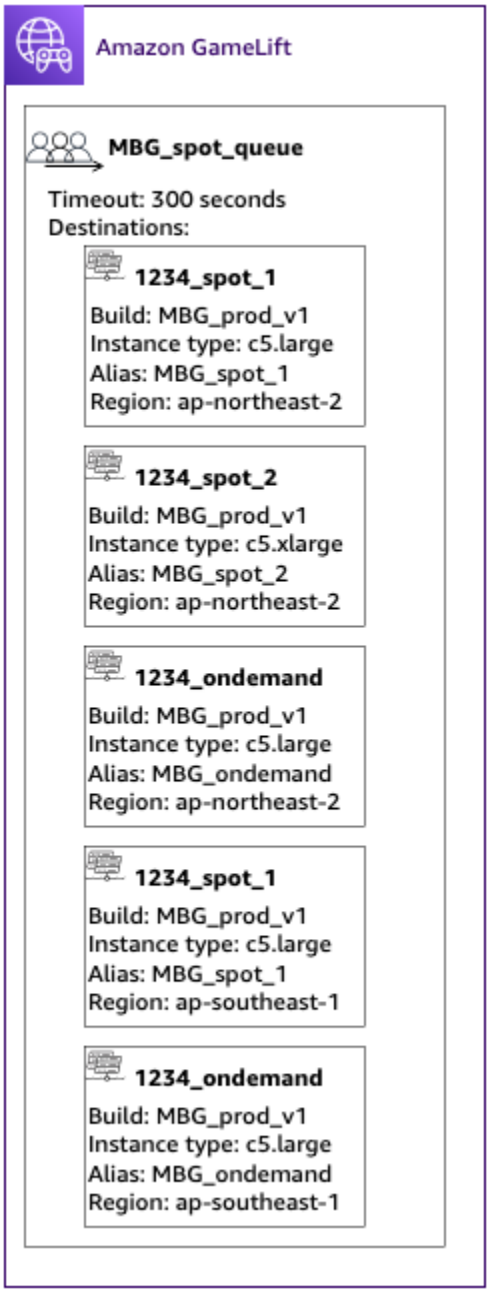
- 기본 타임아웃을 10분으로 설정합니다. 추후에 대기열 제한 시간이 플레이어의 게임 시작 대기 시간에 어떤 영향을 미치는지 테스트할 수 있습니다.
- 지금은 플레이어 지연 시간 정책 섹션을 건너뜁니다. 이 내용은 다음 단계에서 다루도록 하겠습니다.

- 대기열에 있는 플릿의 우선 순위를 지정합니다. 스팟 플릿을 사용할 때는 다음 접근 방식 중 하나를 사용하는 것이 좋습니다.
  - 인프라에서 백업용으로 두 번째 위치에 플릿이 있는 기본 위치를 사용하는 경우 먼저 위치별로 플릿의 우선 순위를 지정한 다음 플릿 유형별로 우선 순위를 지정합니다.
  - 인프라에서 여러 위치를 동일하게 사용하는 경우 플릿 유형별로 플릿의 우선 순위를 지정하여 스팟 플릿을 대기열의 맨 위에 배치합니다.

이 자습서에서는 **MBG_spot_queue** 이름을 사용하여 새 대기열을 만들고 5개 플릿 모두에 별칭을 추가합니다. 그런 다음 첫 번째는 위치별, 두 번째는 플릿 유형별로 배치 우선 순위를 지정합니다.

이 구성을 기반으로 이 대기열은 항상 새 게임 세션을 서울의 스팟 플릿에 배치하도록 시도합니다. 해당 플릿이 가득 차면 대기열은 서울 온디맨드 플릿의 사용 가능한 용량을 백업으로 사용합니다. 3개의 서울 플릿을 모두 사용할 수 없는 경우 Amazon GameLift는 싱가포르 플릿에 게임 세션을 배치합니다.

다음 다이어그램은 제한 시간이 300초인 대기열과 우선 순위가 지정된 대상을 보여줍니다. 목적지는 다음 순서와 같습니다: 1234_spot_1(위치: ap-northeast-2), 1234_spot_2(위치: ap-northeast-2), 1234_ondemand(위치: ap-northeast-2), 1234_spot_1(위치: ap-southeast-1), 1234_ondemand(위치: ap-southeast-1).



5단계: 대기열에 지연 시간 제한 추가

저희 게임에는 게임 세션 배치 요청에 지연 시간 정보가 포함되어 있습니다. 플레이어 그룹을 위한 게임 세션을 생성하는 플레이어 파티 기능도 있습니다. 이상적인 게임 플레이 경험을 제공하는 게임에 참여하기 위해 플레이어가 조금 더 기다릴 수 있도록 할 수 있습니다. 게임 테스트 결과 다음과 같은 관찰 결과가 나왔습니다.

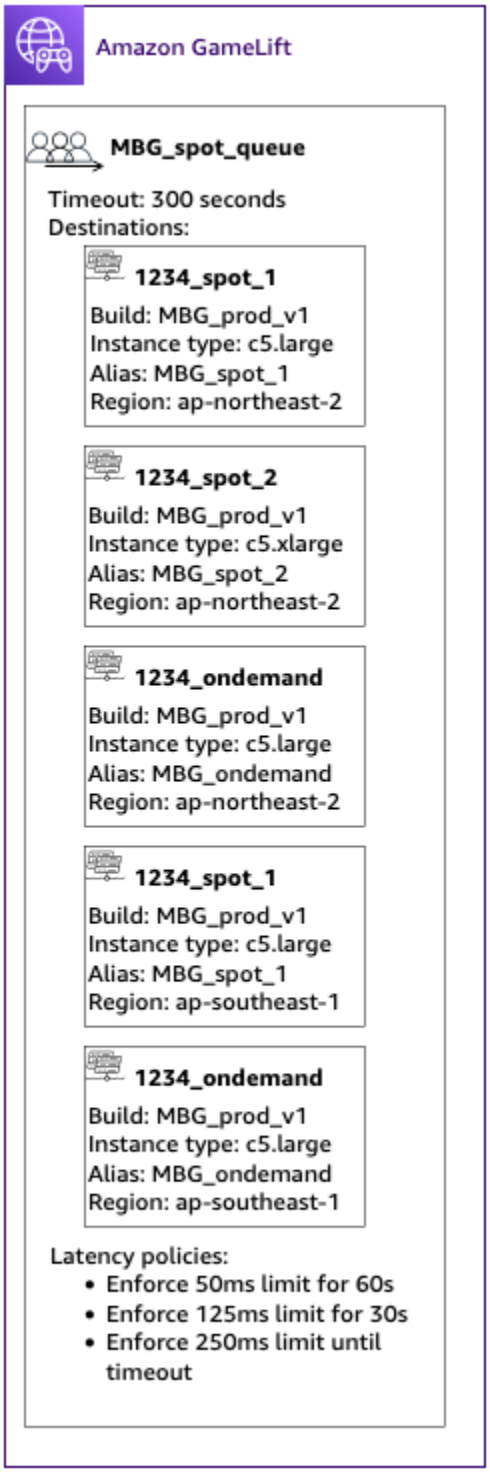
- 50밀리초 미만의 지연 시간이 이상적입니다.

- 지연 시간이 250밀리초를 넘으면 게임을 플레이할 수 없습니다.
- 플레이어는 약 1분 뒤에 대기하기 힘들어집니다.

제한 시간이 300초인 대기열에는 허용 가능한 지연 시간을 제한하는 정책 설명을 추가합니다. 정책 설명에 따라 최대 250밀리초 지연 시간까지 지연 시간 값을 점차 확대할 수 있습니다.

이 정책에 따라 대기열은 처음 1분 동안 지연 시간이 이상적인 위치(50밀리초 미만)를 찾은 다음 제한을 완화합니다. 플레이어 지연 시간이 250밀리초 이상인 경우에는 대기열에서 배치를 진행하지 않습니다.

다음 다이어그램은 플레이어 지연 시간 정책이 추가된 4단계의 대기열을 보여줍니다. 플레이어 지연 시간 정책에는 60초 동안 50ms 제한을 적용하고, 30초 동안 125ms 제한을 적용하며, 제한 시간이 초과될 때까지 250ms 제한을 적용한다고 명시되어 있습니다.



요약

축하합니다! 달성한 성과는 다음과 같습니다.

- 게임 세션 대기열 범위가 플레이어 모집단의 특정 세그먼트에 맞춰져 있습니다.

- 대기열은 스팟 플릿을 효과적으로 사용하며 스팟 중단이 발생해도 복원력이 뛰어납니다.
- 대기열에서는 최고의 플레이어 경험을 위해 플릿을 우선시합니다.
- 대기열에는 플레이어들이 나쁜 게임플레이 경험을 하지 못하도록 지연 시간 제한이 있습니다.

이제 대기열을 사용하여 해당 대기열에서 서비스를 제공하는 플레이어를 위한 게임 세션을 배치할 수 있습니다. 이러한 플레이어에 대한 게임 세션 배치를 요청할 때는 요청에서 이 게임 세션 대기열 이름을 참조합니다. 게임 세션 배치 요청에 대한 자세한 정보는 게임 세션 만들기 또는 Realtime 서버용 게임 클라이언트 통합 섹션을 참조하세요.

다음 단계:

- 대기열을 직접 설계합니다.
- 대기열을 생성합니다.
- 게임 클라이언트와 함께 대기열을 사용합니다.

AWS CloudFormation을 사용하여 리소스 관리

Amazon GameLift 리소스를 관리하는 데 AWS CloudFormation을 사용합니다. AWS CloudFormation에서는 각 리소스를 모델링하는 템플릿을 만든 다음 해당 템플릿을 사용하여 리소스를 생성합니다. 리소스를 업데이트하려면 템플릿을 변경하고 AWS CloudFormation을 사용하여 업데이트를 구현합니다. 리소스를 스택 및 스택 세트라는 논리 그룹으로 구성할 수 있습니다.

AWS CloudFormation을 사용하여 Amazon GameLift 호스팅 리소스를 유지 관리하면 AWS 리소스 세트를 더 효율적인 방법으로 관리할 수 있습니다. 버전 제어를 사용하여 시간 경과에 따른 템플릿 변경을 추적하고 여러 팀원이 수행하는 업데이트를 조정할 수 있습니다. 템플릿을 재사용할 수도 있습니다. 예를 들어 여러 리전에서 게임을 배포할 때 동일한 템플릿을 사용하여 각 리전에서 같은 리소스를 생성할 수 있습니다. 또한 이러한 템플릿을 사용하여 동일한 리소스 세트를 다른 파티션에 배포할 수 있습니다.

AWS CloudFormation에 대한 자세한 내용은 사용 설명서AWS CloudFormation를 참조하세요. Amazon GameLift 리소스에 대한 템플릿 정보를 보려면 Amazon GameLift 리소스 유형 참조를 참조하세요.

모범 사례

AWS CloudFormation 사용에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 모범 사례를 참조하세요. 또한 이러한 모범 사례는 Amazon GameLift와 특별한 관련성이 있습니다.

- AWS CloudFormation을 사용하여 리소스를 일관적으로 관리합니다. AWS CloudFormation 외부에서 리소스를 변경하면 리소스가 리소스 템플릿과 동기화되지 않습니다.
- AWS CloudFormation 스택 및 스택 세트를 사용하여 여러 리소스를 효율적으로 관리합니다.
  - 스택을 사용하여 연결된 리소스 그룹을 관리할 수 있습니다. 예를 들어 빌드가 포함된 스택, 빌드를 참조하는 플릿 및 플릿을 참조하는 별칭이 있습니다. 템플릿을 업데이트하여 빌드를 교체하면 AWS CloudFormation은 빌드에 연결된 플릿도 교체합니다. 그런 다음 AWS CloudFormation은 새 플릿을 가리 키도록 기존 별칭을 업데이트합니다. 자세한 내용을 알아보려면 AWS CloudFormation 사용 설명서의 스택 작업을 참조하세요.
  - 여러 리전 또는 AWS 계정에서 동일한 스택을 배포하는 경우 AWS CloudFormation 스택 세트를 사용합니다. 자세한 내용을 알아보려면 AWS CloudFormation 사용 설명서의 스택 세트 작업을 참조하세요.
- 스팟 인스턴스를 사용하는 경우 온디맨드 플릿을 백업으로 포함시킵니다. 각 리전에 스팟 인스턴스가 있는 플릿 하나 및 온디맨드 인스턴스가 있는 플릿 하나라는 두 개의 플릿이 있는 템플릿을 설정하는 것이 좋습니다.
- 여러 위치에서 리소스를 관리하는 경우 위치별 리소스와 글로벌 리소스를 별도의 스택으로 그룹화합니다.
- 글로벌 리소스를 사용하는 서비스와 가까운 곳에 글로벌 리소스를 배치합니다. 대기열 및 매치메이킹 구성과 같은 리소스는 특정 소스에서 대량의 요청을 수신하는 경향이 있습니다. 리소스를 이러한 요청의 소스와 가까운 곳에 배치하면 요청 이동 시간을 최소화하고 전반적인 성능을 향상할 수 있습니다.
- 매치메이킹 구성을 이 구성이 사용하는 게임 세션 대기열과 동일한 리전에 배치합니다.
- 스택의 각 플릿에 대해 별도의 별칭을 만듭니다.

AWS CloudFormation 스택 사용

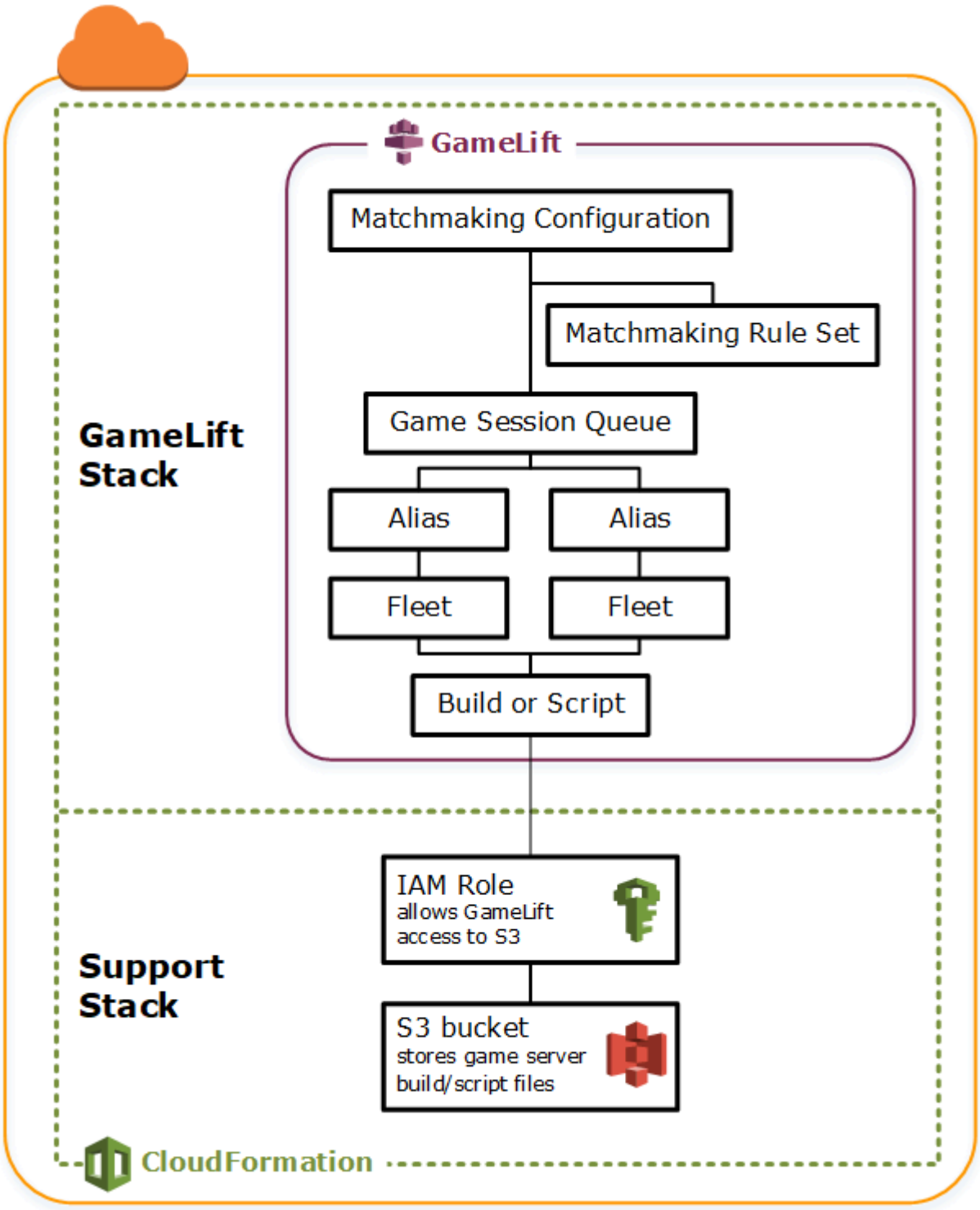
다음은 Amazon GameLift 리소스에 대한 AWS CloudFormation 스택을 설정할 때 사용하도록 권장되는 구조입니다. 최적의 스택 구조는 게임을 한 리전에만 배포하는지 또는 여러 리전에 배포하는지에 따라 다릅니다.

단일 위치용 스택

Amazon GameLift 리소스를 단일 위치에서 관리하려면 2스택 구조를 사용하는 것이 좋습니다.

- 지원 스택 - 이 스택에는 Amazon GameLift 리소스가 종속되는 리소스가 포함되어 있습니다. 최소한 이 스택에는 사용자 지정 게임 서버 또는 Realtime 스크립트 파일을 저장하는 S3 버킷이 포함되어야 합니다. 이 스택에는 Amazon GameLift 빌드 또는 스크립트 리소스를 생성할 때 S3 버킷에서 파일을 검색할 수 있는 Amazon GameLift 권한을 부여하는 IAM 역할도 포함되어야 합니다. 또한 이 스택에는 DynamoDB 테이블, Amazon Redshift 클러스터 및 Lambda 함수와 같이 게임에 사용되는 기타 AWS 리소스도 포함될 수 있습니다.
- Amazon GameLift 스택 - 이 스택에는 빌드 또는 스크립트, 플릿 세트, 별칭, 게임 세션 대기열을 포함한 모든 Amazon GameLift 리소스가 포함되어 있습니다. AWS CloudFormation은 S3 버킷 위치에 저장된 파일을 사용하여 빌드 또는 스크립트 리소스를 생성하고 빌드 또는 스크립트를 하나 이상의 플릿 리소스에 배포합니다. 각 플릿에는 해당 별칭이 있어야 합니다. 게임 세션 대기열은 플릿 별칭 중 일부 또는 전부를 참조합니다. FlexMatch를 매치메이킹에 사용하는 경우 이 스택에는 매치메이킹 구성 및 규칙 세트도 포함됩니다.

아래 다이어그램은 단일 AWS 리전에 리소스를 배포하기 위한 2스택 구조를 보여 줍니다.



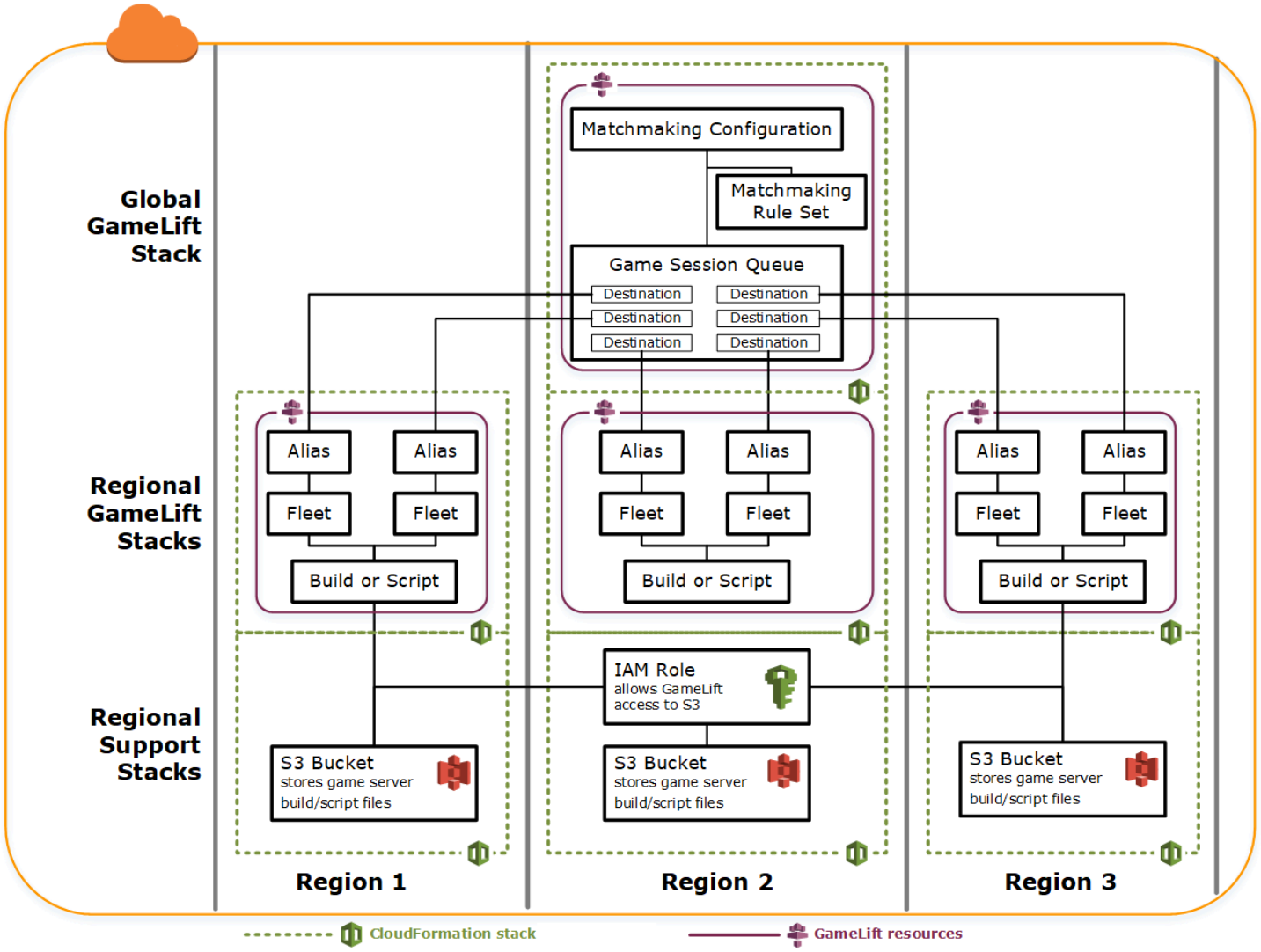
여러 리전에 대한 스택

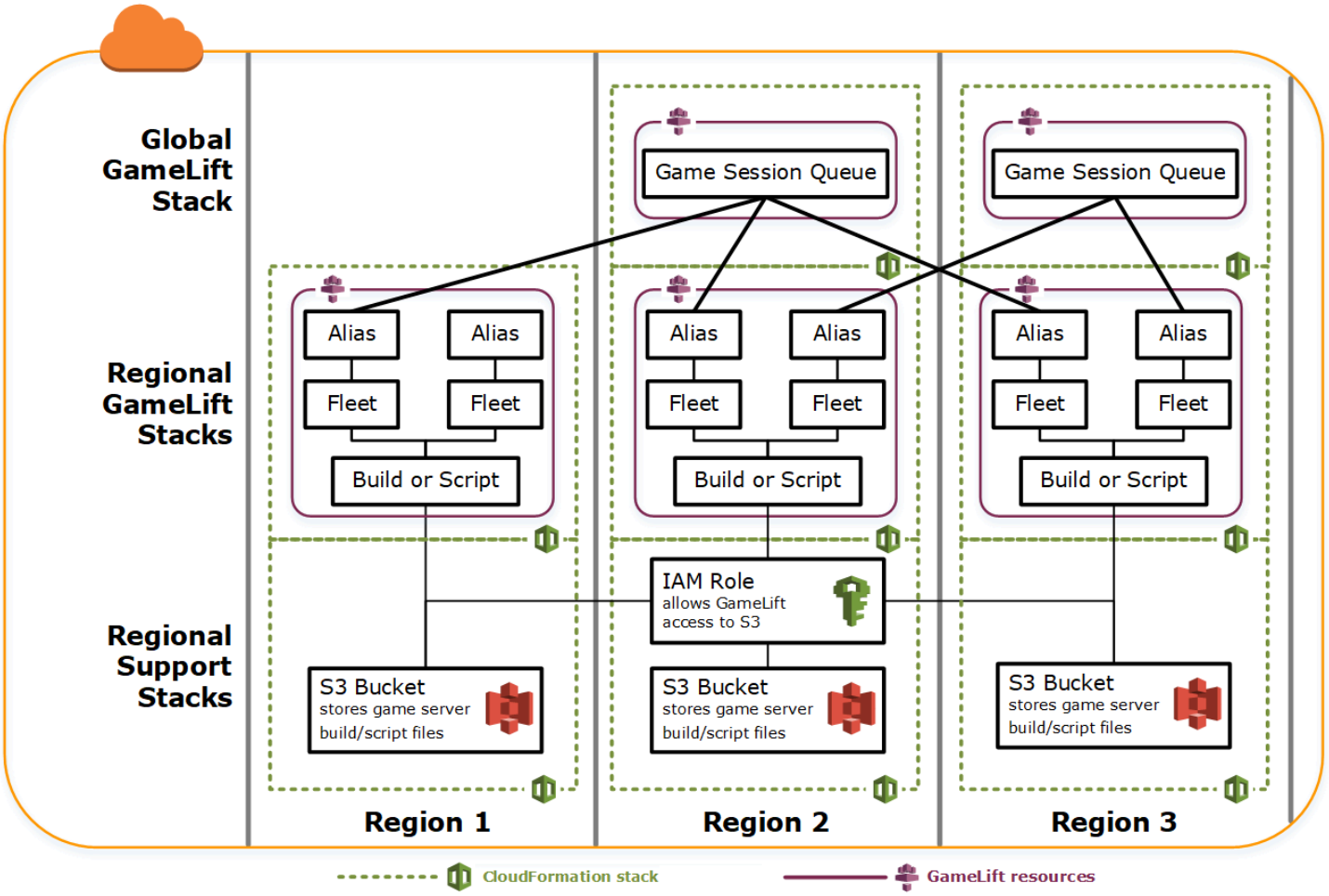
두 개 이상의 리전에 게임을 배포할 때는 리소스가 리전 간에 상호 작용할 수 있는 방식을 명심해야 합니다. Amazon GameLift 플릿과 같은 일부 리소스는 동일한 리전의 기타 리소스만 참조할 수 있습니다.

Amazon GameLift 대기열과 같은 기타 리소스는 리전과 무관합니다. 여러 리전에서 Amazon GameLift 리소스를 관리하려면 다음 구조를 사용하는 것이 좋습니다.

- 리전별 지원 스택 - 이 스택에는 Amazon GameLift 리소스가 종속되는 리소스가 포함되어 있습니다. 이 스택에는 사용자 지정 게임 서버 또는 Realtime 스크립트 파일을 저장하는 S3 버킷이 포함되어야 합니다. DynamoDB 테이블, Amazon Redshift 클러스터 및 Lambda 함수와 같이 게임에 사용되는 기타 AWS 리소스도 포함될 수 있습니다. 이러한 리소스는 대부분 리전별로 다르므로 모든 리전에서 생성해야 합니다. Amazon GameLift에는 이러한 지원 리소스에 대한 액세스를 허용하는 IAM 역할도 필요합니다. IAM 역할은 리전과 무관하므로 어떠한 리전에든 배치되고 기타 모든 지원 스택에서 참조되는 역할 리소스 하나만 있으면 됩니다.
- 리전별 Amazon GameLift 스택 - 이 스택에는 빌드 또는 스크립트, 플릿 세트 및 별칭을 포함하여 게임이 배포되는 각 리전에 존재해야 하는 Amazon GameLift 리소스가 포함되어 있습니다. AWS CloudFormation는 S3 버킷 위치에 파일이 있는 빌드 또는 스크립트 리소스를 생성하고 빌드 또는 스크립트를 하나 이상의 플릿 리소스에 배포합니다. 각 플릿에는 해당 별칭이 있어야 합니다. 게임 세션 대기열은 플릿 별칭 중 일부 또는 전부를 참조합니다. 이러한 유형의 스택을 설명하도록 하나의 템플릿을 유지 관리하고 이 템플릿을 사용하여 모든 리전에서 동일한 리소스 세트를 생성할 수 있습니다.
- 글로벌 Amazon GameLift 스택 - 이 스택에는 게임 세션 대기열과 매치메이킹 리소스가 포함되어 있습니다. 이러한 리소스는 어떠한 리전에든 위치할 수 있으며 일반적으로 동일한 리전에 배치됩니다. 대기열은 어떠한 리전에든 위치하는 플릿 또는 별칭을 참조할 수 있습니다. 추가 대기열을 다른 리전에 배치하려면 추가 글로벌 스택을 생성합니다.

아래 다이어그램은 여러 AWS 리전에서 리소스를 배포하기 위한 다중 스택 구조를 보여 줍니다. 첫 번째 다이어그램은 단일 게임 세션 대기열의 구조를 보여 줍니다. 두 번째 다이어그램은 여러 개의 대기열이 있는 구조를 보여 줍니다.


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



빌드 업데이트

Amazon GameLift 빌드는 빌드와 플릿 간의 관계와 마찬가지로 변경할 수 없습니다. 따라서 새로운 게임 빌드 파일 세트를 사용하도록 호스팅 리소스를 업데이트할 때 다음과 같은 작업이 필요합니다.

- 새 파일 세트를 사용하여 새 빌드를 생성합니다(교체).
- 새 게임 빌드를 배포할 새 플릿 세트를 생성합니다(교체).
- 새 플릿을 가리키도록 별칭을 리디렉션합니다(중단 없이 업데이트).

자세한 정보에 대해서는 AWS CloudFormation 사용 설명서의 스택 리소스의 업데이트 동작을 참조하세요.

자동으로 빌드 업데이트 배포

관련 빌드, 플릿 및 별칭 리소스가 포함된 스택을 업데이트할 때 기본 AWS CloudFormation 동작은 이러한 단계를 순차적으로 자동으로 수행하는 것입니다. 먼저 새 빌드 파일을 새 S3 위치에 업로드하여

이 업데이트를 트리거합니다. 그런 다음 새 S3 위치를 가리키도록 AWS CloudFormation 빌드 템플릿을 수정합니다. 스택을 새 S3 위치로 업데이트하면 다음 AWS CloudFormation 시퀀스가 트리거됩니다.

1. S3에서 새 파일을 검색하고, 파일의 유효성을 검사한 다음, 새 Amazon GameLift 빌드를 생성합니다.
2. 플릿 템플릿에서 빌드 참조를 업데이트합니다. 그러면 새 플릿 생성이 트리거됩니다.
3. 새 플릿이 활성화된 후 별칭에서 플릿 참조를 업데이트합니다. 그러면 별칭이 새 플릿을 대상으로 하도록 업데이트가 트리거됩니다.
4. 이전 플릿을 삭제합니다.
5. 이전 빌드를 삭제합니다.

게임 세션 대기열이 플릿 별칭을 사용하는 경우 별칭이 업데이트되는 즉시 플레이어 트래픽은 새 플릿으로 자동으로 전환됩니다. 게임 세션이 종료되면 기존 플릿에서 플레이어가 점차적으로 드레이닝됩니다. Auto-Scaling은 플레이어 트래픽의 변동에 따라 각 플릿 세트에서 인스턴스를 추가하고 제거하는 작업을 처리합니다. 또는 원하는 초기 인스턴스 수를 지정하여 전환을 위해 빠르게 확장하고 나중에 Auto-Scaling을 활성화할 수 있습니다.

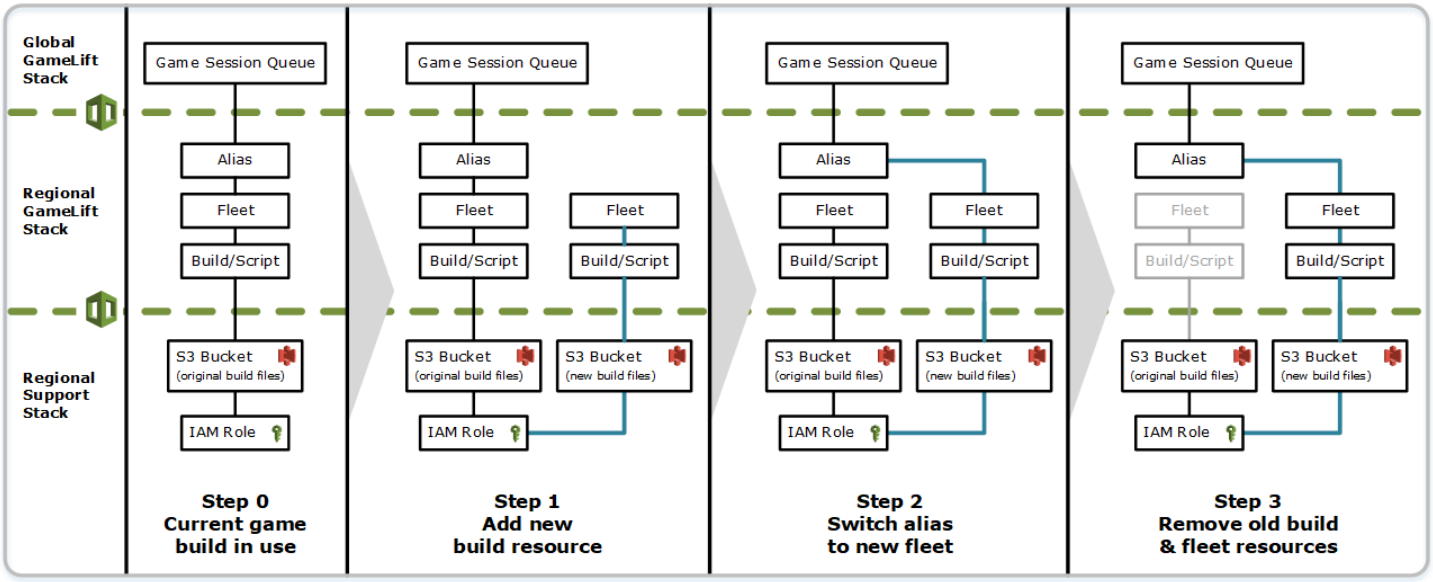
리소스를 삭제하는 대신 AWS CloudFormation에서 리소스를 보존할 수도 있습니다. 자세한 내용은 AWS CloudFormation API 참조의 RetainResources을 참조하세요.

수동으로 빌드 업데이트 배포

새 플릿이 플레이어를 위해 실행되는 시간을 더 세밀하게 제어하려는 경우 몇 가지 옵션을 선택할 수 있습니다. Amazon GameLift 콘솔 또는 CLI를 사용하여 수동으로 별칭을 관리하도록 선택할 수 있습니다. 또는 빌드와 플릿을 교체하도록 빌드 템플릿을 업데이트하는 대신, 두 번째 빌드 및 플릿 정의 세트를 템플릿에 추가할 수 있습니다. 템플릿을 업데이트하면 AWS CloudFormation에서 두 번째 빌드 리소스와 해당 플릿이 생성됩니다. 기존 리소스는 교체되지 않으므로 삭제되지 않으며 별칭은 여전히 원래 플릿을 가리킵니다.

이 접근 방식의 주요 장점은 유연성을 제공한다는 것입니다. 빌드의 새 버전에 대해 별도의 리소스를 생성하고, 새 리소스를 테스트한 다음, 새 플릿을 플레이어에게 실행하는 시점을 제어할 수 있습니다. 잠재적인 단점은 짧은 기간 동안 각 리전에서 두 배 정도 많은 리소스가 필요하다는 것입니다.

다음 다이어그램에서 이 프로세스를 보여 줍니다.



롤백 작동 방식

리소스 업데이트를 실행할 때 한 단계가 성공적으로 완료되지 않으면 AWS CloudFormation에서 자동으로 롤백이 시작됩니다. 이 프로세스는 각 단계를 순차적으로 반전하여 새로 생성된 리소스를 삭제합니다.

롤백을 수동으로 트리거해야하는 경우 빌드 템플릿의 S3 위치 키를 다시 원래 위치로 변경하고 스택을 업데이트합니다. 새 Amazon GameLift 빌드와 플릿이 생성되고, 플릿이 활성화된 후 별칭이 새 플릿으로 전환됩니다. 별칭을 따로 관리하는 경우 새 플릿을 가리키도록 별칭을 전환해야 합니다.

실패하거나 중단되는 롤백을 처리하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 업데이트 롤백 계속하기를 참조하세요.

Amazon GameLift용 VPC 피어링

이 주제에서는 Amazon GameLift 호스팅 게임 서버와 다른 비 Amazon GameLift 리소스 사이의 VPC 피어링 연결을 설정하는 방법을 안내합니다. Amazon Virtual Private Cloud(VPC) 피어링 연결을 사용하여 게임 서버에서 웹 서비스, 리포지토리 등과 같은 다른 AWS 리소스와 직접 비공개로 통신할 수 있습니다. AWS에서 실행되고 액세스 권한이 있는 AWS 계정으로 관리되는 모든 리소스와의 VPC 피어링을 설정할 수 있습니다.

> **Note**
>
> VPC 피어링은 고급 기능입니다. 게임 서버에서 다른 AWS 리소스와 직접 비공개로 통신할 수 있도록 하는 기본 옵션에 대해 알아보려면 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

Amazon VPC 및 VPC 피어링에 대해 잘 알고 있는 경우 Amazon GameLift 게임 서버와의 피어링 설정은 약간 다릅니다. 게임 서버(Amazon GameLift 서비스로 제어됨)를 포함하는 VPC에 액세스할 권한이 없으므로 VPC 피어링을 직접 요청할 수 없습니다. 그 대신 비 Amazon GameLift 리소스를 포함하는 VPC를 사전 승인하여 Amazon GameLift 서비스의 피어링 요청을 수락합니다. 그런 다음 Amazon GameLift를 트리거하여 방금 승인한 VPC 피어링을 요청합니다. Amazon GameLift는 피어링 연결 생성, 라우팅 테이블 설정 및 연결 구성 작업을 처리합니다.

기존 플릿용 VPC 피어링을 설정하려면

1. AWS 계정 ID 및 자격 증명을 가져옵니다.

   다음 AWS 계정에 대한 ID 및 로그인 자격 증명이 필요합니다. AWS Management Console에 로그인한 후 계정 설정을 확인하여 AWS 계정 ID를 찾을 수 있습니다. 자격 증명을 가져오려면 IAM 콘솔로 이동합니다.

   - Amazon GameLift 게임 서버를 관리하는 데 사용되는 AWS 계정입니다.
   - 비 Amazon GameLift 리소스를 관리하는 데 사용되는 AWS 계정입니다.

   Amazon GameLift 리소스와 비 Amazon GameLift 리소스에 대해 동일한 계정을 사용할 경우 한 계정에 대해서만 ID와 자격 증명이 필요합니다.

2. 각 VPC에 대한 ID를 가져옵니다.

   피어링할 두 VPC에 대해 다음 정보를 가져옵니다.

   - Amazon GameLift 게임 서버용 VPC - Amazon GameLift 플릿 ID입니다. EC2 인스턴스 플릿의 Amazon GameLift에 게임 서버가 배포됩니다. 플릿은 해당 VPC에 자동으로 배치되며 Amazon GameLift 서비스에 의해 관리됩니다. VPC에 직접 액세스할 권한이 없어 플릿 ID로 확인됩니다.
   - 비 Amazon GameLift AWS 리소스용 VPC - AWS에서 실행되고 액세스 권한이 있는 AWS 계정에 의해 관리되는 모든 리소스와의 VPC 피어링을 설정할 수 있습니다. 이러한 리소스에 대한 VPC를 아직 생성하지 않은 경우에는 Amazon VPC 시작하기를 참조하세요. VPC를 생성한 경

우 Amazon VPC용 AWS Management Console에 로그인한 후 VPC를 확인하여 VPC ID를 찾을 수 있습니다.

> **Note**
>
> 피어링을 설정할 때는 두 VPC가 동일한 리전에 있어야 합니다. Amazon GameLift 플릿 게임 서버용 VPC는 플릿과 동일한 리전에 있습니다.

3. VPC 피어링을 승인합니다.

   이 단계에서는 비 Amazon GameLift 리소스에 대해 VPC가 있는 게임 서버와 함께 VPC를 피어링하기 위해 Amazon GameLift에서 향후 요청을 사전 승인합니다. 이 작업은 VPC에 대한 보안 그룹을 업데이트합니다.

   VPC 피어링을 승인하려면 Amazon GameLift Service API CreateVpcPeeringAuthorization()을 호출하거나 AWS CLI 명령 `create-vpc-peering-authorization`을 사용합니다. 비 Amazon GameLift 리소스를 관리하는 계정을 사용하여 호출합니다. 다음 정보를 식별합니다.

   - 피어 VPC ID - 비 Amazon GameLift 리소스를 포함하는 VPC를 위한 것입니다.
   - Amazon GameLift AWS 계정 ID - Amazon GameLift 플릿을 관리하는 데 사용하는 계정입니다.

   VPC 피어링을 승인한 경우 별도로 취소하지 않는 한 승인은 24시간 동안 유효합니다. 다음 작업을 사용하여 VPC 피어링 승인을 관리할 수 있습니다.

   - DescribeVpcPeeringAuthorizations()(AWS CLI `describe-vpc-peering-authorizations`)
   - DeleteVpcPeeringAuthorization()(AWS CLI `delete-vpc-peering-authorization`)

4. 피어링 연결을 요청합니다.

   유효한 승인을 통해 Amazon GameLift가 피어링 연결을 설정하도록 요청할 수 있습니다.

   VPC 피어링을 요청하려면 Amazon GameLift Service API CreateVpcPeeringConnection()을 호출하거나 AWS CLI 명령 `create-vpc-peering-connection`을 사용합니다. Amazon GameLift 게임 서버를 관리하는 계정을 사용하여 호출합니다. 다음 정보를 사용하여 피어링할 두 VPC를 식별합니다.

   - 피어 VPC ID 및 AWS 계정 ID - 이것은 비 Amazon GameLift 리소스용 VPC와 해당 VPC를 관리하는 데 사용하는 계정입니다. VPC ID는 유효한 피어링 승인의 ID와 일치해야 합니다.

- 플릿 ID - 이것으로 Amazon GameLift 게임 서버용 VPC를 확인합니다.

5. 피어링 연결 상태를 추적합니다.

   VPC 피어링 연결 요청은 비동기 작업입니다. 피어링 요청의 상태를 추적하여 성공 또는 실패 사례를 처리하려면 다음 옵션 중 하나를 사용합니다.

   - `DescribeVpcPeeringConnections()`을 사용하여 지속적으로 폴링합니다. 이 작업에서는 요청 상태를 포함하여 VPC 피어링 연결 기록을 검색합니다. 피어링 연결이 생성된 경우 VPC에 할당되는 프라이빗 IP 주소의 CIDR 블록이 연결 기록에 포함되어 있습니다.
   - DescribeFleetEvents()를 사용하여 성공 및 실패 이벤트를 비롯한 VPC 피어링 연결과 연관된 플릿 이벤트를 처리합니다.

피어링 연결이 설정되면 다음 작업을 사용하여 관리할 수 있습니다.

- DescribeVpcPeeringConnections()(AWS CLI `describe-vpc-peering-connections`)
- DeleteVpcPeeringConnection()(AWS CLI `delete-vpc-peering-connection`)

새 플릿으로 VPC 피어링을 설정하려면

새 Amazon GameLift 플릿을 생성하는 동시에 VPC 피어링 연결을 요청할 수 있습니다.

1. AWS 계정 ID 및 자격 증명을 가져옵니다.

   다음 두 개의 AWS 계정에 대한 ID 및 로그인 자격 증명이 필요합니다. AWS Management Console에 로그인한 후 계정 설정을 확인하여 AWS 계정 ID를 찾을 수 있습니다. 자격 증명을 가져오려면 IAM 콘솔로 이동합니다.

   - Amazon GameLift 게임 서버를 관리하는 데 사용되는 AWS 계정입니다.
   - 비 Amazon GameLift 리소스를 관리하는 데 사용되는 AWS 계정입니다.

   Amazon GameLift 리소스와 비 Amazon GameLift 리소스에 대해 동일한 계정을 사용할 경우 한 계정에 대해서만 ID와 자격 증명이 필요합니다.

2. 비 Amazon GameLift AWS 리소스용 VPC ID를 받습니다.

   이러한 리소스에 대한 VPC를 아직 생성하지 않았다면 지금 생성합니다(Amazon VPC 시작하기 참조). 새 플릿을 생성할 위치와 동일한 리전에 새 VPC를 생성해야 합니다. 비 Amazon GameLift

리소스가 Amazon GameLift에서 사용하는 계정과 다른 AWS 계정 또는 사용자/사용자 그룹에서 관리되는 경우, 다음 단계에서 승인을 요청할 때 이러한 계정 자격 증명을 사용해야 합니다.

VPC를 생성한 후에는 VPC를 확인하여 Amazon VPC 콘솔에서 VPC ID를 찾을 수 있습니다.

3. 비 Amazon GameLift 리소스와의 VPC 피어링을 승인합니다.

   Amazon GameLift가 새 플릿과 해당 VPC를 생성할 때 비 Amazon GameLift 리소스용 VPC와 피어링하기 위한 요청도 전송합니다. 이 요청을 사전에 승인해야 합니다. 이 단계에서는 VPC에 대한 보안 그룹을 업데이트합니다.

   비 Amazon GameLift 리소스를 관리하는 계정 자격 증명을 사용하여 Amazon GameLift Service API CreateVpcPeeringAuthorization()을 호출하거나 AWS CLI 명령 `create-vpc-peering-authorization`을 사용합니다. 다음 정보를 식별합니다.

   - 피어 VPC ID - 비 Amazon GameLift 리소스를 포함하는 VPC의 ID입니다.
   - Amazon GameLift AWS 계정 ID - Amazon GameLift 플릿을 관리하는 데 사용하는 계정 ID입니다.

   VPC 피어링을 승인한 경우 별도로 취소하지 않는 한 승인은 24시간 동안 유효합니다. 다음 작업을 사용하여 VPC 피어링 승인을 관리할 수 있습니다.

   - DescribeVpcPeeringAuthorizations()(AWS CLI `describe-vpc-peering-authorizations`)
   - DeleteVpcPeeringAuthorization()(AWS CLI `delete-vpc-peering-authorization`)

4. AWS CLI를 사용하는 새로운 플릿을 생성하기에 나와 있는 지침을 따릅니다. 다음 추가 파라미터를 포함합니다.

   - peer-vpc-aws-account-id - 비 Amazon GameLift 리소스와 함께 VPC를 관리할 때 사용하는 계정용 ID입니다.
   - peer-vpc-id - 비 GameLift 계정이 있는 VPC ID입니다.

VPC 피어링 파라미터로 create-fleet을 성공적으로 호출하여 새로운 플릿 및 VPC 피어링 요청을 생성합니다. 플릿의 상태가 New로 설정되고 플릿 활성화 프로세스가 시작됩니다. 피어링 연결 요청의 상태가 initiating-request로 설정됩니다. describe-vpc-peering-connections를 호출하여 피어링 요청이 성공했는지 여부를 추적할 수 있습니다.

새 플릿 및 VPC 피어링 연결을 요청할 경우 두 작업 모두 성공하거나 실패합니다. 생성 프로세스 중에 플릿이 실패할 경우 VPC 피어링이 연결되지 않습니다. 마찬가지로 어떠한 이유로 VPC 피어링 연결이 실패할 경우 새 플릿의 상태가 Activating에서 Active로 전환되지 않습니다.

> **Note**
>
> 새로운 VPC 피어링 연결은 플릿이 활성화될 준비가 될 때까지 완료되지 않습니다. 즉, 해당 연결을 사용할 수 없으며 게임 서버 빌드 설치 프로세스 중에도 사용할 수 없습니다.

다음 예제에서는 새 플릿을 생성하고 미리 설정된 VPC와 새 플릿의 VPC 간에 피어링 연결을 생성합니다. 비 Amazon GameLift AWS 계정 ID와 VPC ID를 조합하여 미리 구출된 VPC를 고유하게 식별합니다.

```
$ 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"
```

복사 가능 버전:

```
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"
```

VPC 피어링 문제 해결

Amazon GameLift 게임 서버에 대한 VPC 피어링 연결을 설정하는 데 문제가 있는 경우, 다음과 같은 일반적인 근본 원인을 확인합니다.

- 요청된 연결에 대한 승인을 찾을 수 없습니다.
  - 비 Amazon GameLift VPC에 대한 VPC 승인 상태를 확인합니다. 존재하지 않거나 만료되었을 수 있습니다.
  - 피어링하려는 두 VPC의 리전을 확인합니다. 동일한 리전에 있지 않으면 피어링할 수 없습니다.
- 두 VPC의 CIDR 블록(잘못된 VPC 피어링 연결 구성 참조)이 중첩되어 있습니다. 피어링된 VPC에 할당되는 IPv4 CIDR 블록은 중첩될 수 없습니다. Amazon GameLift 플릿에 대한 VPC의 CIDR 블록은 자동으로 할당되며 변경할 수 없으므로 비 Amazon GameLift 리소스에 대해 VPC의 CIDR 블록을 변경해야 합니다. 이 문제를 해결하려면:
  - `DescribeVpcPeeringConnections()`을 호출하여 해당 Amazon GameLift 플릿에 대한 이 CIDR 블록을 조회합니다.
  - Amazon VPC 콘솔로 이동하여 비 Amazon GameLift 리소스에 대한 VPC를 찾은 다음 중첩되지 않도록 CIDR 블록을 변경합니다.
- 새 플릿이 활성화되지 않았습니다(새 플릿으로 VPC 피어링 요청 시). 새 플릿이 활성 상태로 진행되지 않으면 피어링할 VPC가 없으므로 피어링 연결이 실패합니다.

콘솔에서 게임 데이터 보기

관리형 Amazon GameLift 서비스는 활성 게임의 데이터를 지속적으로 수집하여 플레이어 행동 및 성과를 이해하는 데 도움을 줍니다. Amazon GameLift 콘솔을 사용하여 빌드, 플릿, 게임 세션 및 플레이어 세션별로 이러한 정보를 확인, 관리 및 분석할 수 있습니다.

주제

현재 Amazon GameLift 상태 보기

Amazon GameLift 대시보드에서는 다음을 볼 수 있습니다.

- 준비 완료, 초기화됨, 실패 상태의 빌드 수. 현재 리전의 빌드에 대한 세부 정보를 보려면 빌드 보기를 선택하세요.
- 모든 상태의 플릿 수. 현재 리전의 플릿에 대한 세부 정보를 보려면 플릿 보기를 선택하세요.
- 현재 리소스.
- 새로운 기능 및 서비스 공지.

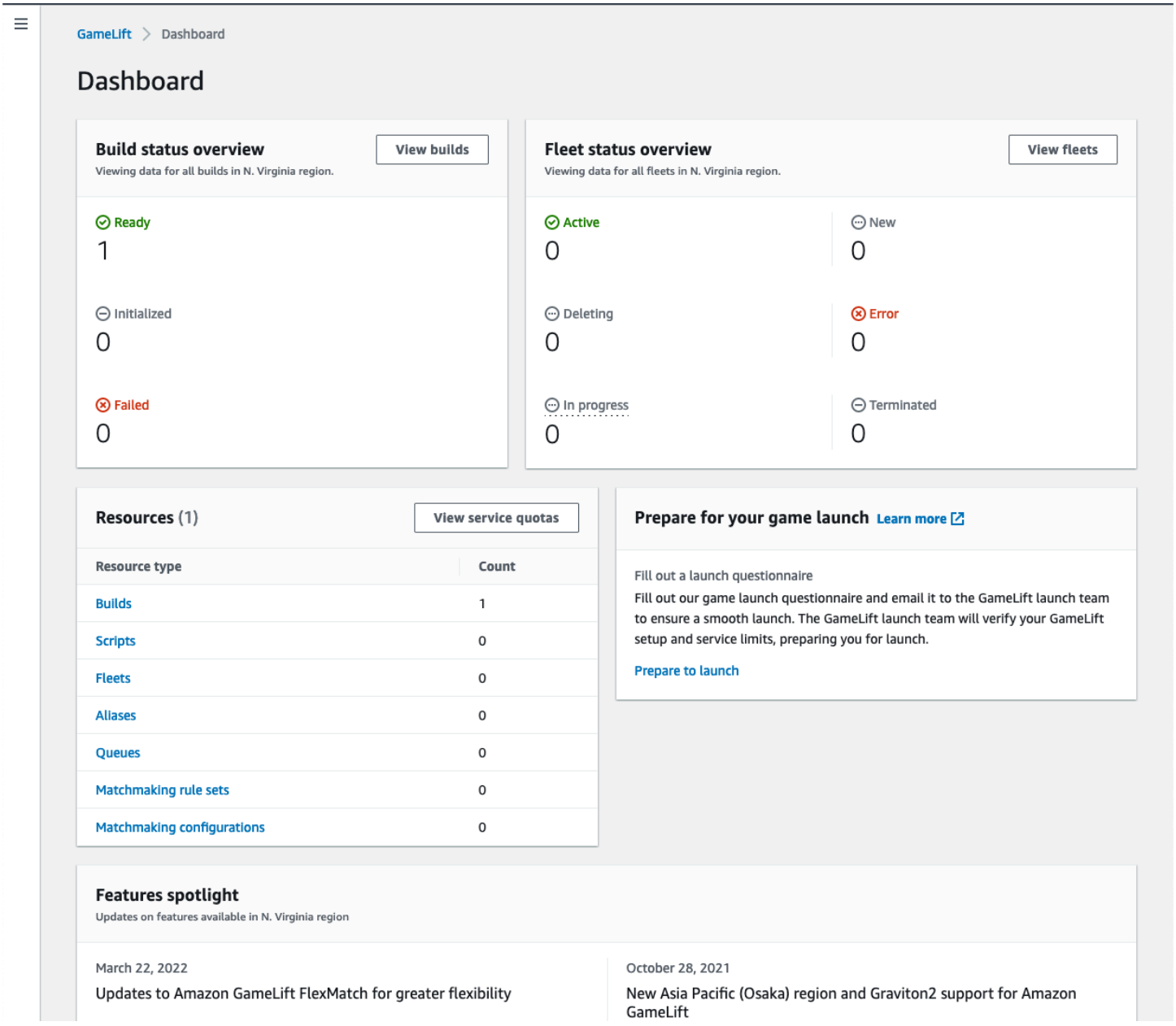
Amazon GameLift 대시보드를 열려면

- Amazon GameLift 콘솔의 탐색 창에서 대시보드를 선택합니다.

대시보드에서 다음을 수행할 수 있습니다.

- 출시 준비를 선택하고 해당하는 출시 설문지를 작성하여 게임 출시를 준비합니다.

- 서비스 할당량 보기를 선택하여 출시를 준비하거나 출시에 대한 응답으로 Service Quotas 증가를 요청합니다.
- 기능 스포트라이트에서 링크를 선택하여 블로그 게시물과 새 기능에 대한 자세한 정보를 볼 수 있습니다.



빌드 보기

Amazon GameLift 콘솔의 빌드 페이지에서 Amazon GameLift에 업로드한 모든 게임 서버 빌드에 대한 정보를 보고 관리할 수 있습니다. 탐색 창에서 호스팅, 빌드를 선택합니다.

빌드 페이지에는 각 빌드에 대한 다음 정보가 표시됩니다.

> **Note**
>
> 빌드 페이지에는 현재 AWS 리전의 빌드만 표시됩니다.

- 이름 - 업로드한 빌드에 해당하는 이름입니다.
- 상태 - 빌드의 상태입니다. 다음 3개 상태 메시지 중 하나를 표시합니다.
  - 초기화됨 - 업로드가 시작되지 않았거나 아직 진행 중입니다.
  - 준비 완료 - 빌드의 플릿 생성 준비가 완료되었습니다.
  - 실패 - Amazon GameLift가 바이너리를 수신하기 전에 빌드 시간을 초과했습니다.
- 생성 시간 - Amazon GameLift에 빌드를 업로드한 날짜 및 시간입니다.
- 빌드 ID - 업로드 시 빌드에 할당된 고유 ID입니다.
- 버전 - 업로드한 빌드의 버전 레이블입니다.
- 운영 체제 - 빌드를 실행하는 운영 체제입니다. 빌드 OS에 따라 플릿의 인스턴스에 설치하는 Amazon GameLift 운영 체제가 결정됩니다.
- 크기 - Amazon GameLift에 업로드한 빌드 파일의 크기를 메가바이트(MB)로 나타낸 값입니다.
- 플릿 - 빌드와 함께 배포된 플릿의 수입니다.

이 페이지에서 다음 작업을 수행할 수 있습니다.

- 빌드 세부 정보를 봅니다. 빌드 이름을 선택하여 빌드 세부 정보 페이지를 엽니다.
- 빌드에서 새로운 플릿을 만듭니다. 빌드를 선택한 다음 플릿 생성을 선택합니다.
- 빌드 목록을 필터링하고 정렬합니다. 표 맨 위의 컨트롤을 사용합니다.
- 빌드를 삭제합니다. 빌드를 선택한 후 삭제를 선택합니다.

빌드 세부 정보

빌드 페이지에서 빌드 이름을 선택하여 빌드 세부 정보 페이지를 엽니다. 세부 정보 페이지의 개요 섹션에 빌드 페이지 내용과 동일한 빌드 요약 정보가 표시됩니다. 플릿 섹션에는 플릿 페이지와 동일한 요약 정보를 포함하여 빌드로 만든 플릿 목록이 표시됩니다.

스크립트 보기

Amazon GameLift 콘솔의 스크립트 페이지에서 Amazon GameLift에 업로드한 모든 Realtime 서버 스크립트에 대한 정보를 확인할 수 있습니다. 탐색 창에서 호스팅, 스크립트를 선택합니다.

스크립트 페이지에는 각 스크립트에 대한 다음 정보가 표시됩니다.

> **Note**
>
> 스크립트 페이지에는 현재 AWS 리전의 스크립트만 표시됩니다.

- 이름 - 업로드한 스크립트에 해당하는 이름입니다.
- ID - 업로드 시 스크립트에 할당된 고유 ID입니다.
- 버전 - 업로드한 스크립트의 버전 레이블입니다.
- 크기 - Amazon GameLift에 업로드한 스크립트 파일의 크기를 메가바이트(MB)로 나타낸 값입니다.
- 생성 시간 - Amazon GameLift에 스크립트를 업로드한 날짜 및 시간입니다.
- 플릿 - 스크립트와 함께 배포된 플릿의 수입니다.

이 페이지에서 다음 작업을 수행할 수 있습니다.

- 스크립트의 세부 정보를 확인합니다. 빌드 이름을 선택하여 스크립트 세부 정보 페이지를 엽니다.
- 스크립트에서 새로운 플릿을 만듭니다. 스크립트를 선택한 다음 플릿 생성을 선택합니다.
- 스크립트 목록을 필터링하고 정렬합니다. 표 맨 위의 컨트롤을 사용합니다.
- 스크립트를 삭제합니다. 스크립트를 선택한 후 삭제를 선택합니다.

스크립트 세부 정보

스크립트 페이지에서 스크립트 이름을 선택하여 스크립트 세부 정보 페이지를 엽니다. 세부 정보 페이지의 개요 섹션에 빌드 페이지 내용과 동일한 스크립트 요약 정보가 표시됩니다. 플릿 섹션에는 플릿 페이지와 동일한 요약 정보를 포함하여 스크립트로 만든 플릿 목록이 표시됩니다.

플릿 보기

AWS 계정 아래에서 Amazon GameLift의 게임을 호스팅하기 위해 생성한 모든 플릿 관련 정보를 볼 수 있습니다. 목록에는 현재 리전에서 생성한 플릿이 표시됩니다. 플릿 페이지에서 새 플릿을 생성하거나

플릿 하나에 대한 추가 세부 정보를 볼 수 있습니다. 플릿의 세부 정보 페이지에는 사용 정보, 지표, 게임 세션 데이터, 플레이어 세션 데이터가 포함됩니다. 플릿 기록을 편집하거나 플릿을 삭제할 수도 있습니다.

플릿 페이지를 보려면 탐색 창에서 플릿을 선택합니다.

플릿 페이지에는 기본적으로 다음 요약 정보가 표시됩니다. 설정(기어) 버튼을 선택하여 표시된 정보를 사용자 지정할 수 있습니다.

- 이름 - 플릿에 부여된 표시 이름입니다.
- 상태 - 플릿의 상태로, 신규, 다운로드 중, 빌드 중, 활성 중 하나일 수 있습니다.
- 생성 시간 - 플릿이 생성된 날짜와 시간입니다.
- 컴퓨팅 유형 - 게임을 호스팅하는 데 사용되는 컴퓨팅 유형입니다. 플릿은 관리형 EC2 플릿 또는 Anywhere 플릿일 수 있습니다.
- 인스턴스 유형 - 플릿 인스턴스의 컴퓨팅 용량을 결정하는 Amazon EC2 인스턴스 유형입니다.
- 활성 인스턴스 - 플릿에서 사용 중인 EC2 인스턴스 수입니다.
- 원하는 인스턴스 - 활성 상태를 유지할 EC2 인스턴스 수입니다.
- 게임 세션 - 플릿에서 실행 중인 활성 게임 세션의 수입니다. 데이터는 5분 지연됩니다.

플릿 세부 정보 보기

대시보드에서 플릿 페이지에 액세스하거나 플릿 이름을 클릭하여 플릿 페이지에 액세스합니다.

플릿 세부 정보 페이지에서 다음 작업을 수행할 수 있습니다.

- 플릿의 속성, 포트 설정 및 런타임 구성을 업데이트합니다.
- 플릿에 대한 위치를 추가하거나 제거합니다.
- 플릿 용량 설정을 변경합니다.
- 대상 추적 자동 크기 조정을 설정하거나 변경합니다.
- 플릿을 삭제합니다.

세부 정보

플릿 설정

- 플릿 ID - 플릿에 할당된 고유 식별자입니다.

- 이름 - 플릿의 이름입니다.
- ARN - 이 플릿에 할당된 식별자입니다. 플릿의 ARN은 Amazon GameLift 리소스로 플릿을 식별하고 리전 및 AWS 계정을 지정합니다.
- 설명 - 플릿에 대한 간략한 설명을 입력합니다.
- 상태 - 플릿의 현재 상태로, 신규, 다운로드 중, 빌드 중, 활성일 수 있습니다.
- 생성 시간 - 플릿이 생성된 때의 날짜와 시간입니다.
- 종료 시간 - 플릿이 종료된 날짜와 시간입니다. 플릿이 아직 활성 상태인 경우 이 값은 비어 있습니다.
- 플릿 유형 - 플릿이 온디맨드 또는 스팟 인스턴스를 사용하는지 나타냅니다.
- EC2 유형 - 플릿을 생성할 때 플릿으로 선택한 Amazon EC2 인스턴스 유형입니다.
- 인스턴스 역할 - 다른 AWS 리소스에 대한 액세스를 관리하는 AWS IAM 역할(플릿 생성 중에 리소스가 제공된 경우)입니다.
- TLS 인증서 - 게임 서버를 인증하고 모든 클라이언트/서버 통신을 암호화하는 데 TLS 인증서를 사용하도록 플릿을 설정했는지 또는 비활성화했는지 여부입니다.
- 지표 그룹 - 여러 플릿에 대한 지표를 집계하는 데 이 그룹이 사용됩니다.
- 게임 크기 조정 보호 정책 - 플릿의 게임 세션 보호를 위한 현재 설정입니다.
- 플레이어당 최대 게임 세션 - 정책 기간 동안 플레이어가 생성할 수 있는 최대 세션 수입니다.
- 정책 기간 - 플레이어가 생성한 세션 수를 재설정할 때까지 기다려야 하는 시간입니다.

빌드 세부 정보

빌드 세부 정보 섹션에는 플릿에서 호스팅되는 빌드가 표시됩니다. 빌드 이름을 선택하여 전체 빌드 세부 정보 페이지를 확인합니다.

런타임 구성

런타임 구성 섹션에는 각 인스턴스에서 실행할 서버 프로세스가 표시됩니다. 여기에는 게임 서버 실행 파일의 경로와 시작 파라미터 옵션이 포함됩니다.

게임 세션 활성화

게임 세션 활성화 섹션에는 동시에 실행되는 서버 프로세스의 수와 프로세스가 활성화될 때까지 기다려야 하는 시간이 표시됩니다.

EC2 포트 설정

포트 섹션에는 IP 주소와 포트 설정 범위를 비롯해 플릿의 연결 권한이 표시됩니다.

지표

지표 탭은 시간 경과에 따른 플릿 측정치를 그래픽으로 보여줍니다. Amazon GameLift에서 지표 사용에 대한 자세한 내용은 Amazon CloudWatch를 사용한 Amazon GameLift 모니터링 섹션을 참조하세요.

이벤트

이벤트 탭은 플릿에 발생한 모든 이벤트의 로그를 이벤트 코드, 메시지, 타임스탬프와 함께 제공합니다. Amazon GameLift API 참조에서 이벤트 설명을 참조하세요.

크기 조정

크기 조정 탭에는 현재 상태 및 시간 경과에 따른 용량 변화를 포함한 플릿 용량과 관련된 정보가 수록됩니다. 또한 용량 제한을 업데이트하고 Auto Scaling을 관리하는 도구를 제공합니다.

용량 크기 조정

각 플릿 위치의 현재 플릿 용량 설정을 확인합니다. 제한 및 용량 변경에 대한 자세한 내용은 Amazon GameLift 호스팅 용량 확장 섹션을 참조하세요.

- AWS 위치 - 플릿 인스턴스가 배포된 위치의 이름입니다.
- 상태 - 플릿 위치의 호스팅 상태입니다. 위치 상태가 `ACTIVE` 상태여야 게임을 호스팅할 수 있습니다.
- 최소 크기 - 해당 위치에 배포해야 하는 최소 인스턴스 수입니다.
- 원하는 인스턴스 - 위치를 유지할 대상 활성 인스턴스 수입니다. 활성 인스턴스 및 원하는 인스턴스가 동일하지 않은 경우 활성 인스턴스가 원하는 인스턴스가 될 때까지 필요에 따라 인스턴스를 시작하거나 종료하는 크기 조정 이벤트가 시작됩니다.
- 최대 크기 - 해당 위치에 배포할 수 있는 최대 인스턴스 수입니다.
- 사용 가능 - 인스턴스의 서비스 한도에서 사용 중인 인스턴스 수를 뺀 값입니다. 이 값은 해당 위치에 추가할 수 있는 최대 인스턴스 수를 나타냅니다.

자동 크기 조정 정책

이 섹션에서는 플릿에 적용된 자동 크기 조정 정책에 대한 정보를 다룹니다. 대상 기반 정책을 설정하거나 업데이트할 수 있습니다. AWS SDK 또는 CLI를 사용하여 정의해야 하는 플릿의 규칙 기반 정책

이 여기에 표시됩니다. 크기 조정에 대한 자세한 내용은 Amazon GameLift로 플릿 용량 Auto Scaling 섹션을 참조하세요.

크기 조정 기록

시간 경과에 따른 용량 변화 그래프를 볼 수 있습니다.

위치

위치 탭에는 플릿 인스턴스가 배포된 모든 위치를 나열합니다. 위치에는 플릿의 홈 리전 및 추가된 모든 원격 위치가 포함됩니다. 이 탭에서 위치를 직접 추가 또는 제거할 수 있습니다.

- 위치 - 플릿 인스턴스가 배포된 위치의 이름입니다.
- 상태 - 플릿 위치의 호스팅 상태입니다. 위치 상태는 해당 위치의 첫 번째 인스턴스를 활성화하는 프로세스를 추적합니다. 위치 상태가 `ACTIVE` 상태여야 게임을 호스팅할 수 있습니다.
- 활성 인스턴스 - 플릿 위치에서 서버 프로세스를 실행하는 인스턴스의 수입니다.
- 활성 서버 - 플릿 위치에서 게임 세션을 호스팅할 수 있는 게임 서버 프로세스의 수입니다.
- 게임 세션 - 플릿 위치의 인스턴스에서 활성화된 게임 세션의 수입니다.
- 플레이어 세션 - 플릿 위치에서 활성화된 게임 세션에 참여하고 있는 플레이어 세션 수로, 개별 플레이어를 나타냅니다.

게임 세션

게임 세션 탭에는 플릿에서 호스팅되는 과거와 현재 게임 세션이 세부 정보와 함께 나열됩니다. 플레이어 세션을 포함한 추가 게임 세션 정보에 액세스하려면 게임 세션 ID를 선택합니다. 플레이어 세션에 대한 자세한 내용은 게임 및 플레이어 세션에서 데이터 보기 섹션을 참조하세요.

게임 및 플레이어 세션에서 데이터 보기

게임 세션 및 개별 플레이어에 대한 정보를 볼 수 있습니다. 게임 세션 및 플레이어 세션에 대한 자세한 내용은 플레이어의 게임 연결 방법 섹션을 참조하세요.

게임 세션 및 플레이어 데이터를 확인하려면

1. Amazon GameLift 콘솔의 탐색 창에서 플릿을 선택합니다.
2. 게임 세션을 호스팅한 플릿 목록에서 플릿을 선택합니다.

3. 게임 세션 탭을 선택합니다. 이 탭에는 플릿에서 호스팅되는 모든 게임 세션이 요약 정보와 함께 나열됩니다.
4. 게임 세션을 선택하면 게임 세션에 대한 추가 정보와 게임에 연결된 플레이어 목록을 볼 수 있습니다.

세부 정보

개요

이 섹션에는 게임 세션 정보의 요약이 표시됩니다.

- 상태 - 게임 세션 상태입니다.
  - 활성화하는 중 - 인스턴스가 게임 세션을 시작합니다.
  - 활성 - 게임 세션이 실행 중이며 플레이어를 수락할 수 있습니다(세션의 플레이어 생성 정책에 따라).
  - 종료됨 - 게임 세션이 종료되었습니다.
- ARN - 게임 세션의 Amazon 리소스 이름(ARN)입니다.
- 이름 - 게임 세션에 대해 생성된 이름입니다.
- 위치 - Amazon GameLift가 게임 세션을 호스팅한 위치입니다.
- 생성 시간 - Amazon GameLift가 스트림 세션을 생성한 날짜 및 시간입니다.
- 종료 시간 - 게임 세션이 종료된 날짜 및 시간입니다.
- DNS 이름 - 게임 세션의 호스트 이름입니다.
- IP 주소 - 게임 세션에 지정된 IP 주소입니다.
- 포트 - 게임 세션에 연결하는 데 사용되는 포트 번호입니다.
- 만든 사람 ID - 게임 세션을 시작한 플레이어의 고유 식별자입니다.
- 플레이어 세션 생성 정책 - 게임 세션이 새 플레이어를 수락하는지 여부를 나타냅니다.
- 게임 크기 조정 보호 정책 - Amazon GameLift가 플릿에서 시작하는 모든 새 인스턴스를 설정하는 게임 세션 보호 유형입니다.

게임 데이터

시작 시 게임 세션으로 전송할 수 있는 올바른 형식의 데이터입니다.

게임 속성

게임 세션에 영향을 미치는 키 및 값 쌍 속성입니다.

매치메이킹 데이터

FlexMatch 매치메이커 JSON입니다. 매치메이커를 검토하고 편집하려면 매치메이킹 구성 보기을 선택합니다. FlexMatch 매치메이킹에 대한 자세한 내용은 매치메이커 구축을 참조하세요.

플레이어 세션

각 게임 세션에 대해 다음과 같은 플레이어 세션 데이터가 수집됩니다.

- 플레이어 세션 ID - 플레이어 세션에 할당된 식별자입니다.
- 플레이어 ID - 플레이어의 고유 식별자입니다. 이 ID를 선택하여 플레이어에 대한 추가 정보를 얻습니다.
- 상태 - 플레이어 세션의 상태입니다. 가능한 상태는 다음과 같습니다.
  - 예약됨 - 플레이어 세션이 예약되었지만 플레이어가 연결되지 않았습니다.
  - 활성 - 플레이어 세션이 게임 서버에 연결되어 있습니다.
  - 완료됨 - 플레이어 세션이 종료되었으며 더 이상 플레이어가 연결되지 않습니다.
  - 시간 초과 - 플레이어를 연결하지 못했습니다.
- 생성 시간 - 플레이어가 게임 세션에 연결된 시간입니다.
- 종료 시간 - 플레이어가 게임 세션 연결을 해제한 시간입니다.
- 플레이어 데이터 - 플레이어 세션 생성 중에 제공된 플레이어에 대한 정보입니다.

플레이어 정보

현재 리전의 모든 플릿에서 플레이어가 연결한 모든 게임 목록을 포함하여 선택한 플레이어에 대한 추가 정보를 볼 수 있습니다. 이 정보에는 각 플레이어 세션의 상태, 시작 시간, 종료 시간 및 총 연결 시간이 포함됩니다. 관련 게임 세션 및 플릿에 대한 데이터를 보도록 선택할 수 있습니다.

별칭 보기

별칭 페이지에는 현재 리전에서 생성한 플릿 별칭에 대한 정보가 표시됩니다. 별칭 페이지를 보려면 탐색 창에서 별칭을 선택합니다.

별칭 페이지에서 다음 작업을 수행할 수 있습니다.

- 새 별칭을 만듭니다. 별칭 생성을 선택합니다.
- 별칭 테이블을 필터링하고 정렬합니다. 표 맨 위의 컨트롤을 사용합니다.
- 별칭 세부 정보를 봅니다. 별칭 이름을 선택하여 별칭 세부 정보 페이지를 엽니다.
- 별칭을 삭제합니다. 별칭을 선택한 다음 삭제를 선택합니다.

별칭 세부 정보

별칭 세부 정보 페이지에는 별칭에 대한 정보가 표시됩니다.

이 페이지에서 다음을 수행할 수 있습니다.

- 별칭을 편집합니다. 편집을 선택합니다.
- 별칭에 연결한 플릿을 볼 수 있습니다.
- 별칭을 삭제합니다. 삭제를 선택합니다.

별칭 세부 정보에 포함된 내용은 다음과 같습니다.

- ID - 별칭을 식별하는 데 사용되는 고유 번호입니다.
- 설명 - 별칭에 대한 설명입니다.
- ARN - 별칭의 Amazon 리소스 이름(ARN)입니다.
- 생성 - 별칭이 생성된 날짜와 시간입니다.
- 최종 업데이트 - 별칭이 마지막으로 업데이트된 날짜 및 시간입니다.
- 라우팅 유형 - 다음 항목으로 구성된 별칭의 라우팅 유형입니다.
  - 단순 - 플레이어 트래픽을 지정된 플릿 ID로 라우팅합니다. 언제든 별칭에 대한 플릿 ID를 업데이트할 수 있습니다.
  - 터미널 - 메시지를 클라이언트로 다시 전달합니다. 예를 들어 만료된 클라이언트를 이용하는 플레이어를 업그레이드를 받을 수 있는 위치로 보낼 수 있습니다.
- 태그 — 별칭을 식별하는 데 사용되는 키 및 값 쌍입니다.

대기열 보기

기존 게임 세션 배치 대기열 전체에 대한 정보를 볼 수 있습니다. 대기열 페이지에는 현재 리전에 생성된 대기열이 표시됩니다. 대기열 페이지에서 새로운 대기열을 생성하거나 기존 대기열을 삭제하거나 선택한 대기열에 대한 세부 정보 페이지를 열 수 있습니다. 각 대기열 세부 정보 페이지에는 대기열의

구성 및 계측치 데이터가 들어 있습니다. 대기열에 대한 자세한 내용은 게임 세션 배치에 대한 Amazon GameLift 대기열 설정 섹션을 참조하세요.

대기열 페이지에 각 대기열에 대해 다음과 같은 요약 정보가 표시됩니다.

- 대기열 이름 - 대기열에 지정된 이름입니다. 새로운 게임 세션 요청에서 이 이름으로 대기열을 지정합니다.
- 대기열 제한 시간 - 시간 초과되기 전에 대기열에 게임 세션 배치 요청이 유지되는 최대 시간(초)입니다.
- 대기열의 대상 - 대기열 구성에 나열되어 있는 플릿 수입니다. Amazon GameLift는 대기열에 있는 모든 플릿에 새 게임 세션을 배치합니다.

대기열 세부 정보 보기

임의의 대기열에 대해 대기열 구성, 측정치 등 세부 정보에 액세스할 수 있습니다. 대기열 세부 정보 페이지를 열려면 대기열 페이지로 이동하여 대기열 이름을 선택합니다.

대기열 세부 정보 페이지에는 요약 표와 추가 정보가 수록된 탭이 표시됩니다. 이 페이지에서 다음 작업을 수행할 수 있습니다.

- 대기열 구성, 대상 목록, 플레이어 지연 시간 정책을 업데이트합니다. 편집을 선택합니다.
- 대기열을 삭제합니다. 대기열이 삭제되면 해당 대기열 이름을 참조하는 새 게임 세션에 대한 요청은 모두 실패합니다. 삭제를 선택합니다.

> **Note**
>
> 삭제된 대기열을 복원하려면 삭제된 대기열의 이름을 사용하여 새 대기열을 생성합니다.

세부 정보

개요

개요 섹션에는 대기열의 Amazon 리소스 이름(ARN) 및 제한 시간이 표시됩니다. Amazon GameLift의 다른 작업이나 영역에서 대기열을 참조할 때 ARN을 사용할 수 있습니다. 제한 시간은 시간 초과되기 전에 대기열에 게임 세션 배치 요청이 유지되는 최대 시간(초)입니다.

이벤트 알림

이벤트 알림 섹션에는 Amazon GameLift가 이벤트 알림을 게시하는 SNS 주제 및 이 대기열에서 생성되는 모든 이벤트에 추가되는 이벤트 데이터가 나열됩니다.

태그

태그 테이블에는 리소스에 태그를 지정하는 데 사용된 키와 값이 표시됩니다. 태그 지정에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.

지표

지표 탭은 시간 경과에 따른 대기열 측정치를 그래픽으로 보여줍니다.

대기열 지표에는 리전별로 구성된 성공적인 배치를 포함하여 대기열 전체의 배치 활동을 설명하는 다양한 정보가 포함됩니다. 리전 데이터를 사용하여 게임을 호스팅하는 위치를 파악할 수 있습니다. 리전 배치 지표는 전체 대기열 설계와 관련된 문제를 발견하는 데 도움이 될 수 있습니다.

대기열 지표는 Amazon CloudWatch에서도 사용할 수 있습니다. 사용 가능한 지표에 대한 자세한 설명은 Amazon GameLift 대기열 지표 섹션을 참조하세요.

대상

대상 탭에는 대기열에 대해 나열된 모든 집합 또는 별칭이 표시됩니다.

Amazon GameLift가 대상에서 새 게임 세션 호스팅에 사용할 수 있는 리소스를 검색할 때는 여기 나열된 기본 순서를 검색합니다. 나열된 첫 번째 대상에 용량이 있으면 Amazon GameLift가 새 게임 세션을 해당 위치에 배치합니다. 플레이어 지연 시간 데이터를 제공하여 개별 게임 세션 배치 요청이 기본 순서보다 우선하도록 할 수 있습니다. 이 데이터는 Amazon GameLift가 평균 플레이어 지연 시간이 가장 낮은 사용 가능한 대상을 검색하도록 지시합니다. 대기열에 대한 자세한 내용은 게임 세션 대기열 설계 섹션을 참조하세요.

세션 배치

플레이어 지연 시간 정책

플레이어 지연 시간 정책 섹션에는 대기열에서 사용하는 모든 정책이 표시됩니다. 테이블에는 정책이 적용되는 순서대로 나열되어 있습니다.

위치

위치 섹션에는 이 대기열이 게임 세션을 배치할 수 있는 위치가 표시됩니다.

우선 순위

우선 순위 섹션에는 대기열이 게임 세션 세부 정보를 평가하는 순서가 표시됩니다.

위치 순서

위치 순서 섹션에는 게임 세션을 배치할 때 대기열이 사용하는 기본 순서가 표시됩니다. 다른 유형의 우선 순위를 정의하지 않은 경우 대기열은 이 순서를 사용합니다.

아마존 모니터링 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

모니터링은 Amazon 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 GameLift 있어 중요한 부분입니다. GameLiftAmazon에서 메트릭을 사용하는 주요 용도는 세 가지입니다. 시스템 상태를 모니터링하고 경보를 설정하고, 게임 서버 성능 및 사용량을 추적하고, 수동 또는 자동 크기 조정을 사용하여 용량을 관리하는 것입니다.

AWS 는 Amazon을 감시하고, 문제 발생 시 보고하고 GameLift, 적절한 경우 자동 조치를 취할 수 있는 다음과 같은 모니터링 도구를 제공합니다.

- 아마존 GameLift 콘솔
- Amazon CloudWatch -- Amazon GameLift 지표는 물론 AWS 서비스에서 실행 중인 다른 AWS 리소스 및 애플리케이션에 대한 지표도 실시간으로 모니터링할 수 있습니다. CloudWatch 사용자 지정 대시보드를 생성하는 도구와 지표가 지정된 임계값에 도달하면 알림을 보내거나 조치를 취하는 경보를 설정하는 기능을 비롯한 모니터링 기능 세트를 제공합니다.
- AWS CloudTrail— Amazon 및 기타 AWS 서비스에 대해 사용자 계정에서 또는 사용자 AWS 계정을 대신하여 수행한 모든 API 호출 GameLift 및 관련 이벤트를 캡처합니다. 지정한 Amazon S3 버킷에 로그 파일로 데이터를 전달합니다. 어떤 사용자와 계정이 전화를 걸었는지 AWS, 어떤 소스 IP 주소에서 전화를 걸었는지, 언제 호출이 발생했는지 확인할 수 있습니다.
- 게임 세션 로그 - 게임 세션의 사용자 지정 서버 메시지를 Amazon S3에 저장된 로그 파일로 출력할 수 있습니다.

주제

- Amazon CloudWatch를 사용한 Amazon GameLift 모니터링
- AWS CloudTrail를 사용하여 Amazon GameLift API 호출 로깅
- Amazon GameLift에서 서버 메시지 로깅

Amazon CloudWatch를 사용한 Amazon GameLift 모니터링

원시 데이터를 수집하고 읽기 가능하며 실시간에 가까운 지표로 처리하는 AWS 서비스인 Amazon CloudWatch를 통해 Amazon GameLift를 모니터링할 수 있습니다. 이러한 통계는 Amazon GameLift를

통해 호스팅하는 게임 서버가 어떻게 실행되고 있는지에 대한 기록 데이터를 제공하기 위해 15개월간 보관됩니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수 있습니다. 자세한 정보는 Amazon CloudWatch 사용 설명서를 참조하세요.

다음 표는 Amazon GameLift에서 사용 가능한 지표와 차원을 열거한 것입니다. CloudWatch에서 사용할 수 있는 모든 지표는 Amazon GameLift 콘솔에서 사용할 수 있으며, 이를 통해 데이터를 여러 개의 사용자 지정할 수 있는 그래프로 제공합니다. 게임의 CloudWatch 지표에 액세스하려면 , AWS Management Console, AWS CLI 또는 CloudWatch API를 사용합니다.

지표에 위치가 없는 경우 홈 위치를 사용합니다.

Amazon GameLift 지표의 차원

Amazon GameLift는 다음 차원을 기준으로 지표 필터링을 지원합니다.

차원	설명
`Location`	플릿 배포 위치에 대한 지표를 필터링합니다. 지표에 위치가 없는 경우 홈 위치를 사용합니다.
`FleetId`	단일 플릿에 대한 지표를 필터링합니다. 이 차원은 인스턴스, 서버 프로세스, 게임 세션 및 플레이어 세션에 대한 모든 플릿 지표에 사용할 수 있습니다.
`MetricGroup`	플릿 모음에 대한 지표를 필터링합니다. 지표 그룹 이름을 플릿 속성에 추가하면 플릿이 지표 그룹에 추가됩니다(UpdateFleetAttributes() 참조). 이 차원은 인스턴스, 서버 프로세스, 게임 세션 및 플레이어 세션에 대한 모든 플릿 지표에 사용할 수 있습니다.
`QueueName`	단일 대기열에 대한 지표를 필터링합니다. 이 차원은 게임 세션 배치 대기열에 국한된 지표에 사용됩니다.
`ConfigurationName`	단일 매치메이킹 구성에 대한 지표를 필터링합니다. 이 차원은 매치메이킹 구성의 지표에 사용됩니다.
`ConfigurationName-RuleName`	매치메이킹 구성과 매치메이킹 규칙의 교차에 대한 지표를 필터링합니다. 이 차원은 매치메이킹 규칙에 국한된 지표에 사용됩니다.

차원	설명
InstanceType	“c4.large”와 같은 EC2 인스턴스 유형 지정에 대한 지표를 필터링합니다. 이 차원은 스팟 인스턴스의 지표에 사용됩니다.
OperatingSystem	인스턴스의 운영 체제에 대한 지표를 필터링합니다. 이 차원은 스팟 인스턴스의 지표에사용됩니다.
GameServerGroup	게임 서버 그룹에 대한 FleetIQ 지표를 필터링합니다.

Amazon GameLift 플릿 지표

AWS/GameLift 네임스페이스에는 플릿 또는 플릿 그룹에서 다음과 같은 활동 관련 지표가 포함되어 있습니다. 플릿은 관리형 Amazon GameLift 솔루션과 함께 사용됩니다. Amazon GameLift 서비스는 1분마다 지표를 CloudWatch에 전송합니다.

인스턴스

지표	설명
ActiveInstances	활성 서버 프로세스가 실행 중인 것을 의미하는 ACTIVE 상태의 인스턴스 수. 여기에는 유휴 상태의 인스턴스와 게임 세션을 하나 이상 호스팅하고 있는 인스턴스도 포함됩니다. 이 지표는 현재 총 인스턴스 용량을 측정하기 때문에 Auto Scaling에 사용할 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
DesiredInstances	Amazon GameLift가 플릿에서 유지 관리하기 위해 작업 중인 목표 활성 인스턴스 수. Auto Scaling 기능을 사용할 경우 이 값은 현재 적용 중인 조정 정책에 따라

지표	설명
	결정됩니다. Auto Scaling 기능을 사용하지 않을 때는 수동으로 값을 설정해야 합니다. 플릿 지표 그룹의 데이터를 확인할 때는 이 지표를 사용하지 못합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
IdleInstances	현재 0개의 게임 세션을 호스팅하고 있는 활성 인스턴스 수. 이 지표는 사용할 수는 있지만 아직 미사용 중인 용량을 측정합니다. Auto Scaling에 사용할 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
MaxInstances	플릿에 허용되는 최대 인스턴스 수. 플릿의 최대 인스턴스 수에 따라 수동 또는 자동 확장 시 최대 용량이 결정됩니다. 플릿 지표 그룹의 데이터를 확인할 때는 이 지표를 사용하지 못합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
MinInstances	플릿에 허용되는 최소 인스턴스 수. 플릿의 최소 인스턴스 수에 따라 수동 또는 자동 축소 시 최소 용량이 결정됩니다. 플릿 지표 그룹의 데이터를 확인할 때는 이 지표를 사용하지 못합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
PercentIdleInstances	유휴 상태인 모든 활성 인스턴스의 비율(IdleInstances / ActiveInstances 로 계산). 이 지표는 Auto Scaling에 사용할 수 있습니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
RecycledInstances	재활용 및 교체된 스팟 인스턴스의 수. Amazon GameLift는 현재 게임 세션을 호스팅하고 있지 않고 중단 가능성이 높은 스팟 인스턴스를 재활용합니다. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

지표	설명
InstanceInterruptions	중단된 스팟 인스턴스의 수. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
CPUUtilization	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. Amazon EC2에서 인스턴스 실행에 사용하는 물리적 CPU 시간의 비율로, 사용자 코드와 Amazon EC2 코드를 실행하는 데 소요되는 시간이 모두 포함됩니다. 레거시 디바이스 시뮬레이션, 비 레거시 디바이스 구성, 인터럽트가 많은 워크로드, 실시간 마이그레이션, 실시간 업데이트 등의 요인으로 인해 운영 체제의 도구에 CloudWatch와 다른 비율이 표시될 수 있습니다. 단위: 백분율
NetworkIn	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스가 모든 네트워크 인터페이스에서 받은 바이트 수입니다. 이 측정치는 단일 인스턴스에서 애플리케이션으로 들어오는 네트워크 트래픽의 볼륨을 식별합니다. 단위: 바이트

지표	설명
NetworkOut	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스가 모든 네트워크 인터페이스에서 보낸 바이트 수입니다. 이 측정치는 단일 인스턴스에서 애플리케이션으로 나가는 네트워크 트래픽의 볼륨을 식별합니다. 단위: 바이트
DiskReadBytes	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스에 사용할 수 있는 모든 인스턴스 스토어 볼륨에서 읽은 바이트 수. 이 지표는 애플리케이션이 인스턴스의 하드 디스크에서 읽는 데이터 볼륨을 결정하는 데 사용됩니다. 이를 사용하여 애플리케이션의 속도를 결정할 수 있습니다. 단위: 바이트
DiskWriteBytes	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스에 사용할 수 있는 모든 인스턴스 스토어 볼륨에 쓴 바이트 수. 이 지표는 애플리케이션이 인스턴스의 하드 디스크에 쓰는 데이터 볼륨을 결정하는 데 사용됩니다. 이를 사용하여 애플리케이션의 속도를 결정할 수 있습니다. 단위: 바이트

지표	설명
DiskReadOps	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 지정된 시간 내에 인스턴스에 사용할 수 있는 모든 인스턴스 스토어 볼륨에서 읽기 작업 완료. 기간의 평균 IOPS(초당 I/O 작업 수)를 계산하려면 기간의 총 작업 수를 해당 기간의 초 수로 나누세요. 단위: 개수
DiskWriteOps	EC2 지표. Amazon GameLift의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 지정된 시간 내에 인스턴스에 사용할 수 있는 모든 인스턴스 스토어 볼륨에 대한 쓰기 작업 완료. 기간의 평균 IOPS(초당 I/O 작업 수)를 계산하려면 기간의 총 작업 수를 해당 기간의 초 수로 나누세요. 단위: 개수

서버 프로세스

지표	설명
ActiveServerProcesses	현재 실행 중이며, 게임 세션을 호스팅할 수 있다는 것을 의미하는 ACTIVE 상태의 서버 프로세스 수. 여기에는 유휴 상태의 서버 프로세스와 게임 세션을 호스팅하고 있는 서버 프로세스도 포함됩니다. 이 지표는 현재 총 서버 프로세스 용량을 측정합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
HealthyServerProcesses	정상 상태를 보고하고 있는 활성 서버 프로세스 수. 이 지표는 플릿의 전반적인 게임 서버 상태를 추적하는 데 유용합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
PercentHealthyServerProcesses	정상 상태를 보고하고 있는 모든 활성 서버 프로세스의 비율(HealthyServerProcesses / ActiveServerProcesses 로 계산). 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
ServerProcessAbnormalTerminations	마지막 보고 이후 비정상적인 환경으로 인해 종료된 서버 프로세스 수. 이 지표에는 Amazon GameLift 서비스에서 시작되었다가 종료된 서버 프로세스도 포함됩니다. 이러한 경우는 서버 프로세스가 응답을 멈추거나, 계속해서 상태 검사 실패를 보고하거나, ProcessEnding() 호출을 통해 정상적으로 종료되지 않았을 때 발생합니다. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

지표	설명
ServerProcessActivations	마지막 보고 이후 상태가 ACTIVATING에서 ACTIVE로 전환된 서버 프로세스 수. 서버 프로세스는 활성 상태가 되어야만 게임 세션을 호스팅할 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
ServerProcessTerminations	마지막 보고 이후 종료된 서버 프로세스 수. 여기에는 정상적인 프로세스 종료와 비정상적인 프로세스 종료를 포함하여 어떤 이유든지 TERMINATED 상태로 전환된 서버 프로세스가 모두 포함됩니다. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

게임 세션

지표	설명
ActivatingGameSessions	시작 진행 중인 것을 의미하는 ACTIVATING 상태의 게임 세션 수. 게임 세션은 활성 상태가 되어야만 플레이어를 호스팅할 수 있습니다. 일정 시간 높은 수를 유지할 경우 게임 세션이 ACTIVATING에서 ACTIVE 상태로 전환되지 않는 것을 의미할 수도 있습니다. Auto Scaling에 사용할 수 있습니다. 단위: 개수

지표	설명
	상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
`ActiveGameSessions`	플레이어를 호스팅할 수 있으며, 0명 이상의 플레이어를 호스팅하고 있다는 것을 의미하는 ACTIVE 상태의 게임 세션 수. 이 지표는 현재 호스팅 중인 총 게임 세션 수를 측정합니다. Auto Scaling에 사용할 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
AvailableGameSessions	현재 게임 세션을 호스팅하는 데 사용되지 않는 활성 상태의 서버 프로세스로, 지연 없이 새 게임 세션을 시작하여 새 서버 프로세스 또는 인스턴스를 가동할 수 있습니다. Auto Scaling에 사용할 수 있습니다. **Note** 동시 게임 세션 활성화를 제한하는 플릿의 경우 ConcurrentActivatableGameSessions 지표를 사용합니다. 이 지표는 어떤 유형의 지연 없이 시작할 수 있는 새 게임 세션의 수를 더 정확하게 나타냅니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
ConcurrentActivatableGameSessions	현재 게임 세션을 호스팅하는 데 사용되지 않는 활성 상태의 서버 프로세스로, 새 게임 세션을 즉시 시작할 수 있습니다. 이 지표는 다음과 같은 점에서 AvailableGameSessions 과 다릅니다. 게임 세션 활성화 제한으로 인해 현재 새 게임 세션을 활성화할 수 없는 서버 프로세스는 계산하지 않는 다는 점입니다. (플릿 RuntimeConfiguration 옵션 설정 MaxConcurrentGameSessionActivations 참조). 게임 세션 활성화를 제한하지 않는 플릿의 경우 이 지표는 AvailableGameSessions 과 동일합니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
PercentAvailableGameSessions	모든 활성 서버 프로세스(정상 또는 비정상)에서 아직 사용 중이지 않은 게임 세션 슬롯의 비율(AvailableGameSessions / [ActiveGameSessions + AvailableGameSessions + unhealthy server processes] 로 계산). Auto Scaling에 사용할 수 있습니다. 단위: 백분율 관련 CloudWatch 통계: Average 차원: 위치

지표	설명
GameSessionInterruptions	중단된 스팟 인스턴스의 게임 세션 수 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

플레이어 세션

지표	설명
CurrentPlayerSessions	ACTIVE 상태(플레이어가 활성 게임 세션에 연결된 상태)이거나, 혹은 RESERVED 상태(플레이어에게 게임 세션 슬롯이 할당되었지만 아직 연결되지 않은 상태)인 플레이어 세션 수. Auto Scaling에 사용할 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum
PlayerSessionActivations	마지막 보고 이후 상태가 RESERVED에서 ACTIVE로 전환된 플레이어 세션 수. 이러한 경우는 플레이어가 활성 게임 세션에 성공적으로 연결되었을 때 발생합니다. 단위: 개수 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum

Amazon GameLift 대기열 지표

Amazon GameLift 네임스페이스에는 게임 세션 배치 대기열에서 다음과 같은 활동 관련 지표가 포함되어 있습니다. 대기열은 관리형 Amazon GameLift 솔루션과 함께 사용됩니다. Amazon GameLift 서비스는 1분마다 지표를 CloudWatch에 전송합니다.

지표	설명
AverageWaitTime	대기열에서 상태가 PENDING인 게임 세션 배치 요청이 실행될 때까지 기다린 평균 대기 시간 단위: 초 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum 차원: 위치
FirstChoiceNotViable	성공적으로 위치된 게임 세션은 첫 번째 선택 플릿에 위치하지 않습니다. 이 플릿이 (고 간섭 속도를 갖는 스팟 플릿처럼) 실행 가능하지 않다고 여겨지기 때문입니다. 이 지표는 지연 시간이 아닌 비용을 기반으로 합니다. 첫 번째 선택 플릿은 대기열에 나열된 첫 번째 플릿이거나, 배치 요청에 플레이어 지연 시간 데이터가 포함된 경우 FleetIQ 우선순위 지정에 따라 선택한 첫 번째 플릿입니다. 사용할 수 있는 스팟 플릿이 없는 경우에는 해당 지역의 플릿이 선택될 수 있습니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
FirstChoiceOutOfCapacity	성공적으로 위치된 게임 세션은 첫 번째 선택 플릿에 위치하지 않습니다. 이 플릿이 가능한 리소스가 없기 때문입니다. 첫 번째 선택 플릿은 대기열에 나열된 첫 번째 플릿이거나, 배치 요청에 플레이어 지연 시간 데이터가 포함된 경우 정의된 FleetIQ 우선순위 지정에 따라 선택한 첫 번째 플릿입니다.

지표	설명
	단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
LowestLatencyPlacement	게임 세션은 플레이어의 대기열 최저 가능 지연 시간을 제공하는 리전에 성공적으로 배치됩니다. 이 지표는 배치 요청에 플레이어 지연 시간 데이터가 포함되는 경우에만 생성됩니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
LowestPricePlacement	게임 세션은 선택한 리전에 대해 대기열 최저 가격을 제공하는 플릿에 성공적으로 배치됩니다. 이 플릿은 스팟 플릿이거나 대기열이 스팟 인스턴스가 없을 경우 온디맨드 인스턴스일 수 있습니다. 이 지표는 배치 요청에 플레이어 지연 시간 데이터가 포함되는 경우에만 생성됩니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
Placement <region name>	게임 세션은 특정 리전에 위치한 플릿에 성공적으로 배치됩니다. 이 지표는 리전에 따라 PlacementsSucceeded 지표를 나눕니다. 단위: 개수 관련 CloudWatch 통계: Sum

지표	설명
PlacementsCanceled	마지막 보고 이후 시간 초과 이전에 취소된 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsFailed	마지막 보고 이후 모든 이유에 따라 실패한 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsStarted	마지막 보고 이후 대기열에 새롭게 추가된 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsSucceeded	마지막 보고 이후 새로운 게임 세션까지 이어진 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsTimedOut	마지막 보고 이후 대기열의 시간 제한에 걸려 실행되지 못한 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum

지표	설명
QueueDepth	대기열에서 상태가 PENDING인 게임 세션 배치 요청 수 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum 차원: 위치

Amazon GameLift 매치메이킹 지표

Amazon GameLift 네임스페이스에는 매치메이킹 구성 및 매치메이킹 규칙에 따른 FlexMatch 활동과 관련한 지표가 포함되어 있습니다. FlexMatch 매치메이킹은 관리형 Amazon GameLift 솔루션과 함께 사용됩니다. Amazon GameLift 서비스는 1분마다 지표를 CloudWatch에 전송합니다.

매치메이킹 작업의 시퀀스에 대한 자세한 내용은 Amazon GameLift FlexMatch 작동 방식을 참조하세요.

매치메이킹 구성

지표	설명
CurrentTickets	현재 처리되고 있거나 처리 대기 중인 매치메이킹 요청입니다. 단위: 개수 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
MatchAcceptancesTimedOut	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 수락 과정에서 시간을 초과한 잠재적 매치입니다. 단위: 개수

지표	설명
	관련 CloudWatch 통계: Sum
MatchesAccepted	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 수락된 잠재적 매치입니다. 단위: 개수 관련 CloudWatch 통계: Sum
MatchesCreated	마지막 보고 이후 생성된 잠재적 매치입니다. 단위: 개수 관련 CloudWatch 통계: Sum
MatchesPlaced	마지막 보고 이후 게임 세션에 성공적으로 배치된 매치입니다. 단위: 개수 관련 CloudWatch 통계: Sum
MatchesRejected	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 적어도 한 명의 플레이어에 의해 거부된 잠재적 매치입니다. 단위: 개수 관련 CloudWatch 통계: Sum
PlayersStarted	마지막 보고 이후 추가된 매치메이킹 티켓의 플레이어입니다. 단위: 개수 관련 CloudWatch 통계: Sum

지표	설명
TicketsFailed	마지막 보고 이후 매치 실패로 이어진 매치메이킹 요청입니다. 단위: 개수 관련 CloudWatch 통계: Sum
TicketsStarted	마지막 보고 이후 생성된 신규 매치메이킹 요청입니다. 단위: 개수 관련 CloudWatch 통계: Sum
TicketsTimedOut	마지막 보고 이후 타임아웃 한도에 도달한 매치메이킹 요청입니다. 단위: 개수 관련 CloudWatch 통계: Sum
TimeToMatch	마지막 보고 이전에 취소된 잠재적 매치에 보내진 매치메이킹 요청의 경우, 티켓 생성과 잠재적 매치 생성 간의 시간차입니다. 단위: 초 관련 CloudWatch 통계: Data Samples, Average, Minimum, Maximum
TimeToTicketCancel	마지막 보고 이전에 취소된 매치메이킹 요청의 경우, 티켓 생성과 취소 간의 시간차입니다. 단위: 초 관련 CloudWatch 통계: Data Samples, Average, Minimum, Maximum

지표	설명
TimeToTicketSuccess	마지막 보고 이전에 성공한 매치메이킹 요청의 경우, 티켓 생성과 성공한 매치 배치 간의 시간차입니다. 단위: 초 관련 CloudWatch 통계: Data Samples, Average, Minimum, Maximum

매치메이킹 규칙

지표	설명
RuleEvaluationsPassed	마지막 보고 이후 통과된 매치메이킹 프로세스 동안의 규칙 평가입니다. 이 지표는 상위 50개 규칙으로 제한됩니다. 단위: 개수 관련 CloudWatch 통계: Sum
RuleEvaluationsFailed	마지막 보고 이후 실패한 매치메이킹 동안의 규칙 평가입니다. 이 지표는 상위 50개 규칙으로 제한됩니다. 단위: 개수 관련 CloudWatch 통계: Sum

Amazon GameLift FleetIQ 지표

Amazon GameLift 네임스페이스에는 게임 호스팅을 위한 FleetIQ 독립형 솔루션의 일부로서 FleetIQ 게임 서버 그룹 및 게임 서버 활동에 대한 지표가 포함되어 있습니다. Amazon GameLift 서비스는 1분마다 지표를 CloudWatch에 전송합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon CloudWatch를 사용하여 오토 스케일링 및 인스턴스 모니터링을 참조하세요.

지표	설명
AvailableGameServers	게임을 실행하는 데 사용할 수 있고 현재 게임 플레이로 점유되지 않은 게임 서버 수입니다. 여기에는 클레임되었지만 아직 AVAILABLE 상태인 게임 서버도 포함됩니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup
UtilizedGameServers	현재 게임 플레이로 점유된 게임 서버 수입니다. 이 숫자에는 UTILIZED 상태인 게임 서버가 포함됩니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup
DrainingAvailableGameServers	현재 게임 플레이를 지원하지 않으며 종료 예정된 인스턴스의 게임 서버 수입니다. 이러한 게임 서버는 새 클레임 요청에 대한 응답으로 클레임되는 우선 순위가 가장 낮습니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup
DrainingUtilizedGameServers	현재 게임 플레이를 지원하며 종료 예정된 인스턴스의 게임 서버 수입니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup

지표	설명
PercentUtilizedGameServers	현재 게임 실행을 지원하는 게임 서버의 비율입니다. 이 지표는 현재 사용 중인 게임 서버 용량을 나타냅니다. 플레이어 수요에 맞게 인스턴스를 동적으로 추가 및 제거할 수 있는 Auto Scaling 정책을 실행하는 데 유용합니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: GameServerGroup
GameServerInterruptions	제한된 스팟 가용성으로 인해 중단된 스팟 인스턴스의 게임 서버입니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup, InstanceType
InstanceInterruptions	제한된 가용성으로 인해 중단된 스팟 인스턴스입니다. 단위: 개수 관련 CloudWatch 통계: Sum 차원: GameServerGroup, InstanceType

AWS CloudTrail를 사용하여 Amazon GameLift API 호출 로깅

Amazon GameLift는 Amazon GameLift에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail와 통합됩니다. CloudTrail은 Amazon GameLift에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon GameLift 콘솔로부터의 호출과 Amazon GameLift API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Amazon GameLift 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하

지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon GameLift에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon GameLift 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. Amazon GameLift에서 활동이 수행되면 해당 활동은 이벤트 기록에서 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amazon GameLift의 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 지역에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 Amazon GameLift 작업은 CloudTrail에서 로깅되며 Amazon GameLift API 참조에 설명되어 있습니다. 예를 들어 `CreateGameSession`, `CreatePlayerSession`, `UpdateGameSession` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

Amazon GameLift 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateFleet` 및 `DescribeFleetAttributes` 작업을 보여 주는 CloudTrail 로그 항목을 나타낸 예제입니다.

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


Amazon GameLift에서 서버 메시지 로깅

Amazon GameLift 서버의 사용자 지정 서버 메시지를 로그 파일로 캡처할 수 있습니다. 로깅을 구성하는 방법은 사용자 지정 서버를 사용하는지 아니면 Realtime 서버를 사용하는지에 따라 다릅니다(이 장의 해당 하위 섹션 참조).

주제

- 서버 메시지 로깅(사용자 지정 서버)
- 서버 메시지 로깅(Realtime 서버)

서버 메시지 로깅(사용자 지정 서버)

Amazon GameLift 사용자 지정 서버의 사용자 지정 서버 메시지를 로그 파일로 캡처할 수 있습니다. Realtime 서버 로깅에 대한 자세한 내용은 서버 메시지 로깅(Realtime 서버)을 참조하세요.

> **Important**
>
> 게임 세션당 로그 파일 크기에는 제한이 있습니다 (의 Amazon GameLift 엔드포인트 및 할당량 참조). AWS 일반 참조 게임 세션이 종료되면 Amazon은 서버 로그를 Amazon Simple Storage Service (Amazon S3) 에 GameLift 업로드합니다. GameLift Amazon은 제한을 초과하는 로그를 업로드하지 않습니다. 로그는 매우 빠르게 증가하여 크기 제한을 초과할 수 있습니다. 로그를 모니터링하고 로그 출력을 필요한 메시지로만 제한해야 합니다.

사용자 지정 서버의 로깅 구성

Amazon GameLift 사용자 지정 서버를 사용하면 서버 프로세스 구성의 일부로 구성하는 자체 코드를 작성하여 로깅을 수행합니다. Amazon은 로깅 구성을 GameLift 사용하여 각 게임 세션이 끝날 때 Amazon S3에 업로드해야 하는 파일을 식별합니다.

다음 지침은 간소화된 코드 예제를 사용하여 로깅을 구성하는 방법을 보여줍니다.

C++

로깅을 구성하려면(C++)

1. 게임 서버 로그 파일의 디렉터리 경로인 문자열 벡터를 생성합니다.

```
std::string serverLog("serverOut.log");        // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. 벡터를 ProcessParameters객체의 벡터로 제공하십시오. LogParameters

```
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. ProcessReady() 를 호출할 때 ProcessParameters객체를 제공하십시오.

```
Aws::GameLift::GenericOutcome outcome =
   Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

전체 예제에 대한 자세한 내용은 ProcessReady() 섹션을 참조하세요.

C#

로깅을 구성하려면(C#)

1. 게임 서버 로그 파일의 디렉터리 경로인 문자열 목록을 생성합니다.

```
List<string> logPaths = new List<string>();
logPaths.Add("C:\\game\\serverOut.txt");     // Example of a log file that the
 game server writes
```

2. 목록을 ProcessParameters객체의 목록으로 제공하십시오. LogParameters

```
var processReadyParameter = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(logPaths));
```

3. ProcessReady() 를 호출할 때 ProcessParameters객체를 제공하십시오.

```
var processReadyOutcome =
   GameLiftServerAPI.ProcessReady(processReadyParameter);
```

전체 예제에 대한 자세한 내용은 ProcessReady() 섹션을 참조하세요.

로그에 기록

로그 파일은 서버 프로세스가 시작된 후에 존재합니다. 어떤 방법으로든 파일에 기록하여 로그에 기록할 수 있습니다. 서버의 표준 출력 및 오류 출력을 모두 캡처하려면 다음 예제와 같이 출력 스트림을 로그 파일에 다시 매핑합니다.

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"));
```

서버 로그 액세스

게임 세션이 종료되면 Amazon은 GameLift 자동으로 로그를 Amazon S3 버킷에 저장하고 14일 동안 보관합니다. GetGameSessionLogUrlAPI 작업을 사용하여 게임 세션의 로그 위치를 가져올 수 있습니다. 로그를 다운로드하려면 작업에서 반환하는 URL을 사용합니다.

서버 메시지 로깅(Realtime 서버)

Realtime 서버의 사용자 지정 서버 메시지를 로그 파일로 캡처할 수 있습니다. 사용자 지정 서버 로깅에 대한 자세한 내용은 서버 메시지 로깅(사용자 지정 서버)을 참조하세요.

로그 파일에 출력할 수 있는 메시지 유형에는 여러 가지가 있습니다(서버 스크립트의 메시지 로깅 참조). 사용자 지정 메시지 외에도, Realtime 서버는 동일한 메시지 유형을 사용하여 시스템 메시지를 출력하고 동일한 로그 파일에 기록합니다. 플릿의 로깅 수준을 조정하여 서버에서 생성되는 로깅 메시지의 양을 줄일 수 있습니다(로깅 수준 조정 참조).

> **Important**
>
> 게임 세션당 로그 파일 크기에는 제한이 있습니다 (의 Amazon GameLift 엔드포인트 및 할당량 참조). AWS 일반 참조 게임 세션이 종료되면 Amazon은 서버 로그를 Amazon Simple Storage Service (Amazon S3) 에 GameLift 업로드합니다. GameLift Amazon은 제한을 초과하는 로그를 업로드하지 않습니다. 로그는 매우 빠르게 증가하여 크기 제한을 초과할 수 있습니다. 로그를 모니터링하고 로그 출력을 필요한 메시지로만 제한해야 합니다.

서버 스크립트의 메시지 로깅

Realtime 서버용 스크립트에서 사용자 지정 메시지를 출력할 수 있습니다. 다음 단계에 따라 로그 파일에 서버 메시지를 전송합니다.

1. 로거 객체에 대한 참조를 보관할 변수를 생성합니다.

```
var logger;
```

2. `init()` 함수에서는 세션 객체에서 로거를 가져와 로거 변수에 할당합니다.

```
function init(rtSession) {
    session = rtSession;
    logger = session.getLogger();
}
```

3. 로거에서 적절한 함수를 호출하여 메시지를 출력합니다.

   디버그 메시지

   ```
   logger.debug("This is my debug message...");
   ```

   정보 메시지

   ```
   logger.info("This is my info message...");
   ```

   경고 메시지

   ```
   logger.warn("This is my warn message...");
   ```

   오류 메시지

   ```
   logger.error("This is my error message...");
   ```

   치명적 오류 메시지

   ```
   logger.fatal("This is my fatal error message...");
   ```

   고객에게 치명적인 오류 메시지 발생

   ```
   logger.cxfatal("This is my customer experience fatal error message...");
   ```

스크립트의 로깅 문의 형식의 예는 Realtime 서버 스크립트 예제을 참조하세요.

로그 파일의 출력은 예제 로그의 다음 줄에 표시된 것처럼 메시지 유형 (DEBUGINFOWARNERRORFATAL,,,,,CXFATAL)을 나타냅니다.

```
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.
```

서버 로그 액세스

게임 세션이 종료되면 Amazon은 GameLift 자동으로 로그를 Amazon S3에 저장하고 14일 동안 보관합니다. GetGameSessionLogUrl API 호출을 사용하여 게임 세션의 로그 위치를 가져올 수 있습니다. API 호출에서 반환된 URL을 사용하여 로그를 다운로드합니다.

로깅 수준 조정

로그는 매우 빠르게 증가하여 크기 제한을 초과할 수 있습니다. 로그를 모니터링하고 로그 출력을 필요한 메시지로만 제한해야 합니다. Realtime 서버의 경우 플릿의 런타임 구성에 `loggingLevel:`*`LOGGING_LEVEL`* 형식으로 파라미터를 제공하여 로깅 수준을 조정할 수 있습니다. 여기에서 *`LOGGING_LEVEL`*은 다음 값 중 하나입니다.

1. `debug`
2. `info`(기본값)
3. `warn`
4. `error`
5. `fatal`
6. `cxfatal`

이 목록은 가장 심각하지 않은 상태(`debug`)에서 가장 심각한 상태(`cxfatal`) 순으로 정렬됩니다. 단일 `loggingLevel`을 설정하면 서버는 해당 심각도 수준 또는 더 높은 심각도 수준에서만 메시지를 기록합니다. 예를 들어, `loggingLevel:error`의 설정을 통해 플릿의 모든 서버가 로그에 `error`, `fatal`, `cxfatal` 메시지만 기록하도록 할 수 있습니다.

플릿을 생성할 때 및 그 이후에 플릿에 대한 로깅 수준을 설정할 수 있습니다. 플릿이 실행된 후 플릿의 로깅 수준을 변경하면 업데이트 이후에 생성된 게임 세션의 로그에만 영향을 미칩니다. 기존 게임 세션의 로그는 영향을 받지 않습니다. 플릿을 생성할 때 로깅 수준을 설정하지 않으면 서버에서 로깅 수준을 기본값(`info`)으로 설정합니다. 로깅 수준 설정에 대한 지침은 다음 섹션을 참조하세요.

Realtime 서버 플릿 생성 시 로깅 수준 설정(콘솔)

Amazon GameLift 관리형 플릿 생성의 지침에 따라 플릿을 생성하고 다음을 추가합니다.

- 프로세스 관리 단계의 서버 프로세스 할당 하위 단계에서 로깅 수준 키-값 쌍(예:`loggingLevel:error`)을 시작 파라미터 값으로 제공합니다. 영숫자가 아닌 문자(쉼표 제외)를 사용하여 로깅 수준을 추가 파라미터(예: `loggingLevel:error +map Winter444`)와 구분합니다.

Realtime 서버 플릿 생성 시 로깅 수준 설정(AWS CLI)

Amazon GameLift 관리형 플릿 생성의 지침에 따라 플릿을 생성하고 다음을 추가합니다.

- create-fleet의 --runtime-configuration 파라미터 인수에서 로깅 수준 키-값 쌍(예:loggingLevel:error)을 Parameters의 값으로 제공합니다. 영숫자가 아닌 문자(쉼표 제외)를 사용하여 로깅 수준을 추가 파라미터와 구분합니다. 다음 예를 참조하세요.

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,
                    MaxConcurrentGameSessionActivations=2,
                    ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,
                                    Parameters=loggingLevel:error +map Winter444,
                                    ConcurrentExecutions=10}]"
```

Realtime 서버 플릿 실행 시 로깅 수준 설정(콘솔)

의 지침에 따라 Amazon GameLift Console을 사용하여 플릿을 업데이트하고 다음을 추가하십시오. 플릿 구성 업데이트

- 플릿 편집 페이지의 서버 프로세스 할당에서 로깅 수준 키-값 쌍(예:loggingLevel:error)을 시작 파라미터 값으로 제공합니다. 영숫자가 아닌 문자(쉼표 제외)를 사용하여 로깅 수준을 추가 파라미터(예: loggingLevel:error +map Winter444)와 구분합니다.

Realtime 서버 플릿 실행 시 로깅 수준 설정(AWS CLI)

플릿 구성 업데이트의 지침에 따라 AWS CLI를 사용하여 플릿을 업데이트하고 다음을 추가합니다.

- update-runtime-configuration의 --runtime-configuration 파라미터 인수에서 로깅 수준 키-값 쌍(예:loggingLevel:error)을 Parameters의 값으로 제공합니다. 영숫자가 아닌 문자(쉼표 제외)를 사용하여 로깅 수준을 추가 파라미터와 구분합니다. 다음 예를 참조하세요.

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,
                    MaxConcurrentGameSessionActivations=2,
                    ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,
                                    Parameters=loggingLevel:error +map Winter444,
                                    ConcurrentExecutions=10}]"
```

아마존의 보안 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

클라우드 보안은 최우선 과제입니다. AWS AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 사용자와 사용자 AWS 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. GameLiftAmazon에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내 서비스 규정 준수 참조하십시오.
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 회사의 요구 사항, 해당 법률AWS 및 규정을 비롯한 기타 요인에 대한 책임도 귀하에게 있습니다.

이 설명서는 Amazon을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 GameLift 됩니다. 다음 주제는 보안 및 규정 준수 목표를 GameLift 충족하도록 Amazon을 구성하는 방법을 보여줍니다. 또한 Amazon GameLift 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- 아마존에서의 데이터 보호 GameLift
- Amazon의 자격 증명 및 액세스 관리 GameLift
- Amazon GameLift를 사용한 로깅 및 모니터링
- Amazon에 대한 규정 준수 검증 GameLift
- 아마존의 레질리언스 GameLift
- 아마존의 인프라 보안 GameLift
- Amazon의 구성 및 취약성 분석 GameLift
- Amazon의 보안 모범 사례 GameLift

아마존에서의 데이터 보호 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

AWS 공동 책임 모델 Amazon의 데이터 보호에 적용됩니다 GameLift. 이 모델에 설명된 대로 AWS 은(는) 모두를 실행하는 글로벌 인프라를 보호할 책임이 AWS 클라우드있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon GameLift 또는 다른 곳에서 콘솔 AWS CLI, API 또는 AWS SDK를 AWS 서비스 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함해서는 안 됩니다.

Amazon GameLift 관련 데이터는 다음과 같이 처리됩니다.

- Amazon에 업로드한 게임 서버 빌드와 GameLift 스크립트는 Amazon S3에 저장됩니다. 이 데이터는 업로드한 후에 고객이 직접 액세스할 수 없습니다. 권한 있는 사용자는 파일을 업로드하기 위한 임시 액세스 권한을 얻을 수 있지만 Amazon S3에서 직접 파일을 보거나 업데이트할 수는 없습니다. 스크립트와 빌드를 삭제하려면 Amazon GameLift 콘솔 또는 서비스 API를 사용합니다.
- 게임 세션 로그 데이터는 게임 세션이 완료된 후 일정 기간 동안 Amazon S3에 저장됩니다. 승인된 사용자는 Amazon GameLift 콘솔의 링크를 통해 다운로드하거나 서비스 API를 호출하여 로그 데이터에 액세스할 수 있습니다.
- 지표 및 이벤트 데이터는 Amazon에 저장되며 Amazon GameLift GameLift 콘솔을 통해 또는 서비스 API 호출을 통해 액세스할 수 있습니다. 플릿, 인스턴스, 게임 세션 배치, 매치메이킹 티켓, 게임 세션 및 플레이어 세션에서 데이터를 검색할 수 있습니다. Amazon CloudWatch 및 CloudWatch 이벤트를 통해서도 데이터에 액세스할 수 있습니다.
- 고객이 제공한 데이터는 Amazon에 저장됩니다. GameLift 권한 있는 사용자는 서비스 API 호출을 통해 액세스할 수 있습니다. 잠재적으로 민감한 데이터에는 플레이어 데이터, 플레이어 세션 및 게임 세션 데이터(연결 정보 포함), 자동 매치메이커 데이터 등이 포함될 수 있습니다.

> **Note**
>
> 요청에 사용자 지정 플레이어 ID를 제공하는 경우 이러한 값은 익명화된 UUID이며 플레이어 식별 정보를 포함하지 않아야 합니다.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

저장 시 암호화

Amazon GameLift 관련 데이터의 저장 중 암호화는 다음과 같이 처리됩니다.

- 게임 서버 빌드 및 스크립트는 서버 측 암호화를 사용하여 Amazon S3 버킷에 저장됩니다.
- 고객이 제공한 데이터는 암호화된 형식으로 GameLift Amazon에 저장됩니다.

전송 중 암호화

Amazon GameLift API에 대한 연결은 보안 (SSL) 연결을 통해 이루어지며 AWS 서명 버전 4를 사용하여 인증됩니다 (AWS CLI 또는 AWS SDK를 통해 연결하는 경우 서명은 자동으로 처리됨). 인증은 연결에 사용되는 보안 자격 증명에 대한 IAM 정의 액세스 정책을 사용하여 관리됩니다.

게임 클라이언트와 게임 서버 간의 직접 통신은 다음과 같습니다.

- Amazon GameLift 리소스에서 호스팅되는 사용자 지정 게임 서버의 경우 통신에는 Amazon GameLift 서비스가 포함되지 않습니다. 이 통신의 암호화는 고객의 책임입니다. TLS가 활성화된 플릿을 사용하면 게임 클라이언트가 연결 시 게임 서버를 인증하고 게임 클라이언트와 게임 서버 간의 모든 통신을 암호화할 수 있습니다.
- TLS 인증서 생성이 활성화된 Realtime 서버의 경우 Realtime Client SDK를 사용하는 게임 클라이언트와 Realtime 서버 간의 트래픽이 전송 중에 암호화됩니다. TCP 트래픽은 TLS 1.2를 사용하여 암호화되고, UDP 트래픽은 DTLS 1.2를 사용하여 암호화됩니다.

인터네트워크 트래픽 개인 정보

Amazon GameLift 인스턴스에 원격으로 안전하게 액세스할 수 있습니다. Linux를 사용하는 인스턴스의 경우 SSH가 원격 액세스를 위한 보안 통신 채널을 제공합니다. Windows를 실행하는 인스턴스의 경우 RDP(원격 데스크톱 프로토콜) 클라이언트를 사용합니다. Amazon GameLift FleetiQ를 사용하면 Systems AWS Manager 세션 관리자 및 명령 실행을 사용한 인스턴스에 대한 원격 액세스가 TLS 1.2를 사용하여 암호화되고, 연결 생성 요청은 SigV4를 사용하여 서명됩니다. 관리형 Amazon GameLift 인스턴스 연결에 대한 도움말은 을 참조하십시오Amazon GameLift 플릿 인스턴스에 원격으로 연결.

Amazon의 자격 증명 및 액세스 관리 GameLift

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 Amazon GameLift 리소스를 사용하기 위해 인증 (로그인) 및 권한 부여 (권한 보유) 할 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- 고객
- 보안 인증을 통한 인증
- 정책을 사용한 액세스 관리
- 아마존이 IAM과 협력하는 GameLift 방식
- Amazon의 ID 기반 정책 예제 GameLift
- Amazon GameLift 자격 증명 및 액세스 문제 해결

고객

사용 방법 AWS Identity and Access Management (IAM) 은 Amazon에서 수행하는 작업에 따라 다릅니다. GameLift

서비스 사용자 — Amazon GameLift 서비스를 사용하여 업무를 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 Amazon GameLift 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. GameLiftAmazon에서 기능에 액세스할 수 없는 경우 을 참조하십시오Amazon GameLift 자격 증명 및 액세스 문제 해결.

서비스 관리자 — 회사에서 Amazon GameLift 리소스를 담당하는 경우 Amazon에 대한 전체 액세스 권한이 있을 것입니다 GameLift. 서비스 사용자가 액세스해야 하는 Amazon GameLift 기능 및 리소스를 결정하는 것은 귀하의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사에서 GameLift Amazon에서 IAM을 사용하는 방법에 대한 자세한 내용은 을 참조하십시오아마존이 IAM과 협력하는 GameLift 방식.

IAM 관리자 — IAM 관리자인 경우 Amazon에 대한 액세스를 관리하는 정책을 작성하는 방법에 대해 자세히 알아보는 것이 좋습니다. GameLift IAM에서 사용할 수 있는 Amazon GameLift 자격 증명 기반 정책의 예를 보려면 을 참조하십시오. Amazon의 ID 기반 정책 예제 GameLift

보안 인증을 통한 인증

인증은 ID 보안 인증을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자나 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

보안 인증 소스를 통해 제공된 보안 인증 정보를 사용하여 페더레이션형 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증이 페더레이션형 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS

도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은(는) 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

AWS 계정를 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대해 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 보안 인증은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에는 루트 사용자를 가급적 사용하지 않는 것이 좋습니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 이 정보를 사용합니다. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 안내서의 루트 사용자 보안 인증이 필요한 태스크 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 보안 인증 공급자와의 페더레이션을 사용하여 임시 보안 인증 정보를 사용하여 AWS 서비스에 액세스하도록 요구합니다.

페더레이션 ID는 엔터프라이즈 사용자 디렉터리, 웹 보안 인증 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 보안 인증 소스를 통해 제공된 보안 인증을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 인증은 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 보안 인증을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한 보안 인증 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까? 섹션을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 보안 인증입니다. 가능하다면 암호 및 액세스 키와 같은 장기 보안 인증 정보가 있는 IAM 사용자를 생성하는 대신, 임시 보안 인증 정보를 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 보안 인증 정보가 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체 섹션을 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 보안 인증입니다. 귀하는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님) 섹션을 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 태스크를 직접적으로 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용를 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션형 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션형 ID가 인증되면 이 ID는 역할과 연결되며 역할에 의해 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기 부분을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 보안 인증 정보에서 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연결합니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트 섹션을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 역할을(프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이 섹션을 참조하세요.
- 교차 서비스 액세스 – 일부 AWS 서비스은(는) 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 직접적으로 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.

- 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하십시오.
- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
- 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여 섹션을 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 AWSID 또는 리소스에 연결하여 AWS내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS은(는) 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되는지 또는 거부되는지를 결정합니다. 대부분의 정책은 AWS에 JSON 설명서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스이(가) 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 유형

AWS은(는) 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계 섹션을 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations은(는) 기업이 소유하는 여러 개의 AWS 계정을(를) 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 계정 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지 여부를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

아마존이 IAM과 협력하는 GameLift 방식

IAM을 사용하여 Amazon에 대한 액세스를 관리하기 전에 Amazon에서 GameLift 사용할 수 있는 IAM 기능에 대해 알아보십시오. GameLift

Amazon에서 사용할 수 있는 IAM 기능 GameLift

IAM 특성	아마존 GameLift 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
보안 주체 권한	예
Service roles(서비스 역할)	예
Service-linked roles(서비스 연결 역할)	아니요

Amazon GameLift 및 기타 AWS 서비스가 대부분의 IAM 기능을 어떻게 사용하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 연동되는 AWS 서비스를 참조하십시오.

Amazon을 위한 ID 기반 정책 GameLift

자격 증명 기반 정책 지원	예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon의 ID 기반 정책 예제 GameLift

Amazon GameLift ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon의 ID 기반 정책 예제 GameLift

Amazon 내 리소스 기반 정책 GameLift

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스이(가) 포함될 수 있습니다.

크로스 계정 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 신뢰 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야 합니다. 엔터티에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

아마존에 대한 정책 조치 GameLift

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWS API 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함합니다.

Amazon GameLift 작업 목록은 서비스 승인 GameLift 참조에서 Amazon이 정의한 작업을 참조하십시오.

Amazon의 정책 조치는 조치 앞에 다음 접두사를 GameLift 사용합니다.

```
gamelift
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "gamelift:action1",
      "gamelift:action2"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe(이)라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "gamelift:Describe*"
```

Amazon GameLift ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon의 ID 기반 정책 예제 GameLift

아마존용 정책 리소스 GameLift

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문장에는 Resource 또는 NotResource 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon GameLift 리소스 유형 및 해당 ARN 목록은 서비스 인증 참조의 GameLift Amazon이 정의한 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업에 대해 알아보려면 Amazon에서 정의한 작업을 참조하십시오. GameLift

일부 Amazon GameLift 리소스에는 ARN 값이 있어 IAM 정책을 사용하여 리소스의 액세스를 관리할 수 있습니다. Amazon GameLift 플릿 리소스에는 다음 구문의 ARN이 있습니다.

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

ARN 형식에 대한 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

예를 들어, 명령문에 fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa 플릿을 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

특정 계정에 속하는 모든 플릿을 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

Amazon GameLift ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon의 ID 기반 정책 예제 GameLift

Amazon용 정책 조건 키 GameLift

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 선택 사항입니다. 같음이나 미만 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우 AWS는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 `OR` 태스크를 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS은(는) 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon GameLift 조건 키 목록은 서비스 인증 GameLift 참조의 Amazon용 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 Amazon에서 정의한 작업을 참조하십시오 GameLift.

Amazon GameLift ID 기반 정책의 예를 보려면 을 참조하십시오. Amazon의 ID 기반 정책 예제 GameLift

아마존의 ACL GameLift

ACL 지원	아니요

ACL(액세스 제어 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

아마존과 함께하는 ABAC GameLift

ABAC 지원(정책의 태그)	예

ABAC(속성 기반 액세스 제어)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 엔터티 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC란 무엇입니까?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

해당 리소스 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예는 태그를 기반으로 Amazon GameLift 플릿 보기 섹션을 참조하세요.

Amazon에서 임시 자격 증명 사용 GameLift

임시 보안 인증 지원	예

일부 AWS 서비스은(는) 임시 보안 인증을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM을 사용하는 AWS 서비스을(를) 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 보안 인증을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 보안 인증을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 보안 인증을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 보안 인증을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 보안 인증을 동적으로 생성할 것을 권장합니다. 자세한 내용은 IAM의 임시 보안 자격 증명을(를) 참조하세요.

Amazon에 대한 크로스 서비스 주체 권한 GameLift

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하십시오.

아마존의 서비스 역할 GameLift

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon GameLift 기능이 손상될 수 있습니다. Amazon에서 지침을 GameLift 제공하는 경우에만 서비스 역할을 편집하십시오.

Amazon GameLift 호스팅 게임 서버가 AWS Lambda 함수 또는 Amazon DynamoDB 데이터베이스와 같은 다른 AWS 리소스에 액세스할 수 있도록 허용합니다. 게임 서버는 Amazon이 GameLift 관리하는 플릿에서 호스팅되므로 Amazon이 다른 AWS 리소스에 GameLift 제한적으로 액세스할 수 있도록 하는 서비스 역할이 필요합니다. 자세한 설명은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

Amazon의 서비스 연결 역할 GameLift

서비스 연결 역할 지원	아니요

서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon의 ID 기반 정책 예제 GameLift

기본적으로 사용자와 역할에는 Amazon GameLift 리소스를 생성하거나 수정할 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형의 ARN 형식을 비롯하여 GameLift Amazon에서 정의하는 작업 및 리소스 유형에 대한 자세한 내용은 서비스 인증 참조의 GameLift Amazon용 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- 아마존 GameLift 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 게임 세션을 위한 플레이어 액세스 허용
- Amazon GameLift 대기열 하나에 대한 액세스 허용
- 태그를 기반으로 Amazon GameLift 플릿 보기
- Amazon S3의 게임 빌드 파일에 액세스

정책 모범 사례

ID 기반 정책은 누군가가 사용자 계정에서 Amazon GameLift 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- Apply least-privilege permissions(최소 권한 적용) – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 least-privilege permissions(최소 권한)으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- Use conditions in IAM policies to further restrict access(IAM 정책의 조건을 사용하여 액세스 추가 제한) – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 생성할 수 있습니다. 특정 AWS 서비스(예: AWS CloudFormation)를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 생성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- Require multi-factor authentication(MFA) (다중 인증 필요) – AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 호출할 때 MFA를 요구하려면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

아마존 GameLift 콘솔 사용

Amazon GameLift 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 내 Amazon GameLift 리소스를 나열하고 세부 정보를 볼 수 있어야 AWS 계정 합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

해당 개체가 Amazon GameLift 콘솔을 계속 사용할 수 있도록 하려면 다음 예제와 의 구문을 사용하여 사용자 및 그룹에 권한을 추가하십시오관리자 권한 예제. 자세한 설명은 Amazon의 사용자 권한 관리 GameLift 섹션을 참조하세요.

AWS CLI또는 AWS API 작업을 GameLift 통해 Amazon을 사용하는 사용자에게는 최소 콘솔 권한이 필요하지 않습니다. 대신 사용자가 수행해야 하는 작업에만 액세스를 제한할 수 있습니다. 예를 들어 게임 클라이언트를 대신하여 행동하는 플레이어 사용자는 게임 세션을 요청하고 플레이어를 게임에 참여시키는 등의 작업을 수행하기 위한 액세스 권한이 필요합니다.

모든 Amazon GameLift 콘솔 기능을 사용하는 데 필요한 권한에 대한 자세한 내용은 에서 관리자를 위한 권한 구문을 참조하십시오관리자 권한 예제.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 보안 인증에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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": "*"
        }
    ]
}
```

게임 세션을 위한 플레이어 액세스 허용

플레이어를 게임 세션에 참여시키려면 게임 클라이언트와 백엔드 서비스에 권한이 필요합니다. 이러한 시나리오의 정책 예는 플레이어 사용자 권한 예제 섹션을 참조하세요.

Amazon GameLift 대기열 하나에 대한 액세스 허용

다음 예는 사용자에게 특정 Amazon GameLift 대기열에 대한 액세스 권한을 제공합니다.

이 정책은 사용자에게 다음 작업을 통해 대기열 대상을 추가, 업데이트 및 삭제할 권한을 부여합니다. `gamelift:UpdateGameSessionQueue`, `gamelift:DeleteGameSessionQueue`, `gamelift:DescribeGameSessionQueues`. 표시된 것처럼 이 정책은 `Resource` 요소를 사용하여 단일 대기열 `gamesessionqueue/examplequeue123`로 액세스를 제한합니다.

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

태그를 기반으로 Amazon GameLift 플릿 보기

ID 기반 정책의 조건을 사용하여 태그를 기반으로 Amazon GameLift 리소스에 대한 액세스를 제어할 수 있습니다. 이 예제는 `Owner` 태그가 사용자의 사용자 이름과 일치하는 경우 플릿을 볼 수 있는 정책을 생성하는 방법에 대해 보여 줍니다. 이 정책은 콘솔에서 이 작업을 완료하는 데 필요한 권한도 부여합니다.

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

Amazon S3의 게임 빌드 파일에 액세스

게임 서버를 Amazon과 통합한 후 빌드 파일을 Amazon GameLift S3에 업로드합니다. Amazon이 빌드 파일에 GameLift 액세스하도록 하려면 다음 정책을 사용하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": "arn:aws:s3:::bucket-name/object-name"
        }
    ]
}
```

Amazon GameLift 게임 파일 업로드에 대한 자세한 내용은 을 참조하십시오Amazon GameLift에 사용자 지정 서버 빌드 업로드.

Amazon GameLift 자격 증명 및 액세스 문제 해결

다음 정보를 사용하면 Amazon And AWS Identity and Access Management (IAM) 를 사용할 때 발생할 수 있는 일반적인 문제를 GameLift 진단하고 해결하는 데 도움이 됩니다.

주제

- Amazon에서 작업을 수행할 권한이 없습니다. GameLift
- IAM을 수행할 권한이 없습니다. PassRole
- 외부 사용자가 내 Amazon GameLift 리소스에 액세스할 수 있도록 AWS 계정 허용하고 싶습니다.

Amazon에서 작업을 수행할 권한이 없습니다. GameLift

AWS Management Console에서 작업을 수행할 권한이 없다는 메시지가 나타나는 경우 AWS 계정 관리자에게 지원을 요청합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 대기열에 대한 세부 정보를 보려고 하지만 `gamelift:DescribeGameSessionQueues` 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

이 경우 Mateo는 `gamelift:DescribeGameSessionQueues` 작업을 사용하여 `examplequeue123` 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류 메시지가 표시되는 경우 Amazon에 역할을 넘길 수 있도록 정책을 업데이트해야 GameLift 합니다. `iam:PassRole`

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 이라는 IAM 사용자가 GameLift Amazon에서 콘솔을 사용하여 작업을 `marymajor` 수행하려고 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 보안 인증을 제공한 사용자입니다.

외부 사용자가 내 Amazon GameLift 리소스에 액세스할 수 있도록 AWS 계정 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- Amazon에서 이러한 기능을 GameLift 지원하는지 알아보려면 을 참조하십시오아마존이 IAM과 협력하는 GameLift 방식.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하십시오.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- 자격 증명 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 페더레이션)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

Amazon GameLift를 사용한 로깅 및 모니터링

모니터링은 Amazon GameLift와 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 발생하는 다중 지점 실패를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.

AWS 및 Amazon GameLift는 게임 호스팅 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 다양한 도구를 제공합니다.

Amazon CloudWatch 경보

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시합니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책으로 알림이 전송됩니다. CloudWatch 경보는 특정 상태가 아니라 상태 변경에 의해 트리거되고 지정된 수의 기간 동안 유지됩니다. 자세한 내용은 Amazon CloudWatch를 사용한 Amazon GameLift 모니터링 섹션을 참조하세요.

AWS CloudTrail 로그

CloudTrail은 Amazon GameLift에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 기록을 제공합니다. CloudTrail에서 수집한 정보를 사용하여 Amazon GameLift에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail를 사용하여 Amazon GameLift API 호출 로깅 섹션을 참조하세요.

Amazon에 대한 규정 준수 검증 GameLift

GameLift Amazon은 AWS 규정 준수 프로그램의 범위에 포함되지 않습니다.

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램 범위 내 규정 준수AWS 서비스 프로그램별 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.

- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

아마존의 레질리언스 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

AWS 글로벌 인프라는 지역 및 가용 영역을 중심으로 구축됩니다. AWS AWS 지역은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이러한 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복

조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

AWS 글로벌 인프라 외에도 GameLift Amazon은 데이터 복원력 요구 사항을 지원하는 데 도움이 되는 다음과 같은 기능을 제공합니다.

- 다중 지역 대기열 — Amazon GameLift 게임 세션 대기열은 사용 가능한 호스팅 리소스가 있는 새 게임 세션을 배치하는 데 사용됩니다. 여러 리전에 걸쳐 있는 대기열은 한 리전이 중단될 경우 게임 세션 배치를 리디렉션할 수 있습니다. 게임 세션 대기열 생성에 대한 자세한 내용과 모범 사례는 게임 세션 대기열 설계 섹션을 참조하세요.
- 자동 용량 조정 — Amazon GameLift 조정 도구를 사용하여 호스팅 리소스의 상태와 가용성을 유지합니다. 이러한 도구는 게임 및 플레이어의 요구에 맞게 플릿 용량을 조정할 수 있는 다양한 옵션을 제공합니다. 조정에 대한 자세한 내용은 Amazon GameLift 호스팅 용량 확장 섹션을 참조하세요.
- 인스턴스 간 분산 — Amazon은 플릿 크기에 따라 들어오는 트래픽을 여러 인스턴스에 GameLift 분산합니다. 인스턴스가 비정상이거나 응답하지 않는 경우 프로덕션 게임의 가용성을 유지하기 위해 인스턴스가 여러 개 있어야 합니다.
- Amazon S3 스토리지 — Amazon에 업로드된 게임 서버 빌드와 스크립트는 여러 데이터 센터 복제를 사용하여 복원력을 높이는 표준 스토리지 클래스를 사용하여 Amazon S3에 저장됩니다. GameLift 게임 세션 로그도 스탠다드 스토리지 클래스를 사용하여 Amazon S3에 저장됩니다.

아마존의 인프라 보안 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

GameLift Amazon은 관리형 서비스로서 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차에 따라 보호됩니다.

AWS 게시된 API 호출을 사용하여 네트워크를 GameLift 통해 Amazon에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

Amazon GameLift 서비스는 모든 플릿을 Amazon 가상 사설 클라우드 (VPC) 에 배치하여 각 플릿이 클라우드의 논리적으로 격리된 영역에 존재하도록 AWS 합니다. Amazon GameLift 정책을 사용하여 특정 VPC 엔드포인트 또는 특정 VPC에서의 액세스를 제어할 수 있습니다. 이렇게 하면 네트워크 내의 AWS 특정 VPC로부터 특정 Amazon GameLift 리소스에 대한 네트워크 액세스를 효과적으로 분리할 수 있습니다. 플릿을 만들 때 포트 번호 및 IP 주소의 범위를 지정합니다. 이러한 범위는 인바운드 트래픽이 플릿 VPC에서 호스팅된 게임 서버에 액세스하는 방법을 제한합니다. 플릿 액세스 설정을 선택할 때 표준 보안 모범 사례를 사용합니다.

Amazon의 구성 및 취약성 분석 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

구성 및 IT 제어는 AWS 와 고객 간의 공동 책임입니다. 자세한 내용은 공동 책임 모델을 참조하십시오. AWS AWS 게스트 운영 체제 (OS) 및 데이터베이스 패치, 방화벽 구성, 재해 복구와 같은 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하세요. Amazon Web Services: 보안 프로세스 개요(백서).

다음 보안 모범 사례는 GameLift Amazon의 구성 및 취약성 분석도 다룹니다.

- 고객은 게임 호스팅을 위해 Amazon GameLift 인스턴스에 배포되는 소프트웨어를 관리할 책임이 있습니다. 구체적으로 설명하면 다음과 같습니다.
  - 업데이트 및 보안 패치를 포함하여 고객이 제공한 게임 서버 애플리케이션 소프트웨어를 유지 관리해야 합니다. 게임 서버 소프트웨어를 업데이트하려면 새 빌드를 GameLift Amazon에 업로드하고 새 플릿을 생성한 다음 트래픽을 새 플릿으로 리디렉션하십시오.
  - 운영 체제를 포함하는 기본 Amazon Machine Image(AMI) 는 새 플릿이 생성될 때만 업데이트됩니다. 운영 체제 및 AMI의 일부인 기타 애플리케이션을 패치, 업데이트 및 보호하려면 게임 서버 업데이트에 관계없이 정기적으로 플릿을 재활용합니다.
- 고객은 SDK, Amazon Server SDK, 실시간 GameLift 서버용 Amazon GameLift 클라이언트 AWS SDK를 비롯한 최신 SDK 버전으로 게임을 정기적으로 업데이트하는 것을 고려해야 합니다.

Amazon의 보안 모범 사례 GameLift

Amazon GameLift FleetiQ를 Amazon EC2에서 독립형 기능으로 사용하는 경우, Amazon EC2 사용 설명서의 Amazon EC2의 보안을 참조하십시오.

GameLift Amazon은 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용하십시오.

인터넷 포트를 열지 마십시오.

인터넷 포트를 열면 보안상 위험이 있으므로 인터넷 포트를 열지 않는 것이 좋습니다. 예를 들어 다음과 같이 원격 데스크톱 포트를 여는 UpdateFleetPortSettings데 사용하는 경우

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
      {
        "FromPort": 3389,
        "IpRange": "0.0.0.0/0",
        "Protocol": "RDP",
        "ToPort": 3389
      }
  ]
}
```

그러면 인터넷상의 모든 사용자가 인스턴스에 액세스할 수 있게 됩니다.

대신 특정 IP 주소 또는 주소 범위로 포트를 여십시오. 예를 들면 다음과 같습니다.

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
      {
        "FromPort": 3389,
        "IpRange": "54.186.139.221/32",
        "Protocol": "TCP",
        "ToPort": 3389
      }
  ]
}
```

자세히 알아보기

Amazon을 보다 안전하게 사용할 수 있는 방법에 대한 GameLift 자세한 내용은 AWS Well-Architected Tool 보안 원칙을 참조하십시오. .

Amazon GameLift 참조 가이드

이 섹션에는 Amazon GameLift를 사용하기 위한 참조 문서가 포함되어 있습니다.

주제

- Amazon GameLift Service API 참조(AWS SDK)
- Amazon GameLift Realtime 서버 참조
- Amazon GameLift Server SDK 참조
- 게임 세션 배치 이벤트

Amazon GameLift Service API 참조(AWS SDK)

이 주제에서는 사용자 지정 게임 서버 및 Realtime 서버 호스팅을 포함하여 Amazon GameLift 관리형 호스팅 솔루션과 함께 사용할 수 있는 작업 기반 API 작업 목록을 제공합니다. 이러한 작업은 `aws.gamelift` 네임스페이스의 AWS SDK에 패키징됩니다. AWS SDK를 다운로드하거나 Amazon GameLift API 참조 문서를 확인하세요.

API에는 관리형 게임 호스팅을 위한 두 가지 작업 세트가 포함되어 있습니다.

- Amazon GameLift 호스팅 리소스 설정 및 관리
- 게임 세션 시작 및 플레이어 참여

Amazon GameLift 서비스 API에는 다른 Amazon GameLift 도구 및 솔루션과 함께 사용할 수 있는 작업도 포함되어 있습니다. FleetIQ API 목록에 대해서는 FleetIQ API 작업을 참조하세요. 매치메이킹을 위한 FlexMatch API 목록에 대해서는 FlexMatch API 작업을 참조하세요.

Amazon GameLift 호스팅 리소스 설정 및 관리

이러한 작업을 호출하여 게임 서버의 호스팅 리소스 구성, 플레이어 수요에 맞게 용량 조정, 성능 및 사용률 지표에 액세스 등을 수행할 수 있습니다. 이러한 API 작업은 Realtime 서버를 포함하여 Amazon GameLift에서 호스팅되는 게임 서버에서 사용됩니다. 대부분의 리소스 관리 작업에 Amazon GameLift 콘솔을 사용하거나 AWS Command Line Interface(AWS CLI) 도구 또는 AWS SDK를 사용하여 서비스를 호출할 수 있습니다.

배포를 위한 게임 서버 준비

호스팅 리소스에서 배포 및 출시를 준비하기 위해 게임의 게임 서버 코드를 업로드하고 구성합니다.

사용자 지정 게임 서버 빌드 관리

- upload-build - 로컬 경로에서 빌드 파일을 업로드하고 새 Amazon GameLift 빌드 리소스를 생성합니다. AWS CLI 명령으로만 사용할 수 있는 이 작업은 게임 서버 빌드를 업로드하는 가장 일반적인 방법입니다.
- CreateBuild - Amazon S3 버킷에 저장된 파일을 사용하여 새 빌드를 생성합니다.
- ListBuilds - Amazon GameLift 리전에 업로드된 모든 빌드의 목록을 가져옵니다.
- DescribeBuild - 빌드와 관련된 정보를 검색합니다.
- UpdateBuild - 빌드 이름 및 버전을 포함하여 빌드 메타데이터를 변경합니다.
- DeleteBuild - Amazon GameLift에서 빌드를 제거합니다.

Realtime 서버 구성 스크립트 관리

- CreateScript - JavaScript 파일을 업로드하고 새 Amazon GameLift 스크립트 리소스를 생성합니다.
- ListScripts - Amazon GameLift 리전에 업로드된 모든 Realtime 스크립트의 목록을 가져옵니다.
- DescribeScript - Realtime 스크립트와 관련된 정보를 검색합니다.
- UpdateScript - 스크립트 메타데이터를 변경하고 수정된 스크립트 콘텐츠를 업로드합니다.
- DeleteScript - Amazon GameLift에서 Realtime 스크립트를 제거합니다.

호스팅용 컴퓨팅 리소스 설정

호스팅 리소스를 구성하고 게임 서버 빌드 또는 Realtime 구성 스크립트를 배포합니다.

플릿 생성 및 관리

- CreateFleet - 게임 서버를 실행할 새로운 Amazon GameLift 컴퓨팅 리소스 플릿을 구성하고 배포합니다. 일단 배포되면 게임 서버가 구성된 대로 자동으로 시작되고 게임 세션을 호스팅할 준비가 됩니다.
- ListFleets - Amazon GameLift 리전에 있는 모든 플릿의 목록을 가져옵니다.
- DeleteFleet - 더 이상 게임 서버나 호스팅 플레이어를 실행하지 않는 플릿을 종료합니다.
- 플릿 위치 보기/업데이트.
  - CreateFleetLocations - 여러 위치를 지원하는 기존 플릿에 원격 위치를 추가합니다.
  - DescribeFleetLocationAttributes - 플릿의 모든 원격 위치 목록을 가져오고 각 위치의 현재 상태를 확인합니다.

  - DeleteFleetLocations - 여러 위치를 지원하는 플릿에서 원격 위치를 제거합니다.
- 플릿 구성 보기/업데이트.
  - DescribeFleetAttributes / UpdateFleetAttributes - 게임 세션 보호와 리소스 생성 제한에 대한 플릿의 메타데이터 및 설정을 보거나 변경합니다.
  - DescribeFleetPortSettings / UpdateFleetPortSettings - 플릿에 허용되는 인바운드 권한(IP 주소 및 포트 설정 범위)을 보거나 변경합니다.
  - DescribeRuntimeConfiguration / UpdateRuntimeConfiguration - 플릿의 각 인스턴스에서 실행할 서버 프로세스 및 개수를 보거나 변경합니다.

플릿 용량 관리

- DescribeEC2InstanceLimits - 현재 AWS 계정과 현재 사용 수준에 허용되는 최대 인스턴스 수를 검색합니다.
- DescribeFleetCapacity - 플릿의 홈 리전에 대한 현재 용량 설정을 검색합니다.
- DescribeFleetLocationCapacity - 다중 위치 플릿의 각 위치에 대한 현재 용량 설정을 검색합니다.
- UpdateFleetCapacity - 수동으로 플릿 용량 설정을 조정합니다.
- Auto Scaling 설정:
  - PutScalingPolicy - 대상 기반 자동 크기 조정을 설정하거나, 사용자 지정 자동 크기 조정 정책을 생성하거나, 기존 정책을 업데이트합니다.
  - DescribeScalingPolicies - 기존의 자동 크기 조정 정책을 검색합니다.
  - DeleteScalingPolicy - 자동 크기 조정 정책을 삭제하고 이 정책이 플릿 용량에 영향을 미치는 것을 중지시킵니다.
  - StartFleetActions - 플릿의 자동 크기 조정 정책을 다시 시작합니다.
  - StopFleetActions - 플릿의 자동 크기 조정 정책을 일시 중단합니다.

플릿 활동 모니터링.

- DescribeFleetUtilization - 현재 플릿에서 활성 상태인 서버 프로세스, 게임 세션 및 플레이어의 수에 대한 통계를 검색합니다.
- DescribeFleetLocationUtilization - 다중 위치 플릿의 각 위치에 대한 사용률 통계를 검색합니다.
- DescribeFleetEvents - 지정된 시간 범위 동안 플릿에 대해 기록된 이벤트를 봅니다.
- DescribeGameSessions - 게임의 실행 시간 및 현재 플레이어 수를 포함하여 게임 세션 메타데이터를 검색합니다.

최적의 게임 세션 배치를 위한 대기열 설정

비용, 지연 시간 및 복원력을 고려하여 사용 가능한 최상의 호스팅 리소스와 함께 게임 세션을 배치하기 위해 다중 플릿, 다중 리전 대기열을 설정합니다.

- CreateGameSessionQueue - 게임 세션 배치 요청을 처리할 때 사용하기 위한 대기열을 만듭니다.
- DescribeGameSessionQueues - Amazon GameLift 리전에 정의된 게임 세션 대기열을 검색합니다.
- UpdateGameSessionQueue - 게임 세션 대기열의 구성을 변경합니다.
- DeleteGameSessionQueue - 리전에서 게임 세션 대기열을 제거합니다.

별칭 관리

별칭을 사용하여 플릿을 나타내거나 터미널 대체 대상을 만듭니다. 별칭은 예를 들어 게임 서버 빌드 업데이트 중에 한 플릿에서 다른 플릿으로 게임 활동을 전환할 때 유용합니다.

- CreateAlias - 새 별칭을 정의하고 선택적으로 플릿에 할당합니다.
- ListAliases - Amazon GameLift 리전에 정의된 모든 플릿 별칭을 가져옵니다.
- DescribeAlias - 기존 별칭에 대한 정보를 검색합니다.
- UpdateAlias - 한 플릿에서 다른 플릿으로 리디렉션하는 등 별칭 설정을 변경합니다.
- DeleteAlias - 리전에서 별칭을 제거합니다.
- ResolveAlias - 지정된 별칭이 가리키는 플릿 ID를 가져옵니다.

호스팅 인스턴스 액세스

플릿의 개별 인스턴스에 대한 정보를 보거나 문제 해결을 위해 지정된 플릿 인스턴스에 대한 원격 액세스를 요청합니다.

- DescribeInstances - 인스턴스 ID, IP 주소, 위치, 상태를 포함하여, 플릿의 각 인스턴스에 대한 정보를 가져옵니다.
- GetInstanceAccess - 플릿의 지정된 인스턴스에 원격으로 연결하는 데 필요한 액세스 자격 증명을 요청합니다.

VPC 피어링 설정

Amazon GameLift 호스팅 리소스와 기타 AWS 리소스 간의 VPC 피어링 연결을 생성하고 관리합니다.

- CreateVpcPeeringAuthorization - 사용자의 한 VPC에 피어링 연결에 대한 권한을 부여합니다.
- DescribeVpcPeeringAuthorizations - 유효한 피어링 연결 권한 부여를 검색합니다.
- DeleteVpcPeeringAuthorization - 피어링 연결 권한 부여를 삭제합니다.
- CreateVpcPeeringConnection - Amazon GameLift 플릿에 대한 VPC와 사용자의 한 VPC 사이에 피어링 연결을 설정합니다.
- DescribeVpcPeeringConnections - Amazon GameLift 플릿과의 활성 또는 보류 중인 VPC 피어링 연결에 대한 정보를 검색합니다.
- DeleteVpcPeeringConnection - Amazon GameLift 플릿과의 VPC 피어링 연결을 삭제합니다.

게임 세션 시작 및 플레이어 참여

게임 클라이언트 서비스에서 이러한 작업을 호출하여 새 게임 세션을 시작하고, 기존 게임 세션에 대한 정보를 얻으며, 플레이어를 게임 세션에 참여시킵니다. 이러한 작업은 Amazon GameLift에서 호스팅되는 사용자 지정 게임 서버 사용을 위한 것입니다. Realtime 서버를 사용하는 경우 Realtime Servers Client API(C#) 참조를 사용하여 게임 세션을 관리합니다.

- 하나 이상의 플레이어를 위한 새로운 게임 세션 시작.
  - StartGameSessionPlacement - Amazon GameLift에 문의하여 사용 가능한 최상의 호스팅 리소스를 찾고 새 게임 세션을 시작합니다. 새 게임 세션을 생성할 때는 이 방법을 사용하는 것이 좋습니다. 게임 세션 대기열을 사용하여 다중 리전의 호스팅 가용성을 추적하고 FleetIQ 알고리즘을 사용하여 플레이어 지연 시간, 호스팅 비용, 위치 등을 기반으로 배치 우선 순위를 지정합니다.
  - DescribeGameSessionPlacement - 배치 요청에 대한 세부 정보 및 상태를 가져옵니다.
  - StopGameSessionPlacement - 배치 요청을 취소합니다.
  - CreateGameSession - 특정 플릿 위치에서 비어 있는 새 게임 세션을 시작합니다. 이 작업을 통해 FleetIQ를 사용하여 배치 옵션을 평가하는 대신 게임 세션을 시작할 위치를 더 잘 제어할 수 있습니다. 별도의 단계를 거쳐 새 게임 세션에 플레이어를 추가해야 합니다.
- 플레이어를 기존 게임 세션에 배치. 사용 가능한 플레이어 슬롯이 있는 실행하는 게임 세션을 찾아서 새 플레이어를 위해 예약합니다.
  - CreatePlayerSession - 열린 슬롯 한 개를 게임 세션에 참여할 플레이어 한 명을 위해 예약합니다.
  - CreatePlayerSessions - 열린 슬롯 여러 개를 게임 세션에 참여할 플레이어 여러 명을 위해 예약합니다.
- 게임 세션 및 플레이어 세션 데이터로 작업. 게임 세션 및 플레이어 세션에 대한 정보를 관리합니다.
  - SearchGameSessions - 일련의 검색 기준에 따라 활성 게임 세션 목록을 요청합니다.

- DescribeGameSessions - 활성 상태인 시간 길이 및 현재 플레이어 수를 포함하여 특정 게임 세션에 대한 메타데이터를 검색합니다.
- DescribeGameSessionDetails - 하나 이상의 게임 세션에 대한 게임 세션 보호 설정을 포함한 메타데이터를 검색합니다.
- DescribePlayerSessions - 상태, 재생 시간 및 플레이어 데이터를 포함하여 플레이어 활동에 대한 세부 정보를 가져옵니다.
- UpdateGameSession - 최대 플레이어 수 및 참여 정책과 같은 게임 세션 설정을 변경합니다.
- GetGameSessionLogUrl - 게임 세션에 대해 저장된 로그의 위치를 가져옵니다.

Amazon GameLift Realtime 서버 참조

이 섹션에는 Amazon GameLift Realtime 서버 SDK에 대한 참조 문서가 포함되어 있습니다. Realtime Client API 및 Realtime 서버 스크립트 구성을 위한 지침을 제공합니다.

주제

- Realtime Servers Client API(C#) 참조
- Amazon GameLift Realtime 서버 스크립트 참조

Realtime Servers Client API(C#) 참조

Realtime Client API 참조를 사용하여 Amazon GameLift Realtime 서버에서 사용할 멀티플레이어 게임 클라이언트를 준비합니다. 통합 프로세스에 대한 자세한 내용은 Realtime 서버 준비 섹션을 참조하세요. 클라이언트 API에는 게임 클라이언트가 Realtime 서버에 연결하고 서버를 통해 메시지 및 데이터를 다른 게임 클라이언트와 교환하기 위해 사용할 수 있는 비동기 API 호출 및 비동기 콜백이 포함됩니다.

이 API는 다음 라이브러리에서 정의합니다.

Client.cs

- 동기식 작업
- 비동기 콜백
- 데이터 형식

Realtime Client API를 설정하려면

1. Amazon GameLift Realtime Client SDK를 다운로드합니다.
2. C# SDK 라이브러리를 빌드합니다. 솔루션 파일 `GameLiftRealtimeClientSdkNet45.sln`을 찾습니다. `README.md` 최소 요구 사항과 추가 빌드 옵션은 C# Server SDK의 파일을 참조하십시오. IDE에서 솔루션 파일을 로드합니다. SDK 라이브러리를 생성하려면 NuGet 패키지를 복원하고 솔루션을 빌드합니다.
3. Realtime Client 라이브러리를 게임 클라이언트 프로젝트에 추가합니다.

Realtime Servers Client API(C#) 참조: 작업

이 C# Realtime Client API 참조는 Amazon GameLift 플릿에 배포된 Realtime 서버에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 될 수 있습니다. 통합 프로세스에 대한 자세한 내용은 Realtime 서버 준비 섹션을 참조하세요.

- 동기식 작업
- 비동기 콜백
- 데이터 형식

Client()

새로운 클라이언트를 초기화하여 Realtime 서버와 통신하고 사용할 연결 유형을 식별합니다.

구문

```
public Client(ClientConfiguration configuration)
```

파라미터

clientConfiguration

클라이언트/서버 연결 유형을 지정하는 구성 세부 정보입니다. 이 파라미터 없이 Client()를 호출하도록 선택할 수 있습니다. 하지만 이 접근 방식을 선택하면 기본적으로 비보안 연결이 설정됩니다.

형식: ClientConfiguration

필수 항목 여부: 아니요

반환 값

Realtime 서버와 통신하는 데 사용하기 위한 Realtime 클라이언트의 인스턴스를 반환합니다

Connect()

게임 세션을 호스팅하고 있는 서버 프로세스에 연결을 요청합니다.

구문

```
public ConnectionStatus Connect(string endpoint, int remoteTcpPort, int listenPort,
 ConnectionToken token)
```

파라미터

endpoint

연결할 게임 세션의 DNS 이름 또는 IP 주소입니다. 엔드포인트는 `GameSession` 객체에서 지정되며, 이 객체는 AWS SDK Amazon GameLift API 작업 StartGameSessionPlacement, CreateGameSession 또는 DescribeGameSessions에 대한 클라이언트 호출에 응답하여 반환됩니다.

> **Note**
>
> TLS 인증서를 사용하는 플릿에서 Realtime 서버를 실행하는 경우 DNS 이름을 사용해야 합니다.

유형: 문자열

필수 항목 여부: 예

remoteTcpPort

게임 세션에 할당된 TCP 연결의 포트 번호입니다. 이 정보는 `GameSession` 객체에서 지정되며 이 객체는 StartGameSessionPlacement CreateGameSession 또는 DescribeGameSession 요청에 응답하여 반환됩니다.

유형: 정수

유효한 값: 1900 ~ 2000.

필수 항목 여부: 예

listenPort

게임 클라이언트가 UDP 채널을 사용하여 전송된 메시지를 수신 대기하는 포트 번호입니다.

유형: 정수

유효한 값: 33400 ~ 33500.

필수 항목 여부: 예

token

서버 프로세스에 요청하는 게임 클라이언트를 식별하는 선택적 정보입니다.

형식: ConnectionToken

필수 항목 여부: 예

반환 값

클라이언트의 연결 상태를 나타내는 ConnectionStatus 열거형 값을 반환합니다.

Disconnect()

게임 세션에 연결된 경우 게임 세션에서 게임 클라이언트의 연결을 해제합니다.

구문

```
public void Disconnect()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

이 메서드는 아무것도 반환하지 않습니다.

NewMessage()

지정된 작업 코드를 사용하여 새 메시지 객체를 생성합니다. 메시지 객체가 반환되면 대상을 지정하고 전송 방법을 업데이트한 다음 필요에 따라 데이터 페이로드를 추가하여 메시지 콘텐츠를 완료합니다. 완료되면 SendMessage()를 사용하여 메시지를 전송합니다.

구문

```
public RTMessage NewMessage(int opCode)
```

파라미터

opCode

플레이어 움직임 또는 서버 알림과 같은 게임 이벤트 또는 작업을 식별하는 개발자 정의 작업 코드입니다.

유형: 정수

필수 항목 여부: 예

반환 값

지정된 작업 코드와 기본 전송 방법을 포함한 RTMessage 객체를 반환합니다. 전송 의도 파라미터는 기본적으로 FAST로 설정됩니다.

SendMessage()

지정된 전송 방법을 사용하여 플레이어 또는 그룹에 메시지를 전송합니다.

구문

```
public void SendMessage(RTMessage message)
```

파라미터

message

대상 수신자, 전송 방법 및 메시지 콘텐츠를 지정하는 메시지 객체입니다.

형식: RTMessage

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다.

JoinGroup()

지정된 그룹의 멤버십에 플레이어를 추가합니다. 그룹에는 게임에 연결된 플레이어가 몇 명이든 포함될 수 있습니다. 참가하면 플레이어는 그룹에 전송된 모든 향후 메시지를 수신하며 전체 그룹에 메시지를 전송할 수 있습니다.

구문

```
public void JoinGroup(int targetGroup)
```

파라미터

targetGroup

플레이어를 추가할 그룹을 식별하는 고유 ID입니다. 그룹 ID는 개발자가 정의합니다.

유형: 정수

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다. 이 요청은 안정적인(TCP) 전송 방법을 사용하여 전송되기 때문에 실패한 요청은 OnError() 콜백을 트리거합니다.

LeaveGroup()

지정된 그룹의 멤버십에서 플레이어를 제거합니다. 그룹에 더 이상 속하지 않으면 플레이어는 그룹에 전송된 메시지를 수신하지 않으며 전체 그룹에 메시지를 전송할 수 없습니다.

구문

```
public void LeaveGroup(int targetGroup)
```

파라미터

targetGroup

플레이어를 제거할 그룹을 식별하는 고유 ID입니다. 그룹 ID는 개발자가 정의합니다.

유형: 정수

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다. 이 요청은 안정적인(TCP) 전송 방법을 사용하여 전송되기 때문에 실패한 요청은 OnError() 콜백을 트리거합니다.

RequestGroupMembership()

지정된 그룹의 플레이어 목록을 게임 클라이언트에 전송하도록 요청합니다. 플레이어가 그룹의 멤버인지 여부와 상관없이 모든 플레이어가 이 정보를 요청할 수 있습니다. 이 요청에 응답하여 멤버십 목록이 OnGroupMembershipUpdated() 콜백을 통해 클라이언트에 전송됩니다.

구문

```
public void RequestGroupMembership(int targetGroup)
```

파라미터

targetGroup

멤버십 정보를 가져올 그룹을 식별하는 고유 ID입니다. 그룹 ID는 개발자가 정의합니다.

유형: 정수

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다.

Realtime Servers Client API(C#) 참조: 비동기 콜백

이 C# Realtime Client API 참조를 사용하여 Amazon GameLift 플릿에 배포된 Realtime 서버에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 될 수 있습니다. 통합 프로세스에 대한 자세한 내용은 Realtime 서버 준비 섹션을 참조하세요.

- 동기식 작업
- 비동기 콜백
- 데이터 형식

게임 클라이언트는 이러한 콜백 메서드를 구현하여 이벤트에 응답해야 합니다. Realtime 서버는 이러한 콜백을 호출하여 게임 관련 정보를 게임 클라이언트에 전송합니다. Realtime 서버 스크립트에서 사용자 지정 게임 로직을 사용하여 동일한 이벤트에 대한 콜백을 구현할 수도 있습니다. Realtime 서버용 스크립트 콜백 섹션을 참조하세요.

콜백 메서드는 `ClientEvents.cs`에 정의되어 있습니다.

OnOpen()

서버 프로세스가 게임 클라이언트의 연결 요청을 수락하고 연결을 열 때 호출됩니다.

구문

```
public void OnOpen()
```

파라미터

이 메서드에는 파라미터가 없습니다.

반환 값

이 메서드는 아무것도 반환하지 않습니다.

OnClose()

게임 세션이 종료된 후와 같이 서버 프로세스가 게임 클라이언트와 연결을 종료할 때 호출됩니다.

구문

```
public void OnClose()
```

파라미터

이 메서드에는 파라미터가 없습니다.

반환 값

이 메서드는 아무것도 반환하지 않습니다.

OnError()

Realtime Client API 요청에 대한 오류가 발생할 때 호출됩니다. 이 콜백을 사용자 지정하여 다양한 콜백 오류를 처리할 수 있습니다.

구문

```
private void OnError(byte[] args)
```

파라미터

이 메서드에는 파라미터가 없습니다.

반환 값

이 메서드는 아무것도 반환하지 않습니다.

OnDataReceived()

게임 클라이언트가 Realtime 서버로부터 메시지를 수신할 때 호출됩니다. 이 메서드는 메시지와 알림이 게임 클라이언트에 수신되는 기본 메서드입니다.

구문

```
public void OnDataReceived(DataReceivedEventArgs dataReceivedEventArgs)
```

파라미터

dataReceivedEventArgs

메시지 활동과 관련된 정보입니다.

형식: DataReceivedEventArgs

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다.

OnGroupMembershipUpdated()

플레이어가 속한 그룹의 멤버십이 업데이트된 후에 호출됩니다. 이 콜백은 클라이언트가 `RequestGroupMembership`를 호출할 때도 호출됩니다.

구문

```
public void OnGroupMembershipUpdated(GroupMembershipEventArgs groupMembershipEventArgs)
```

파라미터

groupMembershipEventArgs

그룹 멤버십 활동과 관련된 정보입니다.

형식: GroupMembershipEventArgs

필수 항목 여부: 예

반환 값

이 메서드는 아무것도 반환하지 않습니다.

Realtime Servers Client API(C#) 참조: 데이터 유형

이 C# Realtime Client API 참조는 Amazon GameLift 플릿에 배포된 Realtime 서버에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 될 수 있습니다. 통합 프로세스에 대한 자세한 내용은 Realtime 서버 준비 섹션을 참조하세요.

- 동기식 작업
- 비동기 콜백
- 데이터 유형

ClientConfiguration

게임 클라이언트가 Realtime 서버에 연결하는 방법에 대한 정보입니다.

목차

ConnectionType

사용할 클라이언트/서버 연결 유형입니다. 보안 또는 비보안을 선택할 수 있습니다. 연결 유형을 지정하지 않는 경우 기본값은 비보안입니다.

> **Note**
>
> TLS 인증서를 사용하는 보안 플릿에서 Realtime 서버에 연결하는 경우 RT_OVER_WSS_DTLS_TLS12 값을 사용해야 합니다.

형식: `ConnectionType` 열거형 값.

필수 항목 여부: 아니요

ConnectionToken

Realtime 서버와 연결을 요청하는 게임 클라이언트 및/또는 플레이어에 대한 정보입니다.

목차

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift가 발행한 고유 ID입니다. 플레이어 세션 ID는 `PlayerSession` 객체에서 지정되며, 이 객체는 GameLift API 작업 StartGameSessionPlacement, CreateGameSession, DescribeGameSessionPlacement 또는 DescribePlayerSessions에 대한 클라이언트 호출에 응답하여 반환됩니다.

유형: 문자열

필수 항목 여부: 예

payload

연결 시 Realtime 서버에 전달할 개발자 정의 정보입니다. 이 정보에는 사용자 지정 로그인 메커니즘에 사용할 수 있는 임의 데이터가 포함됩니다. 예를 들어, 페이로드는 클라이언트가 연결하도록 허용하기 전에 Realtime 서버 스크립트에서 처리할 인증 정보를 제공합니다.

형식: 바이트 배열

필수 항목 여부: 아니요

RTMessage

메시지의 콘텐츠 및 전송 정보입니다. 메시지는 대상 플레이어 또는 대상 그룹을 지정해야 합니다.

목차

opCode

플레이어 움직임 또는 서버 알림과 같은 게임 이벤트 또는 작업을 식별하는 개발자 정의 작업 코드입니다. 메시지의 Op 코드는 제공되는 데이터 페이로드에 컨텍스트를 제공합니다. NewMessage()를 사용하여 생성된 메시지에는 작업 코드가 이미 설정되지 있지만, 언제든지 변경할 수 있습니다.

유형: 정수

필수 항목 여부: 예

targetPlayer

전송되는 메시지의 의도된 수신자인 플레이어를 식별하는 고유 ID입니다. 대상은 서버 자체(서버 ID 사용) 또는 다른 플레이어(플레이어 ID 사용)일 수 있습니다.

유형: 정수

필수 항목 여부: 아니요

targetGroup

전송되는 메시지의 의도된 수신자인 그룹을 식별하는 고유 ID입니다. 그룹 ID는 개발자가 정의합니다.

유형: 정수

필수 항목 여부: 아니요

deliveryIntent

안정적인 TCP 연결 또는 빠른 UDP 채널 중 어떤 방법을 사용하여 메시지를 전송할지를 나타냅니다. NewMessage()를 사용하여 생성된 메시지입니다.

형식: DeliveryIntent enum

유효한 값: FAST | RELIABLE

필수 항목 여부: 예

payload

메시지 콘텐츠입니다. 이 정보는 수반되는 작업 코드를 기반으로 게임 클라이언트에서 처리할 수 있도록 필요에 따라 구조화됩니다. 이 정보에는 게임 상태 데이터 또는 게임 클라이언트 간 또는 게임 클라이언트와 Realtime 서버 간에 통신해야 하는 기타 정보가 포함될 수 있습니다.

형식: 바이트 배열

필수 항목 여부: 아니요

DataReceivedEventArgs

OnDataReceived() 콜백을 통해 제공되는 데이터입니다.

목차

sender

메시지를 시작한 엔터티(플레이어 ID 또는 서버 ID)를 식별하는 고유 ID입니다.

유형: 정수

필수 항목 여부: 예

opCode

플레이어 움직임 또는 서버 알림과 같은 게임 이벤트 또는 작업을 식별하는 개발자 정의 작업 코드입니다. 메시지의 Op 코드는 제공되는 데이터 페이로드에 컨텍스트를 제공합니다.

유형: 정수

필수 항목 여부: 예

data

메시지 콘텐츠입니다. 이 정보는 수반되는 작업 코드를 기반으로 게임 클라이언트에서 처리할 수 있도록 필요에 따라 구조화됩니다. 이 정보에는 게임 상태 데이터 또는 게임 클라이언트 간 또는 게임 클라이언트와 Realtime 서버 간에 통신해야 하는 기타 정보가 포함될 수 있습니다.

형식: 바이트 배열

필수 항목 여부: 아니요

GroupMembershipEventArgs

OnGroupMembershipUpdated() 콜백을 통해 제공되는 데이터입니다.

목차

sender

그룹 멤버십 업데이트를 요청한 플레이어를 식별하는 고유 ID입니다.

유형: 정수

필수 항목 여부: 예

opCode

게임 이벤트 또는 작업을 식별하는 개발자 정의 작업 코드입니다.

유형: 정수

필수 항목 여부: 예

groupId

전송되는 메시지의 의도된 수신자인 그룹을 식별하는 고유 ID입니다. 그룹 ID는 개발자가 정의합니다.

유형: 정수

필수 항목 여부: 예

playerId

지정된 그룹의 현재 멤버인 플레이어 ID의 목록입니다.

형식: 정수 배열

필수 항목 여부: 예

Enums

Realtime Client SDK에 대해 정의된 열거형은 다음과 같이 정의됩니다.

ConnectionStatus

- CONNECTED - TCP 연결만 사용하여 게임 클라이언트가 Realtime 서버에 연결됩니다. 전송 의도와 상관없이 모든 메시지가 TCP를 통해 전송됩니다.
- CONNECTED_SEND_FAST - TCP 및 UDP 연결을 사용하여 게임 클라이언트가 Realtime 서버에 연결됩니다. 하지만 UDP를 통해 메시지를 수신하는 기능은 아직 확인되지 않으므로, 결과적으로 게임 클라이언트에 전송되는 모든 메시지는 TCP를 사용합니다.
- CONNECTED_SEND_AND_RECEIVE_FAST - TCP 및 UDP 연결을 사용하여 게임 클라이언트가 Realtime 서버에 연결됩니다. TCP 또는 UDP를 사용하여 게임 클라이언트가 메시지를 전송하고 수신할 수 있습니다.
- CONNECTING 게임 클라이언트가 연결 요청을 전송했으며 Realtime 서버가 요청을 처리하는 중입니다.
- DISCONNECTED_CLIENT_CALL - 게임 클라이언트의 Disconnect() 요청에 응답하여 Realtime 서버에서 게임 클라이언트의 연결이 해제되었습니다.
- DISCONNECTED - 클라이언트 연결 해제 호출 이외의 다른 이유로 Realtime 서버에서 게임 클라이언트의 연결이 해제되었습니다.

ConnectionType

- RT_OVER_WSS_DTLS_TLS12 - 보안 연결 유형입니다.

  TLS 인증서가 생성된 GameLift 플릿에서 실행 중인 Realtime 서버에 사용됩니다. 보안 연결을 사용하는 경우 TCP 트래픽은 TLS 1.2를 사용하여 암호화되고 UDP 트래픽은 DTLS 1.2를 사용하여 암호화됩니다.
- RT_OVER_WS_UDP_UNSECURED - 비보안 연결 유형입니다.
- RT_OVER_WEBSOCKET - 비보안 연결 유형입니다. 이 값은 더 이상 선호되지 않습니다.

DeliveryIntent

- FAST - UDP 채널을 사용하여 전송됩니다.
- RELIABLE - TCP 연결을 사용하여 전송됩니다.

Amazon GameLift Realtime 서버 스크립트 참조

이러한 리소스를 사용하여 Realtime 스크립트에 사용자 지정 로직을 확장 구축합니다.

주제

- Realtime 서버용 스크립트 콜백
- Realtime 서버 인터페이스

Realtime 서버용 스크립트 콜백

Realtime 스크립트에서 다음 콜백을 구현하여 이벤트에 응답하기 위한 사용자 지정 로직을 제공할 수 있습니다.

Init

Realtime 서버를 초기화하고 Realtime 서버 인터페이스를 수신합니다.

구문

```
init(rtsession)
```

onMessage

수신한 메시지를 서버로 전송할 때 호출됩니다.

구문

```
onMessage(gameMessage)
```

onHealthCheck

게임 세션 상태를 설정할 때 호출됩니다. 기본적인 상태는 정상(또는 `true`)입니다. 이 콜백을 구현하여 사용자 지정 상태를 확인하고 상태를 반환할 수 있습니다.

구문

```
onHealthCheck()
```

onStartGameSession

게임 세션 객체가 전달되고 새 게임 세션이 시작될 때 호출됩니다.

구문

```
onStartGameSession(session)
```

onProcessTerminate

Amazon GameLift 서비스에 의해 서버 프로세스가 종료될 때 호출됩니다. 이것이 게임 세션에서 명확하게 종료하는 트리거 역할을 할 수 있습니다. `processEnding().`를 호출할 필요가 없습니다.

구문

```
onProcessTerminate()
```

onPlayerConnect

플레이어가 연결을 요청하고 초기 확인을 통과한 경우 호출됩니다.

구문

```
onPlayerConnect(connectMessage)
```

onPlayerAccepted

플레이어 연결을 수락할 때 호출됩니다.

구문

```
onPlayerAccepted(player)
```

onPlayerDisconnect

연결 해제 요청을 전송하거나 기타 방법을 통해 플레이어가 게임 세션의 연결을 해제할 때 호출됩니다.

구문

```
onPlayerDisconnect(peerId)
```

onProcessStarted

서버 프로세스를 시작할 때 호출됩니다. 스크립트는 이 콜백을 사용하여 게임 세션을 호스팅하려고 준비하는 데 필요한 사용자 지정 작업을 수행할 수 있습니다.

구문

```
onProcessStarted(args)
```

onSendToPlayer

서버에서 다른 플레이어에게 전송할 메시지를 한 플레이어로부터 수신할 때 호출됩니다. 이 프로세스는 메시지를 전송하기 전에 실행됩니다.

구문

```
onSendToPlayer(gameMessage)
```

onSendToGroup

서버에서 그룹에 전송할 메시지를 한 플레이어로부터 수신할 때 호출됩니다. 이 프로세스는 메시지를 전송하기 전에 실행됩니다.

구문

```
onSendToGroup(gameMessage))
```

onPlayerJoinGroup

플레이어가 그룹에 가입하기 위해 요청을 전송할 때 호출됩니다.

구문

```
onPlayerJoinGroup(groupId, peerId)
```

onPlayerLeaveGroup

플레이어가 그룹에서 나가기 위해 요청을 전송할 때 호출됩니다.

구문

```
onPlayerLeaveGroup(groupId, peerId)
```

Realtime 서버 인터페이스

Realtime 스크립트가 초기화되면 Realtime 서버 인터페이스가 반환됩니다. 이 주제는 인터페이스를 통해 사용할 수 있는 속성과 메서드를 설명합니다. Realtime 스크립트 작성에 대해 자세히 알아보고 Realtime 스크립트 생성에서 자세한 스크립트 예제를 살펴봅니다.

Realtime 인터페이스를 통해 다음과 같은 객체에 액세스합니다.

- 세션
- player
- gameMessage
- 구성

Realtime 세션 객체

이러한 메서드를 사용하여 서버 관련 정보에 액세스하고 서버 관련 작업을 실시합니다.

getPlayers()

현재 게임 세션에 연결되어 있는 플레이어의 피어 ID 목록을 검색합니다. 플레이어 객체의 배열을 반환합니다.

구문

```
rtSession.getPlayers()
```

broadcastGroupMembershipUpdate()

업데이트된 그룹 구성원 자격 목록을 플레이어 그룹으로 전송합니다. 브로드캐스트할 구성원 자격(groupIdToBroadcast)과 업데이트를 수신할 그룹(targetGroupId)을 지정합니다. 그룹 ID는 양의 정수이거나 모든 그룹을 나타내려면 “-1”이어야 합니다. 사용자 정의 그룹 ID의 예에 대해서는 Realtime 서버 스크립트 예제 섹션을 참조하세요.

구문

```
rtSession.broadcastGroupMembershipUpdate(groupIdToBroadcast, targetGroupId)
```

getServerId()

메시지를 서버로 라우팅할 때 사용되는 서버의 고유한 피어 ID 식별자를 검색합니다.

구문

```
rtSession.getServerId()
```

getAllPlayersGroupId()

현재 게임 세션에 연결되어 있는 모든 플레이어가 속한 기본 그룹의 그룹 ID를 검색합니다.

구문

```
rtSession.getAllPlayersGroupId()
```

processEnding()

Realtime 서버가 게임 서버를 종료하게 합니다. 이 기능은 게임 세션에서 명확하게 종료하도록 Realtime 스크립트에서 호출해야 합니다.

구문

```
rtSession.processEnding()
```

GetGameSessionId()

현재 실행 중인 게임 세션의 고유한 ID를 검색합니다.

구문

```
rtSession.getGameSessionId()
```

getLogger()

로깅할 인터페이스를 검색합니다. 이것을 사용하여 게임 세션 로그에 캡처될 문을 기록합니다. 로거는 “정보”, “경고” 및 “오류” 문 사용을 지원합니다. 예: `logger.info("<string>")`.

구문

```
rtSession.getLogger()
```

SendMessage()

Realtime 서버에서 `newTextGameMessage` 또는 `newBinaryGameMessage`를 사용하여 생성된 메시지를 UDP 채널을 사용하는 플레이어 수신자에게 전송합니다. 플레이어의 피어 ID를 사용하여 수신자를 식별합니다.

구문

```
rtSession.sendMessage(gameMessage, targetPlayer)
```

sendGroupMessage()

Realtime 서버에서 `newTextGameMessage` 또는 `newBinaryGameMessage`를 사용하여 생성된 메시지를 UDP 채널을 사용하는 모든 플레이어 그룹에게 전송합니다. 그룹 ID는 양의 정수이거나 모든 그룹을 나타내려면 “-1”이어야 합니다. 사용자 정의 그룹 ID의 예에 대해서는 Realtime 서버 스크립트 예제 섹션을 참조하세요.

구문

```
rtSession.sendGroupMessage(gameMessage, targetGroup)
```

sendReliableMessage()

Realtime 서버에서 `newTextGameMessage` 또는 `newBinaryGameMessage`를 사용하여 생성된 메시지를 TCP 채널을 사용하는 플레이어 수신자에게 전송합니다. 플레이어의 피어 ID를 사용하여 수신자를 식별합니다.

구문

```
rtSession.sendReliableMessage(gameMessage, targetPlayer)
```

sendReliableGroupMessage()

Realtime 서버에서 newTextGameMessage 또는 newBinaryGameMessage를 사용하여 생성된 메시지를 TCP 채널을 사용하는 모든 플레이어 그룹에게 전송합니다. 그룹 ID는 양의 정수이거나 모든 그룹을 나타내려면 “-1"이어야 합니다. 사용자 정의 그룹 ID의 예에 대해서는 Realtime 서버 스크립트 예제 섹션을 참조하세요.

구문

```
rtSession.sendReliableGroupMessage(gameMessage, targetGroup)
```

newTextGameMessage()

SendMessage 기능을 사용하는 플레이어 수신자에게 서버에서 전송할 텍스트가 포함된 새로운 메시지를 생성합니다. 메시지 형식은 Realtime Client SDK에서 사용되는 형식과 비슷합니다(RTMessage 참조). gameMessage 객체를 반환합니다.

구문

```
rtSession.newTextGameMessage(opcode, sender, payload)
```

newBinaryGameMessage()

SendMessage 기능을 사용하는 플레이어 수신자에게 서버에서 전송할 이진 데이터가 포함된 새로운 메시지를 생성합니다. 메시지 형식은 Realtime Client SDK에서 사용되는 형식과 비슷합니다(RTMessage 참조). gameMessage 객체를 반환합니다.

구문

```
rtSession.newBinaryGameMessage(opcode, sender, binaryPayload)
```

플레이어 객체

플레이어 관련 정보에 액세스합니다.

player.peerId

Realtime 서버에 연결할 때 게임 클라이언트에 할당되어 게임 세션에 참가하는 고유한 ID입니다.

player.playerSessionId

Realtime 서버에 연결되고 게임 세션에 참가할 때 게임 클라이언트가 참조한 플레이어 세션 ID입니다.

게임 메시지 객체

이러한 메서드를 사용하여 Realtime 서버가 수신하는 메시지에 액세스합니다. 게임 클라이언트에서 수신되는 메시지는 RTMessage 구조를 갖습니다.

getPayloadAsText()

게임 메시지 페이로드를 텍스트로 가져옵니다.

구문

```
gameMessage.getPayloadAsText()
```

gameMessage.opcode

메시지에 포함된 작업 코드입니다.

gameMessage.payload

메시지에 포함된 페이로드입니다. 텍스트 또는 이진일 수 있습니다.

gameMessage.sender

메시지를 전송한 게임 클라이언트의 피어 ID입니다.

gameMessage.reliable

메시지를 TCP(true) 또는 UDP(false)를 통해 전송했는지 여부를 나타내는 부울입니다.

구성 객체

구성 객체를 사용하여 기본 구성을 재정의할 수 있습니다.

configuration.maxPlayers

Realtime 서버에서 허용할 수 있는 최대 클라이언트/서버 연결 수입니다.

기본값은 32입니다.

configuration.pingIntervalTime

서버가 연결된 모든 클라이언트에 핑 전송을 시도하여 연결이 정상인지 확인하는 시간 간격(밀리초)입니다.

기본값은 3000ms입니다.

Amazon GameLift Server SDK 참조

이 섹션에는 Amazon GameLift Server SDK에 대한 참조 문서가 포함되어 있습니다. Server SDK를 사용하여 사용자 지정 게임 서버를 통합하여 Amazon GameLift 서비스와 통신할 수 있습니다.

주제

- C++용 Amazon GameLift Server SDK 참조
- C#용 Amazon GameLift Server SDK 참조
- Go용 Amazon GameLift Server SDK 참조
- Unreal Engine용 Amazon GameLift Server SDK 참조

C++용 Amazon GameLift Server SDK 참조

이 Amazon GameLift C++ Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- C++용 Amazon GameLift Server SDK 5.x 참조
- Amazon GameLift C++ Server SDK 3.x 참조

C++용 Amazon GameLift Server SDK 5.x 참조

이 Amazon GameLift C++ Server SDK 5.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

> **Note**
>
> 이 주제에서는 C++ Standard Library(`std`)으로 빌드할 때 사용할 수 있는 Amazon GameLift C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DGAMELIFT_USE_STD=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

주제

- 아마존 GameLift 서버 SDK (C++) 5.x 참조: 액션
- 아마존 GameLift 서버 SDK (C++) 참조: 데이터 유형

아마존 GameLift 서버 SDK (C++) 5.x 참조: 액션

이 Amazon GameLift C++ 서버 SDK 참조를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

> **Note**
>
> 이 주제에서는 GameLift C++ 표준 라이브러리 () `std` 로 빌드할 때 사용할 수 있는 Amazon C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DGAMELIFT_USE_STD=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

작업

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessReadyAsync()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()

- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()
- Destroy()

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

반환 값

성공하면 현재 SDK 버전을 the section called “AwsStringOutcome” 객체로 반환합니다. 반환된 객체는 버전 번호(예: `5.0.0`)를 포함합니다. 실패하면 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
 Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

관리형 EC2 플릿을 위한 Amazon GameLift SDK를 초기화합니다. GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 메서드는 호스트 환경에서 서버 파라미터를 읽어 서버와 Amazon GameLift 서비스 간의 통신을 설정합니다.

명령문

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

반환 값

서버 프로세스가 ProcessReady()를 호출할 준비가 되었는지 여부를 나타내는 the section called "InitSDKOutcome" 객체를 반환합니다.

예

```
//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()

플릿의 Amazon GameLift SDK를 초기화합니다. Anywhere GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 방법을 사용하려면 서버와 Amazon GameLift 서비스 간의 통신을 설정하기 위한 명시적인 서버 파라미터가 필요합니다.

명령문

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

파라미터

ServerParameters

Amazon GameLift Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 생성하십시오.

- 게임 서버에 연결하는 데 WebSocket 사용되는 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Anywhere 컴퓨팅을 포함하는 아마존 GameLift 플릿의 ID입니다.
- Amazon GameLift 작업에서 생성된 인증 토큰입니다.

반환 값

서버 프로세스가 ProcessReady()를 호출할 준비가 되었는지 여부를 나타내는 the section called "InitSDKOutcome" 객체를 반환합니다.

> **Note**
>
> Anywhere 플릿에 배포된 게임 빌드에 대한 `InitSDK()`로의 호출이 실패하는 경우 빌드 리소스를 생성할 때 사용된 `ServerSdkVersion` 파라미터를 확인합니다. 명시적으로 이 값을 사용 중인 Server SDK 버전으로 설정해야 합니다. 이 파라미터의 기본값은 4.x이며 호환되지 않습니다. 이 문제를 해결하려면 새 빌드를 생성하여 새 플릿에 배포해야 합니다.

예

아마존 GameLift Anywhere 예시

```
//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()

서버 프로세스가 게임 세션을 호스팅할 준비가 GameLift 되었음을 Amazon에 알립니다. InitSDK() 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

명령문

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- Amazon GameLift 서비스가 서버 프로세스와 통신하기 위해 호출하는 게임 서버 코드에 구현된 콜백 메서드의 이름입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- GameLift Amazon에서 캡처하여 저장하려는 게임 세션별 파일의 경로입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 ProcessReady() 호출 및 위임 함수 구현을 모두 보여줍니다.

```
// 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()

서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 Amazon GameLift 서비스에 알립니다. 이 메서드는 서버 프로세스가 게임 세션을 호스팅할 준비가 된 후 호출되어야 합니다. 파라미터는 Amazon GameLift 호출할 콜백 함수 이름을 지정합니다. 게임 서버 코드는 이 함수를 구현해야 합니다.

이 호출은 비동기식입니다. 동기식 호출을 수행하려면 ProcessReady()를 사용합니다. 자세한 내용은 서버 프로세스 초기화 섹션을 참조하세요.

명령문

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- Amazon GameLift 서비스가 서버 프로세스와 통신하기 위해 호출하는 게임 서버 코드에 구현된 콜백 메서드의 이름입니다.

- 서버 프로세스가 수신하는 포트 번호입니다.
- GameLift Amazon에서 캡처하여 저장하려는 게임 세션별 파일의 경로입니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

서버 프로세스가 종료되고 GameLift 있음을 Amazon에 알립니다. 다른 모든 정리 작업 (활성 게임 세션 종료 포함) 이 끝난 후 프로세스를 종료하기 전에 이 메서드를 호출합니다. `ProcessEnding()`의 결과에 따라 프로세스가 성공(0) 또는 오류(-1)로 종료되고 플릿 이벤트가 생성됩니다. 오류가 발생하여 프로세스가 종료되는 경우 생성되는 플릿 이벤트는 입니다. `SERVER_PROCESS_TERMINATED_UNHEALTHY`

명령문

```
Aws::GameLift::GenericOutcome processEndingOutcome =
 Aws::GameLift::Server::ProcessEnding();
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제에서는 성공 또는 오류 발생 시 서버 프로세스를 종료하기 전에 `ProcessEnding()` 및 `Destroy()`를 호출합니다.

```
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()

서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 GameLift 되었음을 Amazon에 알립니다. 이 작업은 모든 게임 세션 초기화 후 `onStartGameSession()` 콜백 함수의 일부로 호출되어야 합니다.

명령문

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =
 Aws::GameLift::Server::ActivateGameSession();
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 `onStartGameSession()` 위임 함수의 일부로 `ActivateGameSession()`이 호출되는 것을 보여줍니다.

```
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()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다.

명령문

```
GenericOutcome
 UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy
 newPlayerSessionPolicy);
```

파라미터

playerCreationSession정책

형식: `PlayerSessionCreationPolicy` 열거형 값.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
Aws::GameLift::GenericOutcome outcome =

 Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

활성 서버 프로세스가 호스팅된 게임 세션의 ID를 가져옵니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 the section called “GameLiftError”를 반환합니다.

명령문

```
AwsStringOutcome GetGameSessionId()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 the section called “AwsStringOutcome” 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 `Success=True` 및 `GameSessionId=""`를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =
 Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 GameLift Amazon으로부터 `onProcessTerminate()` 콜백을 받은 후 조치를 취합니다. Amazon에서 GameLift 전화를 거는 `onProcessTerminate()` 이유는 다음과 같습니다.

- 서버 프로세스가 상태가 좋지 않다고 보고되었거나 GameLift Amazon에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- 스팟 인스턴스 중단으로 인해 인스턴스가 종료되는 경우

명령문

```
AwsDateTimeOutcome GetTerminationTime()
```

반환 값

성공하면 종료 시간을 `AwsDateTimeOutcome` 객체로 반환합니다. 값은 종료 시간이며, `0001 00:00:00` 이후 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값 `2020-09-13 12:26:40 -000Z`는 `637355968000000000` 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

프로세스에서 알림을 받지 못한 경우 ProcessParameters OnProcessTerminate() 콜백, 오류 메시지가 반환됩니다. 서버 프로세스 종료에 대한 자세한 내용은 서버 프로세스 종료 알림에 응답 섹션을 참조하세요.

예

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
 Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되었으며 검증이 GameLift 필요함을 Amazon에 알립니다. Amazon은 플레이어 세션 ID가 유효한지 GameLift 확인합니다. 플레이어 세션이 검증된 후 Amazon은 플레이어 슬롯의 상태를 예약됨에서 ACTIVE로 GameLift 변경합니다.

명령문

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제는 유효하지 않은 플레이어 세션 ID의 검증 및 거부를 포함하는 연결 요청을 처리합니다.

```
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()

플레이어가 서버 프로세스와의 연결이 GameLift 끊겼음을 Amazon에 알립니다. 이에 대해 Amazon은 플레이어 슬롯을 사용 가능한 것으로 GameLift 변경합니다.

명령문

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

파라미터

`playerSessionId`

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
Aws::GameLift::GenericOutcome disconnectOutcome =
 Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 메서드를 사용하여 다음에 대한 정보를 얻을 수 있습니다.

- 단일 플레이어 세션
- 게임 세션의 모든 플레이어 세션
- 단일 플레이어 ID와 연결된 모든 플레이어 세션

명령문

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

파라미터

DescribePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 the section called “DescribePlayerSessionsRequest” 객체입니다.

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 the section called “DescribePlayerSessionsOutcome” 객체를 반환합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. 제한 값을 NextToken생략하고 10으로 설정하면 Amazon은 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 GameLift 반환합니다.

```
// 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()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. 자세한 내용은 FlexMatch 백필 기능을 참조하십시오.

이 작업은 비동기식입니다. 신규 플레이어가 매칭되면 Amazon은 콜백 함수를 사용하여 업데이트된 매치메이커 데이터를 GameLift 제공합니다. `OnUpdateGameSession()`

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

명령문

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

파라미터

StartMatchBackfillRequest

다음 StartMatchBackfillRequest 정보를 전달하는 객체:

- 채우기 요청에 할당할 티켓 ID. 이 정보는 선택 사항입니다. ID를 제공하지 GameLift 않으면 Amazon에서 ID를 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에 있습니다.

- 채울 게임 세션의 ID입니다.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터입니다.

반환 값

매치 채우기 티켓 ID와 함께 the section called “StartMatchBackfillOutcome” 객체나 오류 메시지를 포함한 결함을 반환합니다.

예

```
// 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()

활성 매치 채우기 요청을 취소합니다. 자세한 내용은 FlexMatch백필 기능을 참조하십시오.

명령문

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

파라미터

StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체:

- 채우기 요청에 할당된 티켓 ID
- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

Amazon GameLift Anywhere 컴퓨팅 리소스와 Amazon 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. GameLift Amazon GameLift Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은, 을 참조하십시오 RegisterCompute.

명령문

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

반환 값

the section called “GetComputeCertificateOutcome”을 반환합니다.

예

```
Aws::GameLift::GetComputeCertificateOutcome certificate =
 Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

GameLift Amazon이 다른 사람과 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. AWS 서비스 자세한 설명은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

구문

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
 request);
```

파라미터

GetFleetRoleCredentialsRequest

반환 값

the section called “GetFleetRoleCredentialsOutcome” 객체를 반환합니다.

예

```
// 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);
```

이 예제에서는 감사 목적으로 RoleSessionName 값(선택 사항)을 사용하여 자격 증명 세션에 이름을 할당하는 방법을 보여줍니다. 역할 세션 이름을 제공하지 않으면 기본값인 “*[fleet-id]*-*[host-id]*”가 사용됩니다.

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
 getFleetRoleCredentialsRequest;
```

```
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");
getFleetRoleCredentialsRequest.SetRoleSessionName("MyFleetRoleSession");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
 Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Destroy()

Amazon GameLift 게임 서버 SDK를 메모리에서 비웁니다. `ProcessEnding()` 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 매 게임 세션 이후에 서버 프로세스를 종료하지 않는 경우, Amazon에 게임 세션을 호스팅할 준비가 되었음을 `InitSDK()` 알리기 전에 `Destroy()` GameLift 호출한 다음 다시 초기화하십시오. `ProcessReady()`

명령문

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

파라미터

파라미터는 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
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);
```

```
}
```

아마존 GameLift 서버 SDK (C++) 참조: 데이터 유형

이 Amazon GameLift C++ 서버 SDK 참조를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

> **Note**
>
> 이 주제에서는 GameLift C++ 표준 라이브러리 () `std` 로 빌드할 때 사용할 수 있는 Amazon C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DDGAMELIFT_USE_STD=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

데이터 타입

- 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
- Enums

LogParameters

게임 세션 중에 생성된 파일을 식별하는 객체로, Amazon에서 게임 세션 종료 후 업로드하고 GameLift 저장하도록 합니다. 게임 서버는 ProcessReady() 호출 시 `ProcessParameters` 객체의 GameLift 일부로 `LogParameters` Amazon에 제공합니다.

속성	설명
LogPaths	Amazon에서 나중에 액세스할 수 있도록 GameLift 저장하려는 게임 서버 로그 파일의 디렉터리 경로 목록입니다. 서버 프로세스는 각 게임 세션 중에 이러한 파일을 생성합니다. 게임 서버에서 파일 경로와 이름을 정의하고 이를 루트 게임 빌드 디렉터리에 저장합니다. 로그 경로는 절대값이어야 합니다. 예를 들어, 게임 빌드가 `MyGame\sessionLogs\` 와 같은 경로에서 게임 세션 로그를 저장하면 이 경로는 Windows 인스턴스에서 `c:\game\MyGame\sessionLogs` 가 됩니다. 유형: `std:vector<std::string>` 필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 a로 GameLift Amazon으로 전송되는 매개변수 세트가 포함되어 ProcessReady() 있습니다.

속성	설명
LogParameters	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon은 나중에 액세스할 수 있도록 파일을 GameLift 복사하고 저장합니다. 유형: `Aws::GameLift::Server:: LogParameters` 필수 항목 여부: 아니요
OnHealthCheck	Amazon이 서버 프로세스에 상태 보고서를 요청하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 60초마다 이 함수를 GameLift 호출하고 응답을 받을 때까지 60초 동안 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon은 서버 프로세스를 정상이 아닌 것으로 GameLift 기록합니다. 유형: `std::function<bool()> onHealthCheck` 필수 항목 여부: 아니요
OnProcessTerminate	Amazon이 서버 프로세스를 강제로 종료하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 이 함수를 호출한 후 서버 프로세스가 종료될 때까지 5분간 GameLift 기다린 후 ProcessEnding() 호출에 응답한 후 서버 프로세스를 종료합니다.

	유형: `std::function<void()> onProcessTerminate` 필수 항목 여부: 예
OnRefreshConnection	Amazon이 게임 서버와의 연결을 새로 고치기 위해 GameLift 호출하는 콜백 함수의 이름입니다. 유형: `void OnRefreshConnectionDelegate()` 필수 항목 여부: 예
OnStartGameSession	새 게임 세션을 활성화하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 클라이언트 요청에 대한 응답으로 이 함수를 GameLift CreateGameSession호출합니다. 콜백 함수는 Amazon GameLift API 참조에 정의된 대로 GameSession객체를 전달합니다. 유형: `const std::function<void (Aws::GameLift::Model::GameSession)> onStartGameSession` 필수 항목 여부: 예

OnUpdateGameSession	업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 업데이트된 매치메이커 데이터를 제공하기 위해 매치 백필 요청이 처리되면 이 함수를 GameLift 호출합니다. GameSession객체, 상태 업데이트 (updateReason), 매치 백필 티켓 ID를 전달합니다. 유형: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession` 필수 항목 여부: 아니요
Port	서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다. 유형: `Integer` 필수 항목 여부: 예

UpdateGameSession

이 데이터 유형은 게임 세션 객체로 업데이트되며, 여기에는 게임 세션이 업데이트된 이유와 게임 세션의 플레이어 세션을 채우기 위해 backfill을 사용하는 경우 관련된 백필 티켓 ID가 포함됩니다.

속성	설명
GameSession	Amazon GameLift API에서 정의한 GameSession객체입니다. `GameSession` 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다.

속성	설명
	유형: `Aws::GameLift::Server::GameSession` 필수 항목 여부: 예
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: `Aws::GameLift::Server::UpdateReason` 필수 항목 여부: 예
BackfillTicketId	게임 세션 업데이트를 시도하는 채우기 티켓의 ID입니다. 유형: `std::string` 필수 항목 여부: 아니요

GameSession

이 데이터 유형은 게임 세션의 세부 정보를 제공합니다.

속성	설명
GameSessionId	게임 세션에 대한 고유 식별자입니다. 게임 세션 ARN의 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 유형: `std::string` 필수 항목 여부: 아니요
명칭	게임 세션을 설명하는 레이블입니다. 유형: `std::string`

속성	설명
	필수 항목 여부: 아니요
FleetId	게임 세션이 실행 중인 플릿의 고유 식별자입니다. 유형: `std::string` 필수 항목 여부: 아니요
MaximumPlayerSessionCount	게임 세션에 연결된 최대 플레이어 수입니다. 유형: `int` 필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `in` 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `std::string` 필수 항목 여부: 아니요
GameSessionData	단일 문자열 값으로 포맷된 사용자 지정 게임 세션 속성의 집합입니다. 유형: `std::string` 필수 항목 여부: 아니요

속성	설명
MatchmakerData	게임 세션을 생성하는 데 사용된 매치메이킹 프로세스에 대한 정보(JSON 구문 사용, 문자열 형식). 사용된 매치메이킹 구성 외에도 플레이어 속성 및 팀 배정을 포함하여 경기에 배정된 모든 플레이어에 대한 데이터가 포함됩니다. 유형: `std::string` 필수 항목 여부: 아니요
GameProperties	키:값 페어로 포맷된 게임 세션에 대한 사용자 정의 속성 집합입니다. 이러한 속성은 새 게임 세션을 시작하라는 요청과 함께 전달됩니다. 유형: `std :: vector < GameProperty >` 필수 항목 여부: 아니요
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. • TLS 지원 플릿: `<unique identifier>.<region identifier>.amazongamelift.com` • TLS 미지원 플릿: `ec2-<unique identifier>.compute.amazonaws.com` TLS 지원 플릿에서 실행되는 게임 세션에 연결할 때는 IP 주소가 아닌 DNS 이름을 사용해야 합니다. 유형: `std::string` 필수 항목 여부: 아니요

ServerParameters

Amazon GameLift Anywhere 플릿의 게임 서버와 Amazon GameLift 서비스 간의 연결을 유지하는 데 사용되는 정보. 이 정보는 InitSDK()에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용하십시오.

속성	설명
webSocketUrl	`GameLiftServerSdkEndpoint` Amazon GameLift Anywhere 컴퓨팅 리소스를 `RegisterCompute` 요청하면 Amazon이 GameLift 반품합니다. 유형: `std::string` 필수 항목 여부: 예
processId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: `std::string` 필수 항목 여부: 예
hostId	`HostID`는 컴퓨터를 등록할 때 사용되는 `ComputeName` 입니다. 자세한 내용은, 을 참조하십시오 RegisterCompute. 유형: `std::string` 필수 항목 여부: 예
fleetId	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은, 을 참조하십시오 RegisterCompute. 유형: `std::string` 필수 항목 여부: 예

속성	설명
authToken	Amazon에서 생성한 인증 토큰으로, Amazon에서 GameLift 서버를 GameLift Amazon에 인증합니다. 자세한 내용은, GetComputeAuthToken을 참조하십시오. 유형: `std::string` 필수 항목 여부: 예

StartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 StartMatchBackfill() 통화를 통해 이 정보를 GameLift Amazon에 전달합니다.

속성	설명
GameSessionArn	고유한 게임 세션 식별자입니다. API 작업 `GetGameSessionId` 는 ARN 형식의 식별자를 반환합니다. 유형: `std::string` 필수 항목 여부: 예
MatchmakingConfigurationArn	매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션에 대한 매치메이커 ARN은 매치메이커 데이터 속성의 게임 세션 객체에 있습니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요. 유형: `std::string` 필수 항목 여부: 예
Players	게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용

속성	설명
	하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 유형: `std::vector<Player>` 필수 항목 여부: 예
TicketId	매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 값을 제공하지 않으면 Amazon에서 값을 GameLift 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다. 유형: `std::string` 필수 항목 여부: 아니요

Player

이 데이터 유형은 매치메이킹 중인 플레이어를 나타냅니다. 매치메이킹 요청을 시작할 때 플레이어는 플레이어 ID, 속성, 지연 시간 데이터를 가지고 있을 수 있습니다. Amazon은 경기가 이루어진 후 팀 정보를 GameLift 추가합니다.

속성	설명
LatencyInMS	플레이어가 특정 위치에 연결되었을 때 발생하는 지연 시간을 나타내는 값 집합으로, 밀리초 단위로 표시됩니다. 이 속성을 사용하는 경우 플레이어는 나열된 위치에서만 매칭됩니다. 매치메이커에 플레이어 지연 시간을 평가하는 규칙이 있는 경우 플레이어는 지연 시간을 보고해야 매칭됩니다. 유형: `Dictionary<string,int>` 필수 항목 여부: 아니요

속성	설명
PlayerAttributes	매치메이킹에 사용할 플레이어 정보가 포함된 키:값 페어의 모음입니다. 플레이어 속성 키는 매치메이킹 규칙 PlayerAttributes 세트에 사용된 것과 일치해야 합니다. 플레이어 속성에 대한 자세한 내용은 을 참조하십시오. AttributeValue 유형: `std::map<std::string,AttributeValue>` 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: `std::string` 필수 항목 여부: 아니요
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: `std::string` 필수 항목 여부: 아니요

DescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon에 대한 DescribePlayerSessions() 호출을 통해 이 정보를 제공합니다 GameLift.

속성	설명
GameSessionId	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다.

속성	설명
	게임 세션 ID 형식은 `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>` 입니다. GameSessionID 는 다음입니다. 사용자 지정 ID 문자열 또는 유형: `std::string` 필수 항목 여부: 아니요
PlayerSessionId	플레이어 세션의 고유 식별자입니다. 이 파라미터를 사용하여 단일 특정 플레이어 세션을 요청합니다. 유형: `std::string` 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 이 파라미터를 사용하여 특정 플레이어에 대한 모든 플레이어 세션을 요청합니다. 플레이어 ID 생성 섹션을 참조하십시오. 유형: `std::string` 필수 항목 여부: 아니요

속성	설명
PlayerSessionStatusFilter	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다. • RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다. • ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 연결되어 있습니다. • COMPLETED - 플레이어 연결이 끊어졌습니다. • TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다. 유형: `std::string` 필수 항목 여부: 아니요
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `std::string` 필수 항목 여부: 아니요
Limit	반환할 최대 결과 수입니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `int` 필수 항목 여부: 아니요

StopMatchBackfillRequest

매치메이킹 채우기 요청을 취소하는 데 사용되는 정보입니다. 게임 서버는 StopMatchBackfill() 통화를 통해 이 정보를 Amazon GameLift 서비스에 전달합니다.

속성	설명
GameSessionArn	취소 중인 요청의 고유한 게임 세션 식별자입니다. 유형: `char[]` 필수 항목 여부: 아니요
MatchmakingConfigurationArn	이 요청을 보낸 매치메이커의 고유 식별자입니다. 유형: `char[]` 필수 항목 여부: 아니요
TicketId	취소할 채우기 요청 티켓의 고유 식별자입니다. 유형: `char[]` 필수 항목 여부: 아니요

AttributeValue

Player 속성 키-값 페어에 이러한 값을 사용합니다. 이 객체를 사용하면 문자열, 숫자, 문자열 배열 또는 데이터 맵과 같은 유효한 데이터 유형을 사용하여 속성 값을 지정할 수 있습니다. 각 `AttributeValue` 객체는 사용 가능한 속성 `S`, `N`, `SL`, 또는 `SDM` 중 정확히 하나를 사용해야 합니다.

속성	설명
AttrType	속성 값의 유형을 지정합니다. 가능한 속성 값 유형은 다음과 같습니다. • NONE • STRING

속성	설명
	• DOUBLE • STRING_LIST • STRING_DOUBLE_MAP 필수 항목 여부: 아니요
S	문자열 속성 값을 나타냅니다. 유형: `std::string` 필수 항목 여부: 아니요
N	숫자 속성 값을 나타냅니다. 유형: `double` 필수 항목 여부: 아니요
SL	문자열 속성 값의 배열을 나타냅니다. 유형: `std::vector<std::string>` 필수 항목 여부: 아니요
SDM	문자열 키와 이중 값의 사전을 나타냅니다. 유형: `std::map<std::string, double>` 필수 항목 여부: 아니요

GetFleetRoleCredentialsRequest

이 데이터 유형을 사용하면 게임 서버가 다른 AWS 리소스에 제한적으로 액세스할 수 있습니다. 자세한 내용은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요.

속성	설명
RoleArn	AWS 리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `std::string` 필수 항목 여부: 아니요
RoleSessionName	세션을 고유하게 식별하는 데 사용할 수 있는 역할 세션 이름입니다. AWS Security Token Service AssumeRole 이 이름은 에 있는 것과 같은 감사 로그에 CloudTrail 노출됩니다. 유형: `std::string` 필수 항목 여부: 아니요

AwsLongOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `long` 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `long&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다.

속성	설명
	유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

AwsStringOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `std::string` 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `long&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError”

속성	설명
	필수 항목 여부: 아니요

DescribePlayerSessionsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “DescribePlayerSessionsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

DescribePlayerSessionsResult

요청과 일치하는 각 플레이어 세션의 속성을 포함하는 객체 모음입니다.

속성	설명
NextToken	결과의 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작 부분에서 시작하려면 값을 지정하지 않습니다. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다. 유형: `std::string` 필수 항목 여부: 예
PlayerSessions	유형: `IList<`the section called “PlayerSession” `>` 필수 항목 여부:
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `std::string&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GenericOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GenericOutcomeCallable

이 데이터 유형은 비동기식 일반 결과입니다. 여기에는 다음과 같은 속성이 있습니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

PlayerSession

이 데이터 유형은 Amazon이 게임 서버로 GameLift 전달하는 플레이어 세션을 나타냅니다. 자세한 내용은 을 참조하십시오 PlayerSession.

속성	설명
CreationTime	유형: `long` 필수 항목 여부: 아니요
FleetId	유형: `std::string` 필수 항목 여부: 아니요
GameSessionId	유형: `std::string` 필수 항목 여부: 아니요
IpAddress	유형: `std::string` 필수 항목 여부: 아니요
PlayerData	유형: `std::string` 필수 항목 여부: 아니요
PlayerId	유형: `std::string` 필수 항목 여부: 아니요
PlayerSessionId	유형: `std::string` 필수 항목 여부: 아니요
Port	유형: `int` 필수 항목 여부: 아니요
상태 표시기	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. PlayerSessionId PlayerId OR 가 제공되면 는 응답에 영향을 주지 않습니다. PlayerSessionStatusFilter 유형: `PlayerSessionStatus` 열거형. 가능한 값은 다음을 포함합니다.

속성	설명
	• ACTIVE • COMPLETED • NOT_SET • RESERVED • TIMEDOUT 필수 항목 여부: 아니요
TerminationTime	유형: `long` 필수 항목 여부: 아니요
DnsName	유형: `std::string` 필수 항목 여부: 아니요

StartMatchBackfillOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “StartMatchBackfillResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `StartMatchBackfillResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다.

속성	설명
	유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

StartMatchBackfillResult

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
TicketId	매치메이킹 티켓의 고유 식별자입니다. 여기에 티켓 ID를 지정하지 않은 경우 GameLift Amazon은 UUID 형태로 티켓 ID를 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하고 매치 결과를 검색할 수 있습니다. 유형: `std::string` 필수 항목 여부: 아니요

GetComputeCertificateOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “GetComputeCertificateResult”

속성	설명
	필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `Aws::GameLift::Server::Model::GetComputeCertificateResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GetComputeCertificateResult

컴퓨팅의 TLS 인증서 경로 및 컴퓨팅 호스트 이름입니다.

속성	설명
CertificatePath	컴퓨팅 리소스에 있는 TLS 인증서의 경로입니다. Amazon GameLift 관리형 플릿을 사용하는 경우 이 경로에는 다음이 포함됩니다. • `certificate.pem` : 최종 사용자 인증서입니다. 전체 인증서 체인은 이 인증서에 추가된 `certificateChain.pem` 의 조합입니다. • `certificateChain.pem` : 루트 인증서와 중간 인증서를 포함하는 인증서 체인입니다.

속성	설명
	• `rootCertificate.pem` : 루트 인증서입니다. • `privateKey.pem` : 최종 사용자 인증서의 프라이빗 키입니다. 유형: `std::string` 필수 항목 여부: 아니요
ComputeName	컴퓨팅 리소스의 이름입니다. 유형: `std::string` 필수 항목 여부: 아니요

GetFleetRoleCredentialsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “GetFleetRoleCredentialsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult` 필수 항목 여부: 아니요

속성	설명
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GetFleetRoleCredentialsResult

속성	설명
AccessKeyId	인증하고 AWS 리소스에 액세스를 제공하기 위한 액세스 키 ID입니다. 유형: `string` 필수 항목 여부: 아니요
AssumedRoleId	서비스 역할이 속한 사용자의 ID입니다. 유형: `string` 필수 항목 여부: 아니요
AssumedRoleUserArn	사용자가 맡을 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `string` 필수 항목 여부: 아니요
Expiration	세션 자격 증명이 만료될 때까지 남은 시간입니다.

속성	설명
	유형: `DateTime` 필수 항목 여부: 아니요
SecretAccessKey	인증에 대한 비밀 액세스 키 ID를 지정합니다. 유형: `string` 필수 항목 여부: 아니요
SessionToken	AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하는 토큰입니다. 유형: `string` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

InitSDKOutcome

> **Note**
>
> `InitSDKOutcome`은 `std` 플래그를 사용하여 SDK를 빌드하는 경우에만 반환됩니다. `nostd` 플래그를 사용하여 빌드하면 the section called “GenericOutcome”이 대신 반환됩니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: `GameLiftErrorType` 열거형. 필수 항목 여부: 아니요
ErrorName	오류 유형 이름입니다. 유형: `std::string` 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: `std::string` 필수 항목 여부: 아니요

Enums

Amazon GameLift 서버 SDK (C++) 에 대해 정의된 열거형은 다음과 같이 정의됩니다.

GameLiftErrorType

오류 유형을 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- BAD_REQUEST_EXCEPTION
- GAMESESSION_ID_NOT_SET - 게임 세션 ID가 설정되지 않았습니다.
- INTERNAL_SERVICE_EXCEPTION
- LOCAL_CONNECTION_FAILED — 아마존에 대한 로컬 연결에 실패했습니다. GameLift
- NETWORK_NOT_INITIALIZED - 네트워크가 초기화되지 않았습니다.
- SERVICE_CALL_FAILED - AWS 서비스 호출이 실패했습니다.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- ALREADY_INITIALEZED — GameLift 아마존 서버 또는 클라이언트가 이미 Initialize () 를 사용하여 초기화되었습니다.
- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALEZED — 아마존 클라이언트가 초기화되지 않았습니다. GameLift
- GAMELIFT_SERVER_NOT_INITIALIZED — 아마존 서버가 초기화되지 않았습니다. GameLift
- GAME_SESSION_ENDED_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY — 아마존 GameLift 서버 게임 세션이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 게임 세션이 준비되었다고 보고할 수 없습니다.
- INITIALIZATION_MISMATCH - 클라이언트 메서드는 Server::Initialize() 이후에 호출되었으며 그 반대의 경우도 마찬가지입니다.
- NOT_INITIALIZED — 아마존 GameLift 서버 또는 클라이언트가 Initialize () 를 사용하여 초기화되지 않았습니다.
- NO_TARGET_ALIASID_SET - 대상 aliasId가 설정되지 않았습니다.
- NO_TARGET_FLEET_SET - 대상 플릿이 설정되지 않았습니다.
- PROCESS_ENDING_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 프로세스가 종료되었음을 보고할 수 없습니다.

- PROCESS_NOT_ACTIVE — 서버 프로세스가 아직 활성화되지 않았고 a에 바인딩되지 않았으므로 수락하거나 처리할 수 없습니다. GameSession PlayerSessions
- PROCESS_NOT_READY - 서버 프로세스를 아직 활성화할 준비가 되지 않았습니다.
- PROCESS_READY_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 프로세스가 준비되었다고 보고할 수 없습니다.
- SDK_VERSION_DETECTION_FAILED - SDK 버전 감지에 실패했습니다.
- STX_CALL_FAILED - XStx 서버 백엔드 구성 요소에 대한 호출이 실패했습니다.
- STX_INITIALIZATION_FAILED - XStx 서버 백엔드 구성 요소를 초기화하지 못했습니다.
- UNEXPECTED_PLAYER_SESSION - 서버에서 등록되지 않은 플레이어 세션을 발견했습니다.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE — 서비스에 메시지를 보내는데 다시 실패했습니다. GameLift WebSocket
- WEBSOCKET_SEND_MESSAGE_FAILURE — 서비스에 메시지를 GameLift 보내지 못했습니다. WebSocket
- MATCH_BACKFILL_REQUEST_VALIDATION - 요청 검증에 실패했습니다.
- PLAYER_SESSION_REQUEST_VALIDATION - 요청 검증에 실패했습니다.

PlayerSessionCreationPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.
- NOT_SET - 게임 세션이 새 플레이어 세션을 수락하거나 거부하도록 설정되지 않았습니다.

Amazon GameLift C++ Server SDK 3.x 참조

이 Amazon GameLift C++ Server SDK 3.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- Amazon GameLift Server SDK(C++) 참조: 작업
- Amazon GameLift Server SDK(C++) 참조: 데이터 유형

Amazon GameLift Server SDK(C++) 참조: 작업

이 Amazon GameLift C++ Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

작업

- AcceptPlayerSession()
- ActivateGameSession()
- DescribePlayerSessions()
- GetGameSessionId()
- GetInstanceCertificate()
- GetSdkVersion()
- GetTerminationTime()
- InitSDK()
- ProcessEnding()
- ProcessReady()
- ProcessReadyAsync()
- RemovePlayerSession()
- StartMatchBackfill()
- StopMatchBackfill()
- TerminateGameSession()
- UpdatePlayerSessionCreationPolicy()
- Destroy()

AcceptPlayerSession()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift가 플레이어 세션 ID가 유효한지, 즉 플레이어 ID가 게임 세

션의 플레이어 슬롯을 예약했는지 확인합니다. 검증되면 Amazon GameLift가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

파라미터

playerSessionId

AWS SDK Amazon GameLift API 작업 CreatePlayerSession에 호출에 대한 응답으로 Amazon GameLift 서비스에서 발급한 고유 ID입니다. 게임 클라이언트가 서버 프로세스에 연결할 때 이 ID를 참조합니다.

유형: std::string

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 잘못된 플레이어 세션 ID 검증 및 거부를 비롯해 연결 요청을 처리하기 위한 함수를 보여줍니다.

```
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()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 시작했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화가 완료된 후 `onStartGameSession()` 콜백 함수의 일부로 호출되어야 합니다.

구문

```
GenericOutcome ActivateGameSession();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 `onStartGameSession()` 콜백 함수의 일부로 `ActivateGameSession()`이 호출되는 것을 보여줍니다.

```
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()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 작업을 사용하면 단일 플레이어 세션 정보, 게임 세션에 있는 모든 플레이어 세션 정보 또는 단일 플레이어 ID와 관련된 모든 플레이어 세션 정보를 가져올 수 있습니다.

구문

```
DescribePlayerSessionsOutcome DescribePlayerSessions (
```

```
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest
&describePlayerSessionsRequest);
```

파라미터

describePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 DescribePlayerSessionsRequest 객체입니다.

필수 항목 여부: 예

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 `DescribePlayerSessionsOutcome` 객체를 반환합니다. 플레이어 세션 객체 구조는 AWS SDK Amazon GameLift API PlayerSession 데이터 유형과 동일합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. `NextToken`을 생략하고 `Limit` 값을 10으로 설정하면, Amazon GameLift가 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 반환합니다.

```
// 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()

서버 프로세스가 활성인 경우 현재 서버 프로세스에서 호스팅하고 있는 게임 세션의 고유 식별자를 가져옵니다. 식별자는 다음과 같은 ARN 형식으로 반환됩니다. `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 `Success=True` 및 `GameSessionId=""`(빈 문자열)를 반환합니다.

구문

```
AwsStringOutcome GetGameSessionId();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 AwsStringOutcome 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

플릿 및 해당 인스턴스와 연결된 pem 인코딩된 TLS 인증서의 파일 위치를 검색합니다. AWS Certificate Manager는 인증서 구성이 GENERATED로 설정된 새 플릿을 생성할 때 이 인증서를 생성합니다. 이 인증서를 사용하여 게임 클라이언트와 보안 연결을 설정하고 클라이언트/서버 통신을 암호화합니다

구문

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 인스턴스에 저장된 플릿의 TLS 인증서 파일과 인증서 체인 위치를 포함한 GetInstanceCertificateOutcome 객체를 반환합니다. 인증서 체인에서 추출한 루트 인증서 파일도 인스턴스에 저장됩니다. 실패하면 오류 메시지를 반환합니다.

인증서 및 인증서 체인 데이터에 대한 자세한 내용은 AWS Certificate Manager API 참조의 GetCertificate 응답 요소를 참조하세요.

예

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

사용 중인 SDK의 현재 버전 번호를 반환합니다.

구문

```
AwsStringOutcome GetSdkVersion();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 현재 SDK 버전을 AwsStringOutcome 객체로 반환합니다. 반환된 문자열에는 버전 번호만 (예: “3.1.5”)을 포함합니다. 실패하면 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
    Aws::GameLift::Server::GetSdkVersion();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift 서비스에서 onProcessTerminate() 콜백을 수신한 후 이 작업을 수행합니다. Amazon GameLift는 다음 이유로 onProcessTerminate()를 호출할 수 있습니다. (1) 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift에 응답하지 않은 경우, (2) 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우 또는 (3) 스팟 중단으로 인해 인스턴스가 종료되는 경우.

프로세스가 onProcessTerminate() 콜백을 수신할 때 반환되는 값은 예상 종료 시간입니다. 프로세스가 onProcessTerminate() 콜백을 받지 못한 경우 오류 메시지가 반환됩니다. 서버 프로세스 종료에 대해 자세히 알아보세요.

구문

```
AwsLongOutcome GetTerminationTime();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 종료 시간을 AwsLongOutcome 객체로 반환합니다. 값은 종료 시간이며, 0001 00:00:00 이후 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값인 2020-09-13 12:26:40 -000Z는 637355968000000000 틱과 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Amazon GameLift SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift 관련 초기화가 진행되기 전, 시작 시 호출되어야 합니다.

구문

```
InitSDKOutcome InitSDK();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하는 경우, 서버 프로세스가 ProcessReady()를 호출할 준비가 되었음을 나타내는 InitSdkOutcome 객체를 반환합니다.

예

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Amazon GameLift 서비스에 서버 프로세스를 중단할 준비가 되었음을 알립니다. 이 메서드는 모든 활성 게임 세션의 종료 등, 다른 모든 정리 작업 이후에 호출되어야 합니다. 이 메서드는 종료 코드 0으로

종료해야 합니다. 0이 아닌 종료 코드를 사용하면 프로세스가 정상적으로 종료되지 않았다는 이벤트 메시지가 표시됩니다.

코드를 0으로 메서드가 종료되면 종료 코드를 성공시켜 프로세스를 종료할 수 있습니다. 오류 코드를 표시하여 프로세스를 종료할 수도 있습니다. 오류 코드를 남기고 종료하면 플릿 이벤트는 프로세스가 비정상적으로 종료(SERVER_PROCESS_TERMINATED_UNHEALTHY)되었음을 나타냅니다.

구문

```
GenericOutcome ProcessEnding();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
if (outcome.Success)
    exit(0);  // exit with success
// otherwise, exit with error code
exit(errorCode);
```

ProcessReady()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 InitSDK()를 호출하고, 서버 프로세스가 게임 세션을 호스트하기 전에 수행해야 하는 설정 작업을 완료한 후에만 이 메서드를 호출할 수 있습니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

이 호출은 동기식입니다. 비동기식 호출을 수행하려면 ProcessReadyAsync()를 사용합니다. 자세한 내용은 서버 프로세스 초기화 섹션을 참조하세요.

구문

```
GenericOutcome ProcessReady(
```

```
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- 게임 서버 코드에 구현되어 있는 콜백 메서드의 이름으로, 서버 프로세스와 통신할 수 있도록 Amazon GameLift 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 ProcessReady() 호출 및 콜백 함수 구현을 모두 보여줍니다.

```
// 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()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 이 메서드는 서버 프로세스가 게임 세션을 호스팅할 준비가 된 후 호출되어야 합니다. 이 파라미터는 특정 상황에서 호출할 Amazon GameLift 콜백 함수의 이름을 지정합니다. 게임 서버 코드는 이 함수를 구현해야 합니다.

이 호출은 비동기식입니다. 동기식 호출을 수행하려면 ProcessReady()를 사용합니다. 자세한 내용은 서버 프로세스 초기화 섹션을 참조하세요.

구문

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- 게임 서버 코드에 구현되어 있는 콜백 메서드의 이름으로, 서버 프로세스와 통신할 수 있도록 Amazon GameLift 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);
```

파라미터

playerSessionId

AWS SDK Amazon GameLift API 작업 CreatePlayerSession에 호출에 대한 응답으로 Amazon GameLift 서비스에서 발급한 고유 ID입니다. 게임 클라이언트가 서버 프로세스에 연결할 때 이 ID를 참조합니다.

유형: std::string

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
Aws::GameLift::GenericOutcome disconnectOutcome =
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. AWS SDK 작업 StartMatchBackfill()도 참조하세요. 이 작업을 사용하면 게임 세션을 호스팅하는 게임 서버 프로세스에서 매치 채우기 요청을 시작할 수 있습니다. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift 서비스는 콜백 함수 `OnUpdateGameSession()`을 호출하여 업데이트된 매치메이커 데이터를 전달합니다.

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

구문

```
StartMatchBackfillOutcome StartMatchBackfill (
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest
 &startBackfillRequest);
```

파라미터

StartMatchBackfillRequest

다음 정보를 전달하는 StartMatchBackfillRequest 객체입니다.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift가 ID를 자동 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에서 수집할 수 있습니다.
- 백필(backfill) 중인 게임 세션의 ID.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터.

필수 항목 여부: 예

반환 값

매치 채우기 티켓과 함께 StartMatchBackfillOutcome 객체를 반환하거나 오류 메시지를 포함한 결함이 있는 StartMatchBackfillOutcome 객체를 반환합니다. 티켓 상태는 AWS SDK 작업 DescribeMatchmaking()을 사용하여 추적할 수 있습니다.

예

```
// 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()

StartMatchBackfill()을 통해 생성된 활성 매치 채우기 요청을 취소합니다. AWS SDK 작업 StopMatchmaking()도 참조하세요. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

구문

```
GenericOutcome StopMatchBackfill (
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);
```

파라미터

StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체입니다.

- 취소 중인 채우기 요청에 할당한 티켓 ID

- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

버전 4.0.1 에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 ProcessEnding()을 호출해야 합니다.

Amazon GameLift 서비스에 서버 프로세스가 현재 게임 세션을 종료했음을 알립니다. 이 작업은 서버 프로세스가 활성 상태를 유지하고 새 게임 세션을 호스팅할 준비가 되면 호출됩니다. 서버 프로세스를 즉시 사용하여 새 게임 세션을 호스팅할 수 있음을 Amazon GameLift에 알리기 때문에 게임 세션 종료 절차가 완료된 후에만 호출해야 합니다.

게임 세션이 중지된 후 서버 프로세스가 종료되는 경우에는 이 작업이 호출되지 않습니다. 대신 ProcessEnding()을 호출하여 게임 세션과 서버 프로세스가 모두 종료되었음을 알립니다.

구문

```
GenericOutcome TerminateGameSession();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

UpdatePlayerSessionCreationPolicy()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다. AWS SDK 작업 UpdateGameSession()도 참조하세요.

구문

```
GenericOutcome UpdatePlayerSessionCreationPolicy(
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

파라미터

newPlayerSessionPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다.

유형: Aws::GameLift::Model::PlayerSessionCreationPolicy 열거형. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
Aws::GameLift::GenericOutcome outcome =
 Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Destroy()

게임 서버 초기화 중에 initSDK()에서 할당한 메모리를 정리합니다. 게임 서버 프로세스를 종료한 후 이 메서드를 사용하면 서버 메모리 낭비를 피할 수 있습니다.

구문

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

파라미터

파라미터는 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제에서는 게임 서버 프로세스가 종료된 후 initSDK에서 할당한 메모리를 정리합니다.

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {
  Aws::GameLift::Server::Destroy();
  exit(0);
}
```

Amazon GameLift Server SDK(C++) 참조: 데이터 유형

이 Amazon GameLift C++ Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

이 API는 `GameLiftServerAPI.h`, `LogParameters.h` 및 `ProcessParameters.h`에 정의되어 있습니다.

- 작업
- 데이터 유형

DescribePlayerSessionsRequest

이 데이터 형식은 검색할 플레이어 세션을 지정하는 데 사용됩니다. 다음과 같이 사용할 수 있습니다.

- 특정 플레이어 세션을 요청하려면 PlayerSessionId를 제공합니다.
- 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 GameSessionId를 제공합니다.
- 지정한 플레이어의 모든 플레이어 세션을 요청하려면 PlayerId를 제공합니다.

플레이어 세션이 대량일 경우 페이지 매김 파라미터를 사용하여 결과를 순차 블록으로 검색합니다.

목차

GameSessionId

고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. <ID string> 값은 사용자 지정 ID 문자열(게임 세션을 만들 때 지정한 경우) 또는 생성 문자열입니다.

유형: 문자열

필수 항목 여부: 아니요

Limit

반환할 최대 결과 수입니다. 결과를 순차적인 일련의 페이지로 가져오려면 이 파라미터를 NextToken과 함께 사용합니다. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 정수

필수 항목 여부: 아니요

NextToken

결과의 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마십시오. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerId

사용자의 고유 식별자입니다. 플레이어 ID는 개발자에 의해 정의됩니다. 플레이어 ID 생성 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionId

플레이어 세션의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionStatusFilter

결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다.

- RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다.
- ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 현재 연결되어 있습니다.
- COMPLETED - 플레이어 연결이 끊어졌습니다.
- TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다.

유형: 문자열

필수 항목 여부: 아니요

LogParameters

이 데이터 유형은 게임 세션 중에 생성된 파일 가운데 게임 세션 종료 시 Amazon GameLift가 업로드 및 저장하도록 하려는 파일을 식별합니다. 이 정보는 ProcessReady() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

logPaths

Amazon GameLift가 향후 사용을 위해 저장하도록 하려는 게임 서버 로그 파일의 디렉터리 경로입니다. 이러한 파일은 각 게임 세션 중에 생성됩니다. 파일 경로와 이름은 게임 서버에서 정의되고 루트 게임 빌드 디렉터리에 저장됩니다. 로그 경로는 절대값이어야 합니다. 예를 들어, 게임 빌드가 `MyGame\sessionlogs\`와 같은 경로에 게임 세션 로그를 저장할 경우 로그 경

로는 `c:\game\MyGame\sessionLogs`(Windows 인스턴스) 또는 `/local/game/MyGame/sessionLogs`(Linux 인스턴스)가 됩니다.

형식: std:vector<std::string>

필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 ProcessReady() 호출 시 Amazon GameLift 서비스로 보낸 파라미터 집합이 포함됩니다.

목차

port

서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다.

유형: 정수

필수 항목 여부: 예

logParameters

게임 세션 로그 파일의 디렉터리 경로 목록을 포함하는 객체입니다.

유형: Aws::GameLift::Server::LogParameters

필수 항목 여부: 아니요

onStartGameSession

Amazon GameLift가 새 게임 세션을 활성화하기 위해 호출하는 콜백 함수 이름입니다. Amazon GameLift는 클라이언트 요청 CreateGameSession에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 GameSession 객체를 전달합니다(Amazon GameLift Service API 참조에서 정의함).

유형: `const std::function<void(Aws::GameLift::Model::GameSession)> onStartGameSession`

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift 서비스가 서버 프로세스를 강제로 종료하기 위해 호출하는 콜백 함수 이름입니다. 이 함수를 호출한 후 Amazon GameLift는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답합니다. 응답이 수신되지 않으면 서버 프로세스를 종료합니다.

유형: `std::function<void()> onProcessTerminate`

필수 항목 여부: 아니요

onHealthCheck

Amazon GameLift 서비스가 서버 프로세스에 상태 보고서를 요청하기 위해 호출하는 콜백 함수의 이름입니다. Amazon GameLift는 60초마다 이 함수를 호출합니다. 이 함수를 호출한 후 Amazon GameLift는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상으로 기록합니다.

유형: `std::function<bool()> onHealthCheck`

필수 항목 여부: 아니요

onUpdateGameSession

업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon GameLift 서비스가 호출하는 콜백 함수 이름입니다. Amazon GameLift는 업데이트된 매치메이커 데이터를 제공하기 위해 매치 채우기 요청을 처리할 때 이 함수를 호출합니다. GameSession 객체, 상태 업데이트 (`updateReason`) 및 매치 채우기 티켓 ID를 전달합니다.

유형: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)> onUpdateGameSession`

필수 항목 여부: 아니요

StartMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 이 정보는 StartMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

고유한 게임 세션 식별자입니다. GetGameSessionId() API 작업은 ARN 형식의 식별자를 반환합니다.

유형: 문자열

필수 항목 여부: 예

MatchmakingConfigurationArn

매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션을 만드는 데 사용된 매치메이커를 찾으려면 게임 세션 객체에서 매치메이커 데이터 속성을 확인합니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요.

유형: 문자열

필수 항목 여부: 예

Players

현재 게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 플레이어 객체 형식에 대한 자세한 내용은 Amazon GameLift API 참조 가이드를 참조하세요. 플레이어 속성, ID 및 팀 배정을 찾으려면 매치메이커 데이터 속성에서 게임 세션 객체를 확인합니다. 매치메이커에서 지연 시간을 사용하는 경우 현재 리전에 대한 업데이트 지연 시간을 수집하여 각 플레이어의 데이터에 포함합니다.

유형: std:vector<player>

필수 항목 여부: 예

TicketId

매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 여기에 값이 제공되지 않으면 Amazon GameLift는 UUID 형식으로 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다.

유형: 문자열

필수 항목 여부: 아니요

StopMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 취소하는 데 사용됩니다. 이 정보는 StopMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

취소 중인 요청과 연결된 고유한 게임 세션 식별자입니다.

유형: 문자열

필수 항목 여부: 예

MatchmakingConfigurationArn

이 요청을 보낸 매치메이커의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 예

TicketId

취소할 채우기 요청 티켓의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 예

C#용 Amazon GameLift Server SDK 참조

이 Amazon GameLift C# Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조
- C#용 Amazon GameLift Server SDK 4.x참조

C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조

이 Amazon GameLift C# Server SDK 5.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift

를 게임 서버에 추가 섹션을 참조하고, Unity용 C# Server SDK 플러그인 사용에 대한 자세한 내용은 Amazon GameLift를 Unity 프로젝트에 통합 섹션을 참조하세요. C#용 Amazon GameLift Server SDK 5.x는 .NET 4.6 및 .NET 6를 지원합니다.

주제

- C# 및 Unity용 Amazon GameLift 서버 SDK 레퍼런스: 액션
- C# 및 Unity용 Amazon GameLift Server SDK 참조: 데이터 유형

C# 및 Unity용 Amazon GameLift 서버 SDK 레퍼런스: 액션

이 Amazon GameLift C# 서버 SDK 참조는 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하고, Unity용 C# Server SDK 플러그인 사용에 대한 자세한 내용은 Amazon GameLift를 Unity 프로젝트에 통합 섹션을 참조하세요.

작업

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()
- GetComputeCertificate()
- GetFleetRoleCredentials()

- Destroy()

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
AwsStringOutcome GetSdkVersion();
```

반환 값

성공하면 현재 SDK 버전을 the section called “AwsStringOutcome” 객체로 반환합니다. 반환된 문자열에는 버전 번호(예: `5.0.0`)가 포함됩니다. 실패하면 오류 메시지를 반환합니다.

예

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitSDK()

관리형 EC2 플릿을 위한 Amazon GameLift SDK를 초기화합니다. GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 메서드는 호스트 환경에서 서버 파라미터를 읽어 서버와 Amazon GameLift 서비스 간의 통신을 설정합니다.

명령문

```
GenericOutcome InitSDK();
```

반환 값

성공하면 서버 프로세스가 호출할 준비가 되었음을 나타내는 InitSdkOutcome 객체를 반환합니다ProcessReady().

예

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

플릿의 Amazon GameLift SDK를 초기화합니다. Anywhere GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 방법을 사용하려면 서버와 Amazon GameLift 서비스 간의 통신을 설정하기 위한 명시적인 서버 파라미터가 필요합니다.

명령문

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

파라미터

ServerParameters

Amazon GameLift Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 생성하십시오.

- 게임 서버에 연결하는 데 WebSocket 사용되는 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Anywhere 컴퓨팅을 포함하는 아마존 GameLift 플릿의 ID입니다.
- Amazon GameLift 작업에서 생성된 인증 토큰.

반환 값

성공하면 서버 프로세스가 호출할 준비가 되었음을 나타내는 InitSdkOutcome 객체를 반환합니다ProcessReady().

> **Note**
>
> Anywhere 플릿에 배포된 게임 빌드에 대한 `InitSDK()`로의 호출이 실패하는 경우 빌드 리소스를 생성할 때 사용된 `ServerSdkVersion` 파라미터를 확인합니다. 명시적으로 이 값을 사용 중인 Server SDK 버전으로 설정해야 합니다. 이 파라미터의 기본값은 4.x이며 호환되지 않습니다. 이 문제를 해결하려면 새 빌드를 생성하여 새 플릿에 배포해야 합니다.

예

```
//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()

서버 프로세스가 게임 세션을 호스팅할 준비가 GameLift 되었음을 Amazon에 알립니다. InitSDK() 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

명령문

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

파라미터

ProcessParameters

`ProcessParameters` 객체에는 서버 프로세스에 대한 정보가 들어 있습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 메서드 및 위임 함수 구현을 모두 보여줍니다.

```
// 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()

서버 프로세스가 종료되고 GameLift 있음을 Amazon에 알립니다. 다른 모든 정리 작업 (활성 게임 세션 종료 포함) 이 끝난 후 프로세스를 종료하기 전에 이 메서드를 호출합니다. `ProcessEnding()`의 결과에 따라 프로세스가 성공(0) 또는 오류(-1)로 종료되고 플릿 이벤트가 생성됩니다. 오류가 발생하여 프로세스가 종료되는 경우 생성되는 플릿 이벤트는 입니다. `SERVER_PROCESS_TERMINATED_UNHEALTHY`

명령문

```
GenericOutcome ProcessEnding()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제에서는 성공 또는 오류 발생 시 서버 프로세스를 종료하기 전에 `ProcessEnding()` 및 `Destroy()`를 호출합니다.

```
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()

서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 GameLift 되었음을 Amazon에 알립니다. 이 작업은 모든 게임 세션 초기화 후 `onStartGameSession()` 콜백 함수의 일부로 호출되어야 합니다.

명령문

```
GenericOutcome ActivateGameSession()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 `onStartGameSession()` 위임 함수의 일부로 `ActivateGameSession()`이 호출되는 것을 보여줍니다.

```
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()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다.

명령문

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
 playerSessionPolicy)
```

파라미터

playerSessionPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다.

유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
GenericOutcome updatePlayerSessionPolicyOutcome =

 GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

활성 서버 프로세스가 호스팅된 게임 세션의 ID를 가져옵니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 the section called “GameLiftError”를 반환합니다.

명령문

```
AwsStringOutcome GetGameSessionId()
```

반환 값

성공하면 게임 세션 ID를 the section called “AwsStringOutcome” 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

예

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 GameLift Amazon으로부터 `onProcessTerminate()` 콜백을 받은 후 이 작업을 수행합니다. Amazon에서 GameLift 전화를 거는 `onProcessTerminate()` 이유는 다음과 같습니다.

- 서버 프로세스가 상태가 좋지 않다고 보고되었거나 GameLift Amazon에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- 스팟 인스턴스 중단으로 인해 인스턴스가 종료되는 경우

명령문

```
AwsDateTimeOutcome GetTerminationTime()
```

반환 값

성공하면 종료 시간을 the section called “AwsDateTimeOutcome” 객체로 반환합니다. 값은 종료 시간이며, `0001 00:00:00` 이후 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값 `2020-09-13 12:26:40 -000Z`는 `637355968000000000` 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되었으며 검증이 GameLift 필요함을 Amazon에 알립니다. Amazon은 플레이어 세션 ID가 유효한지 GameLift 확인합니다. 플레이어 세션이 검증된 후 Amazon은 플레이어 슬롯의 상태를 예약됨에서 ACTIVE로 GameLift 변경합니다.

명령문

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 발급되는 고유 ID.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 잘못된 플레이어 세션 ID 검증 및 거부를 비롯해 연결 요청을 처리하기 위한 함수를 보여줍니다.

```
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()

플레이어가 서버 프로세스와의 연결이 GameLift 끊겼음을 Amazon에 알립니다. 이에 대해 Amazon은 플레이어 슬롯을 사용 가능한 것으로 GameLift 변경합니다.

명령문

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
GenericOutcome removePlayerSessionOutcome =
 GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 작업을 사용하면 단일 플레이어 세션 정보, 게임 세션에 있는 모든 플레이어 세션 정보 또는 단일 플레이어 ID와 관련된 모든 플레이어 세션 정보를 가져올 수 있습니다.

명령문

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

파라미터

DescribePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 the section called “DescribePlayerSessionsRequest” 객체입니다.

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 the section called “DescribePlayerSessionsOutcome” 객체를 반환합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. 제한 값을 NextToken생략하고 10으로 설정하면 GameLift Amazon은 요청과 일치하는 첫 10개의 플레이어 세션 레코드를 반환합니다.

```
// 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()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. 자세한 내용은 FlexMatch백필 기능을 참조하십시오.

이 작업은 비동기식입니다. 신규 플레이어가 매칭되면 Amazon은 콜백 함수를 사용하여 업데이트된 매치메이커 데이터를 GameLift 제공합니다. OnUpdateGameSession()

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

명령문

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

파라미터

StartMatchBackfillRequest

StartMatchBackfillRequest 객체에는 채우기 요청에 대한 정보가 들어 있습니다.

반환 값

매치 채우기 티켓 ID와 함께 the section called “StartMatchBackfillOutcome” 객체나 오류 메시지를 포함한 결함을 반환합니다.

예

```
// 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()

활성 매치 채우기 요청을 취소합니다. 자세한 내용은 백필 기능을 참조하십시오FlexMatch .

명령문

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

파라미터

StopMatchBackfillRequest

중단하려는 매치메이킹 티켓에 대한 세부 정보를 제공하는 StopMatchBackfillRequest 객체입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

게임 서버 및 게임 클라이언트 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. Amazon GameLift Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은, 을 참조하십시오 RegisterCompute.

명령문

```
GetComputeCertificateOutcome GetComputeCertificate();
```

반환 값

다음을 포함하는 GetComputeCertificateResponse 객체를 반환합니다.

- CertificatePath: 컴퓨팅 리소스의 TLS 인증서 경로. Amazon GameLift 관리형 플릿을 사용하는 경우 이 경로에는 다음이 포함됩니다.
  - `certificate.pem`: 최종 사용자 인증서입니다. 전체 인증서 체인은 이 인증서에 추가된 `certificateChain.pem`의 조합입니다.
  - `certificateChain.pem`: 루트 인증서와 중간 인증서를 포함하는 인증서 체인입니다.
  - `rootCertificate.pem`: 루트 인증서입니다.
  - `privateKey.pem`: 최종 사용자 인증서의 프라이빗 키입니다.
- ComputeName: 컴퓨팅 리소스의 이름.

예

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
 GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

GameLift Amazon이 다른 사람과 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. AWS 서비스 자세한 설명은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

구문

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
 request);
```

파라미터

GetFleetRoleCredentialsRequest

AWS 리소스에 대한 제한된 액세스를 게임 서버까지 확장하는 역할 자격 증명입니다.

반환 값

the section called “GetFleetRoleCredentialsOutcome” 객체를 반환합니다.

예

```
// form the fleet credentials request
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new
 GetFleetRoleCredentialsRequest(){
  RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"
};
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =
 GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Destroy()

Amazon GameLift 게임 서버 SDK를 메모리에서 비웁니다. `ProcessEnding()` 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 매 게임 세션 이후에 서버 프로세스를 종료하지 않는 경우, Amazon에 게임 세션을 호스팅할 준비가 되었음을 `InitSDK()` 알리기 전에 `Destroy()` GameLift 호출한 다음 다시 초기화하십시오. `ProcessReady()`

명령문

```
GenericOutcome Destroy()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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);
  }
```

C# 및 Unity용 Amazon GameLift Server SDK 참조: 데이터 유형

이 Amazon GameLift C# Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하고, Unity용 C# Server SDK 플러그인 사용에 대한 자세한 내용은 Amazon GameLift를 Unity 프로젝트에 통합 섹션을 참조하세요.

데이터 유형

- LogParameters
- ProcessParameters
- UpdateGameSession
- GameSession
- ServerParameters
- StartMatchBackfillRequest
- Player

- DescribePlayerSessionsRequest
- StopMatchBackfillRequest
- GetFleetRoleCredentialsRequest
- AttributeValue
- AwsStringOutcome
- GenericOutcome
- DescribePlayerSessionsOutcome
- DescribePlayerSessionsResult
- PlayerSessions
- StartMatchBackfillOutcome
- StartMatchBackfillResult
- GetComputeCertificateOutcome
- GetComputeCertificateResult
- GetFleetRoleCredentialsOutcome
- GetFleetRoleCredentialsResult
- AwsDateTimeOutcome
- GameLiftError
- Enums

LogParameters

이 데이터 유형을 사용하여 게임 세션이 종료된 후 게임 서버가 Amazon GameLift에 업로드할 게임 세션 중에 생성된 파일을 식별합니다. 게임 서버는 ProcessReady() 호출을 통해 `LogParameters to` Amazon GameLift에 전달합니다.

속성	설명
LogPaths	Amazon GameLift가 향후 사용을 위해 저장하도록 하려는 게임 서버 로그 파일의 디렉터리 경로의 목록입니다. 서버 프로세스는 각 게임 세션 중에 이러한 파일을 생성합니다. 게임 서버에서 파일 경로와 이름을 정의하고 이를 루트 게임 빌드 디렉터리에 저장합니다.

	로그 경로는 절대값이어야 합니다. 예를 들어, 게임 빌드가 `MyGame\sessionLogs\` 와 같은 경로에서 게임 세션 로그를 저장하면 이 경로는 Windows 인스턴스에서 `c:\game\MyGame\sessionLogs` 가 됩니다. 유형: `List<String>` 필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 ProcessReady() 호출 시 Amazon GameLift로 보낸 파라미터 집합이 포함됩니다.

속성	설명
LogParameters	게임 세션 로그 파일의 디렉터리 경로 목록을 포함하는 객체입니다. 유형: `Aws::GameLift::Server:: LogParameters` 필수 항목 여부: 예
OnHealthCheck	Amazon GameLift가 서버 프로세스에 상태 보고서를 요청하기 위해 호출하는 콜백 함수의 이름입니다. Amazon GameLift는 60초마다 이 함수를 호출합니다. 이 함수를 호출한 후 Amazon GameLift는 60초 동안 응답을 기다립니다. 아무것도 수신되지 않으면 Amazon GameLift가 서버 프로세스를 비정상으로 기록합니다. 유형: `void OnHealthCheckDelegate()` 필수 항목 여부: 예
OnProcessTerminate	Amazon GameLift가 서버 프로세스를 강제로 종료하기 위해 호출하는 콜백 함수 이름입니다. 이 함수를 호출한 후 Amazon GameLift는 서버

	프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답한 후 서버 프로세스를 종료합니다. 유형: `void OnProcessTerminate Delegate()` 필수 항목 여부: 예
OnStartGameSession	Amazon GameLift가 새 게임 세션을 활성화하기 위해 호출하는 콜백 함수 이름입니다. Amazon GameLift는 클라이언트 요청 CreateGameSession에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 GameSession 객체를 사용합니다(Amazon GameLift API 참조에서 정의함). 유형: `void OnStartGameSession Delegate( GameSession )` 필수 항목 여부: 예
OnUpdateGameSession	업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon GameLift가 호출하는 콜백 함수 이름입니다. Amazon GameLift는 업데이트된 매치메이커 데이터를 제공하기 위해 매치 채우기 요청을 처리할 때 이 함수를 호출합니다. GameSession 객체, 상태 업데이트(`updateReason`) 및 매치 채우기 티켓 ID를 전달합니다. 유형: void OnUpdateGameSessionDelegate (UpdateGameSession) 필수 항목 여부: 아니요

Port	서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다. 유형: `Integer` 필수 항목 여부: 예

UpdateGameSession

게임 세션 객체에 대한 정보가 업데이트되었습니다. 여기에는 게임 세션이 업데이트된 이유가 포함됩니다. 업데이트가 매치 채우기 작업과 관련된 경우 이 데이터 유형에는 채우기 티켓 ID가 포함됩니다.

속성	설명
GameSession	Amazon GameLift API에서 정의한 GameSession 객체입니다. `GameSession` 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 유형: `GameSession GameSession()` 필수 항목 여부: 예
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: `UpdateReason UpdateReason()` 필수 항목 여부: 예
BackfillTicketId	게임 세션 업데이트를 시도하는 채우기 티켓의 ID입니다. 유형: `String` 필수 항목 여부: 예

GameSession

게임 세션의 세부 정보입니다.

속성	설명
GameSessionId	게임 세션에 대한 고유 식별자입니다. 게임 세션 ARN의 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 유형: `String` 필수 항목 여부: 아니요
이름	게임 세션을 설명하는 레이블입니다. 유형: `String` 필수 항목 여부: 아니요
FleetId	게임 세션이 실행 중인 플릿의 고유 식별자입니다. 유형: `String` 필수 항목 여부: 아니요
MaximumPlayerSessionCount	게임 세션에 연결된 최대 플레이어 수입니다. 유형: `Integer` 필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `Integer`

속성	설명
	필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `String` 필수 항목 여부: 아니요
GameSessionData	단일 문자열 값으로 포맷된 사용자 지정 게임 세션 속성의 집합입니다. 유형: `String` 필수 항목 여부: 아니요
MatchmakerData	게임 세션을 생성하는 데 사용된 매치메이킹 프로세스에 대한 정보(JSON 구문 사용, 문자열 형식). 사용된 매치메이킹 구성 외에도 플레이어 속성 및 팀 배정을 포함하여 경기에 배정된 모든 플레이어에 대한 데이터가 포함됩니다. 유형: `String` 필수 항목 여부: 아니요
GameProperties	키:값 페어로 포맷된 게임 세션에 대한 사용자 정의 속성 집합입니다. 이러한 속성은 새 게임 세션을 시작하라는 요청과 함께 전달됩니다. 유형: `Dictionary<string, string>` 필수 항목 여부: 아니요

속성	설명
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. • TLS 지원 플릿: `<unique identifier>.<region identifier>.amazongamelift.com` • TLS 미지원 플릿: `ec2-<unique identifier>.compute.amazonaws.com` TLS 지원 플릿에서 실행되는 게임 세션에 연결할 때는 IP 주소가 아닌 DNS 이름을 사용해야 합니다. 유형: `String` 필수 항목 여부: 아니요

ServerParameters

Amazon GameLift Anywhere 서버와 Amazon GameLift 서비스 간의 연결을 유지하는 데 사용되는 정보입니다. 이 정보는 InitSDK()에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용합니다.

속성	설명
WebSocketUrl	Amazon GameLift Anywhere의 일부로 `RegisterCompute` 를 수행하면`GameLiftServerSdkEndpoint` 가 반환됩니다. 유형: `String` 필수 항목 여부: 예
ProcessId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다.

속성	설명
	유형: `String` 필수 항목 여부: 예
HostId	게임을 호스팅하는 서버 프로세스를 운영하는 호스트의 고유 식별자입니다. hostId는 컴퓨터를 등록할 때 사용되는 ComputeName입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: `String` 필수 항목 여부: 예
FleetId	컴퓨팅이 등록된 플릿의 플릿 ID입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: `String` 필수 항목 여부: 예
AuthToken	Amazon GameLift에서 생성한 인증 토큰으로, Amazon GameLift에 대한 서버를 인증합니다. 자세한 내용은 GetComputeAuthToken을 참조하세요. 유형: `String` 필수 항목 여부: 예

StartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 StartMatchBackfill() 호출을 통해 이 정보를 Amazon GameLift에 전달합니다.

속성	설명
GameSessionArn	고유한 게임 세션 식별자입니다. API 작업 GetGameSessionId 는 ARN 형식의 식별자를 반환합니다. 유형: String 필수 항목 여부: 예
MatchmakingConfigurationArn	매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션에 대한 매치메이커 ARN은 매치메이커 데이터 속성의 게임 세션 객체에 있습니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요. 유형: String 필수 항목 여부: 예
Players	현재 게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 유형: List<Player> 필수 항목 여부: 예
TicketId	매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 값을 제공하지 않으면 Amazon GameLift에서 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다. 유형: String 필수 항목 여부: 아니요

Player

매치메이킹 중인 플레이어를 나타냅니다. 매치메이킹 요청을 시작할 때 플레이어는 플레이어 ID, 속성, 지연 시간 데이터를 가지고 있을 수 있습니다. Amazon GameLift는 매치가 성사된 후 팀 정보를 추가합니다.

속성	설명
LatencyInMS	플레이어가 특정 위치에 연결되었을 때 발생하는 지연 시간을 나타내는 값 집합으로, 밀리초 단위로 표시됩니다. 이 속성을 사용하는 경우 플레이어는 나열된 위치에서만 매칭됩니다. 매치메이커에 플레이어 지연 시간을 평가하는 규칙이 있는 경우 플레이어는 지연 시간을 보고해야 매칭됩니다. 유형: `Dictionary<string, int>` 필수 항목 여부: 아니요
PlayerAttributes	매치메이킹에 사용할 플레이어 정보가 포함된 키:값 페어의 모음입니다. 플레이어 속성 키는 매치메이킹 규칙 세트에 사용되는 PlayerAttributes와 일치해야 합니다. 플레이어 속성에 대한 자세한 내용은 AttributeValue를 참조하세요. 유형: `Dictionary<string, AttributeValue` 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: `String` 필수 항목 여부: 아니요

속성	설명
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: `String` 필수 항목 여부: 아니요

DescribePlayerSessionsRequest

이 데이터 형식은 검색할 플레이어 세션을 지정하는 데 사용됩니다. 이 형식은 다음과 같이 여러 방식으로 사용될 수 있습니다. (1) 특정 플레이어 세션을 요청하는 PlayerSessionId를 제공, (2) 지정한 게임 세션에서 모든 플레이어 세션을 요청하는 GameSessionId를 제공 또는 (3) 지정한 플레이어의 모든 플레이어 세션을 요청하는 PlayerId를 제공. 플레이어 세션이 대량일 경우 페이지 매김 파라미터를 사용하여 결과를 순차 페이지로 검색합니다.

속성	설명
GameSessionId	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>` . <ID string> 값은 사용자 지정 ID 문자열(게임 세션을 만들 때 지정한 경우) 또는 생성 문자열입니다. 유형: `String` 필수 항목 여부: 아니요
PlayerSessionId	플레이어 세션의 고유 식별자입니다. 유형: `String` 필수 항목 여부: 아니요

속성	설명
PlayerId	플레이어의 고유 식별자입니다. 플레이어 ID 생성 섹션을 참조하세요. 유형: `String` 필수 항목 여부: 아니요
PlayerSessionStatusFilter	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다. • RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다. • ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 현재 연결되어 있습니다. • COMPLETED - 플레이어 연결이 끊어졌습니다. • TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다. 유형: `String` 필수 항목 여부: 아니요
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `String` 필수 항목 여부: 아니요

속성	설명
Limit	반환할 최대 결과 수입니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `int` 필수 항목 여부: 아니요

StopMatchBackfillRequest

매치메이킹 채우기 요청을 취소하는 데 사용되는 정보입니다. 게임 서버는 StopMatchBackfill() 호출을 통해 이 정보를 Amazon GameLift 서비스에 전달합니다.

속성	설명
GameSessionArn	취소 중인 요청의 고유한 게임 세션 식별자입니다. 유형: `string` 필수 항목 여부: 예
MatchmakingConfigurationArn	이 요청을 보낸 매치메이커의 고유 식별자입니다. 유형: `string` 필수 항목 여부: 예
TicketId	취소할 채우기 요청 티켓의 고유 식별자입니다. 유형: `string` 필수 항목 여부: 예

GetFleetRoleCredentialsRequest

이 데이터 유형을 사용하면 게임 서버가 다른 AWS 리소스에 제한적으로 액세스할 수 있습니다. 자세한 내용은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요.

속성	설명
RoleArn	AWS 리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `string` 필수 항목 여부: 예
RoleSessionName	역할 자격 증명의 사용을 설명하는 세션 이름입니다. 유형: `string` 필수 항목 여부: 아니요

AttributeValue

Player 속성 키-값 페어에 이러한 값을 사용합니다. 이 객체를 사용하면 문자열, 숫자, 문자열 배열 또는 데이터 맵과 같은 유효한 데이터 유형을 사용하여 속성 값을 지정할 수 있습니다. 각 `AttributeValue` 객체는 사용 가능한 속성 중 하나만 사용할 수 있습니다.

속성	설명
attrType	속성 값의 유형을 지정합니다. 형식: `AttrType` 열거형 값. 필수 항목 여부: 아니요
S	문자열 속성 값을 나타냅니다. 유형: `string`

속성	설명
	필수 항목 여부: 예
N	숫자 속성 값을 나타냅니다. 유형: `double` 필수 항목 여부: 예
SL	문자열 속성 값의 배열을 나타냅니다. 유형: `string[]` 필수 항목 여부: 예
SDM	문자열 키와 이중 값의 사전을 나타냅니다. 유형: `Dictionary<string, double>` 필수 항목 여부: 예

AwsStringOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `string` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다.

속성	설명
	유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GenericOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

DescribePlayerSessionsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “DescribePlayerSessionsResult” 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool

속성	설명
	필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

DescribePlayerSessionsResult

속성	설명
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `string` 필수 항목 여부: 예
PlayerSessions	요청과 일치하는 각 플레이어 세션의 속성을 포함하는 객체 모음입니다. 유형: `IList<`the section called “PlayerSessions” `>` 필수 항목 여부:
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError”

속성	설명
	필수 항목 여부: 아니요

PlayerSessions

속성	설명
CreationTime	유형: `long` 필수 항목 여부: 예
FleetId	유형: `string` 필수 항목 여부: 예
GameSessionId	유형: `string` 필수 항목 여부: 예
IpAddress	유형: `string` 필수 항목 여부: 예
PlayerData	유형: `string` 필수 항목 여부: 예
PlayerId	유형: `string` 필수 항목 여부: 예
PlayerSessionId	유형: `string` 필수 항목 여부: 예
Port	유형: `int` 필수 항목 여부: 예
Status	유형: `PlayerSessionStatus` 열거형.

속성	설명
	필수 항목 여부: 예
TerminationTime	유형: `long` 필수 항목 여부: 예
DnsName	유형: `string` 필수 항목 여부: 예

StartMatchBackfillOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “StartMatchBackfillResult” 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

StartMatchBackfillResult

속성	설명
TicketId	유형: `string` 필수 항목 여부: 예

GetComputeCertificateOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “GetComputeCertificateResult” 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GetComputeCertificateResult

컴퓨팅의 TLS 인증서 경로 및 컴퓨팅 호스트 이름입니다.

속성	설명
CertificatePath	유형: `string` 필수 항목 여부: 예
ComputeName	유형: `string` 필수 항목 여부: 예

GetFleetRoleCredentialsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “GetFleetRoleCredentialsResult” 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GetFleetRoleCredentialsResult

속성	설명
AccessKeyId	인증하고 AWS 리소스에 액세스를 제공하기 위한 액세스 키 ID입니다. 유형: `string` 필수 항목 여부: 아니요
AssumedRoleId	서비스 역할이 속한 사용자의 ID입니다. 유형: `string` 필수 항목 여부: 아니요
AssumedRoleUserArn	사용자가 맡을 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `string` 필수 항목 여부: 아니요
Expiration	세션 자격 증명이 만료될 때까지 남은 시간입니다. 유형: `DateTime` 필수 항목 여부: 아니요
SecretAccessKey	인증에 대한 비밀 액세스 키 ID를 지정합니다. 유형: `string` 필수 항목 여부: 아니요
SessionToken	AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하는 토큰입니다. 유형: `string`

속성	설명
	필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

AwsDateTimeOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `DateTime` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: `GameLiftErrorType` 열거형. 필수 항목 여부: 아니요
ErrorName	오류 유형 이름입니다. 유형: `string` 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: `string` 필수 항목 여부: 아니요

Enums

Amazon GameLift Server SDK(C#)에 대해 정의된 Enums는 다음과 같이 정의됩니다.

AttrType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

오류 유형을 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- SERVICE_CALL_FAILED - AWS 서비스 호출이 실패했습니다.
- LOCAL_CONNECTION_FAILED - Amazon GameLift에 대한 로컬 연결이 실패했습니다.
- NETWORK_NOT_INITIALIZED - 네트워크가 초기화되지 않았습니다.

- GAMESESSION_ID_NOT_SET - 게임 세션 ID가 설정되지 않았습니다.
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION
- ALREADY_INITIALIZED - Amazon GameLift 서버 또는 클라이언트는 이미 Initialize()를 사용하여 초기화되었습니다.
- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALIZED - Amazon GameLift 클라이언트가 초기화되지 않았습니다.
- GAMELIFT_SERVER_NOT_INITIALIZED - Amazon GameLift 서버가 초기화되지 않았습니다.
- GAME_SESSION_ENDED_FAILED - Amazon GameLift Server SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY - Amazon GameLift Server Game Session이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED - Amazon GameLift Server SDK가 서비스에 연락하여 게임 세션이 준비되었음을 보고할 수 없습니다.
- INITIALIZATION_MISMATCH - 클라이언트 메서드는 Server::Initialize() 이후에 호출되었으며 그 반대의 경우도 마찬가지입니다.
- NOT_INITIALIZED - Amazon GameLift 서버 또는 클라이언트는 Initialize()를 사용하여 초기화되었습니다.
- NO_TARGET_ALIASID_SET - 대상 aliasId가 설정되지 않았습니다.
- NO_TARGET_FLEET_SET - 대상 플릿이 설정되지 않았습니다.
- PROCESS_ENDING_FAILED - Amazon GameLift Server SDK가 서비스에 연락하여 프로세스가 종료되었음을 보고할 수 없습니다.
- PROCESS_NOT_ACTIVE - 서버 프로세스가 아직 활성화되지 않았고, GameSession에 바인딩되지 않았으며, PlayerSessions를 수락하거나 처리할 수 없습니다.
- PROCESS_NOT_READY - 서버 프로세스를 아직 활성화할 준비가 되지 않았습니다.
- PROCESS_READY_FAILED – Amazon GameLift Server SDK가 서비스에 연락하여 프로세스가 준비되었음을 보고할 수 없습니다.
- SDK_VERSION_DETECTION_FAILED - SDK 버전 감지에 실패했습니다.
- STX_CALL_FAILED - XStx 서버 백엔드 구성 요소에 대한 호출이 실패했습니다.
- STX_INITIALIZATION_FAILED - XStx 서버 백엔드 구성 요소를 초기화하지 못했습니다.

- UNEXPECTED_PLAYER_SESSION - 서버에서 등록되지 않은 플레이어 세션을 발견했습니다.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE - GameLift Service WebSocket에 메시지를 보내는 재시도에 실패했습니다.
- WEBSOCKET_SEND_MESSAGE_FAILURE - GameLift Service WebSocket에 메시지를 보내는 데 실패했습니다.
- MATCH_BACKFILL_REQUEST_VALIDATION - 요청 검증에 실패했습니다.
- PLAYER_SESSION_REQUEST_VALIDATION - 요청 검증에 실패했습니다.

PlayerSessionCreationPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.
- NOT_SET - 게임 세션이 새 플레이어 세션을 수락하거나 거부하도록 설정되지 않았습니다.

플레이어 세션 상태

- ACTIVE
- COMPLETED
- NOT_SET
- RESERVED
- TIMEDOUT

C#용 Amazon GameLift Server SDK 4.x참조

이 Amazon GameLift C# Server SDK 4.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- Amazon GameLift Server SDK(C#) 참조: 작업

- Amazon GameLift Server SDK(C#) 참조: 데이터 유형

Amazon GameLift Server SDK(C#) 참조: 작업

이 Amazon GameLift C# Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

- 작업
- 데이터 형식

AcceptPlayerSession()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift가 플레이어 세션 ID가 유효한지, 즉 플레이어 ID가 게임 세션의 플레이어 슬롯을 예약했는지 확인합니다. 검증되면 Amazon GameLift가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift가 발행한 고유 ID입니다. 플레이어 세션 ID는 `PlayerSession` 객체에서 지정되며, 이 객체는 GameLift API 작업 StartGameSessionPlacement, CreateGameSession, DescribeGameSessionPlacement 또는 DescribePlayerSessions에 대한 클라이언트 호출에 응답하여 반환됩니다.

유형: 문자열

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 잘못된 플레이어 세션 ID 검증 및 거부를 비롯해 연결 요청을 처리하기 위한 함수를 보여줍니다.

```
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()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화가 완료된 후 onStartGameSession() 콜백 함수의 일부로 호출되어야 합니다.

구문

```
GenericOutcome ActivateGameSession()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 onStartGameSession() 위임 함수의 일부로 ActivateGameSession()이 호출되는 것을 보여줍니다.

```
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()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 작업을 사용하면 단일 플레이어 세션 정보, 게임 세션에 있는 모든 플레이어 세션 정보 또는 단일 플레이어 ID와 관련된 모든 플레이어 세션 정보를 가져올 수 있습니다.

구문

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
 describePlayerSessionsRequest)
```

파라미터

describePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 DescribePlayerSessionsRequest 객체입니다.

필수 항목 여부: 예

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 `DescribePlayerSessionsOutcome` 객체를 반환합니다. 플레이어 세션 객체 구조는 AWS SDK Amazon GameLift API PlayerSession 데이터 유형과 동일합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. NextToken을 생략하고 제한 값을 10으로 설정하면, Amazon GameLift가 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 반환합니다.

```
// 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()

서버 프로세스가 활성인 경우 현재 서버 프로세스에서 호스팅하고 있는 게임 세션의 ID를 가져옵니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 `Success=True` 및 `GameSessionId=""`(빈 문자열)를 반환합니다.

구문

```
AwsStringOutcome GetGameSessionId()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 `AwsStringOutcome` 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

예

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

플릿 및 해당 인스턴스와 연결된 pem 인코딩된 TLS 인증서의 파일 위치를 검색합니다. AWS Certificate Manager는 인증서 구성이 GENERATED로 설정된 새 플릿을 생성할 때 이 인증서를 생성합니다. 이 인증서를 사용하여 게임 클라이언트와 보안 연결을 설정하고 클라이언트/서버 통신을 암호화합니다

구문

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 인스턴스에 저장된 플릿의 TLS 인증서 파일과 인증서 체인 위치를 포함한 `GetInstanceCertificateOutcome` 객체를 반환합니다. 인증서 체인에서 추출한 루트 인증서 파일도 인스턴스에 저장됩니다. 실패하면 오류 메시지를 반환합니다.

인증서 및 인증서 체인 데이터에 대한 자세한 내용은 AWS Certificate Manager API 참조의 GetCertificate 응답 요소를 참조하세요.

예

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
AwsStringOutcome GetSdkVersion()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 현재 SDK 버전을 `AwsStringOutcome` 객체로 반환합니다. 반환된 문자열에는 버전 번호만(예: “3.1.5”)을 포함합니다. 실패하면 오류 메시지를 반환합니다.

예

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift 서비스에서 onProcessTerminate() 콜백을 수신한 후 이 작업을 수행합니다. Amazon GameLift는 다음 이유로 onProcessTerminate()를 호출할 수 있습니다. (1) 상태 불량의 경우(서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift에 응답하지 않은 경우, (2) 스케일다운 이벤트 중에 인스턴스를 종료하는 경우 또는 (3) 스팟 인스턴스 중단으로 인해 인스턴스가 종료되는 경우.

프로세스가 onProcessTerminate() 콜백을 수신할 때 반환되는 값은 예상 종료 시간입니다. 프로세스가 onProcessTerminate() 콜백을 받지 못한 경우 오류 메시지가 반환됩니다. 서버 프로세스 종료에 대해 자세히 알아보세요.

구문

```
AwsDateTimeOutcome GetTerminationTime()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 종료 시간을 AwsDateTimeOutcome 객체로 반환합니다. 값은 종료 시간이며, 0001 00:00:00 이후 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값인 2020-09-13 12:26:40 -000Z는 637355968000000000 틱과 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

Amazon GameLift SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift 관련 초기화가 진행되기 전, 시작 시 호출되어야 합니다.

구문

```
InitSDKOutcome InitSDK()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하는 경우, 서버 프로세스가 ProcessReady()를 호출할 준비가 되었음을 나타내는 InitSdkOutcome 객체를 반환합니다.

예

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

ProcessEnding()

Amazon GameLift 서비스에 서버 프로세스를 중단할 준비가 되었음을 알립니다. 이 메서드는 모든 활성 게임 세션의 종료 등, 다른 모든 정리 작업 이후에 호출되어야 합니다. 이 메서드는 종료 코드 0으로 종료해야 합니다. 0이 아닌 종료 코드를 사용하면 프로세스가 정상적으로 종료되지 않았다는 이벤트 메시지가 표시됩니다.

코드를 0으로 메서드가 종료되면 종료 코드를 성공시켜 프로세스를 종료할 수 있습니다. 오류 코드를 표시하여 프로세스를 종료할 수도 있습니다. 오류 코드를 남기고 종료하면 플릿 이벤트는 프로세스가 비정상적으로 종료(SERVER_PROCESS_TERMINATED_UNHEALTHY)되었음을 나타냅니다.

구문

```
GenericOutcome ProcessEnding()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
```

```
   Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 InitSDK()를 호출하고, 서버 프로세스가 게임 세션을 호스트하기 전에 수행해야 하는 설정 작업을 완료한 후에만 이 메서드를 호출할 수 있습니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

구문

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- 게임 서버 코드에 구현되어 있는 콜백 메서드의 이름으로, 서버 프로세스와 통신할 수 있도록 Amazon GameLift 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 ProcessReady() 호출 및 위임 함수 구현을 모두 보여줍니다.

```
// 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()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift가 발행한 고유 ID입니다. 플레이어 세션 ID는 PlayerSession 객체에서 지정되며, 이 객체는 GameLift API 작업 StartGameSessionPlacement, CreateGameSession, DescribeGameSessionPlacement 또는 DescribePlayerSessions에 대한 클라이언트 호출에 응답하여 반환됩니다.

유형: 문자열

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
Aws::GameLift::GenericOutcome disconnectOutcome =
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. AWS SDK 작업 StartMatchBackfill()도 참조하세요. 이 작업을 사용하면 게임 세션을 호스팅하는 게임 서버 프로세스에서 매치 채우기 요청을 시작할 수 있습니다. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift 서비스는 콜백 함수 OnUpdateGameSession()을 사용하여 업데이트된 매치메이커 데이터를 전달합니다.

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

구문

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
 startBackfillRequest);
```

파라미터

StartMatchBackfillRequest

다음 정보를 전달하는 StartMatchBackfillRequest 객체입니다.

- 백필(backfill) 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift가 ID를 자동 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에서 수집할 수 있습니다.
- 백필(backfill) 중인 게임 세션의 ID.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터.

필수 항목 여부: 예

반환 값

매치 채우기 티켓 ID와 함께 StartMatchBackfillOutcome 객체를 반환하거나 오류 메시지를 포함한 결함이 있는 StartMatchBackfillOutcome 객체를 반환합니다.

예

```
// 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()

StartMatchBackfill()을 통해 생성된 활성 매치 채우기 요청을 취소합니다. AWS SDK 작업 StopMatchmaking()도 참조하세요. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

구문

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

파라미터

StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체입니다.

- 취소 중인 채우기 요청에 할당한 티켓 ID
- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
// 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()

버전 4.0.1 에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 ProcessEnding()을 호출해야 합니다.

Amazon GameLift 서비스에 서버 프로세스가 현재 게임 세션을 종료했음을 알립니다. 이 작업은 서버 프로세스가 활성 상태를 유지하고 새 게임 세션을 호스팅할 준비가 되면 호출됩니다. 서버 프로세스를 즉시 사용하여 새 게임 세션을 호스팅할 수 있음을 Amazon GameLift에 알리기 때문에 게임 세션 종료 절차가 완료된 후에만 호출해야 합니다.

게임 세션이 중지된 후 서버 프로세스가 종료되는 경우에는 이 작업이 호출되지 않습니다. 대신 ProcessEnding()을 호출하여 게임 세션과 서버 프로세스가 모두 종료되었음을 알립니다.

구문

```
GenericOutcome TerminateGameSession()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 게임 세션을 마치는 서버 프로세스를 보여줍니다.

```
// 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()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다. (Amazon GameLift Service API 참조의 UpdateGameSession() 작업도 참조하세요.)

구문

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
 playerSessionPolicy)
```

파라미터

newPlayerSessionPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다.

유형: PlayerSessionCreationPolicy 열거형. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
var updatePlayerSessionCreationPolicyOutcomex =

 GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

Amazon GameLift Server SDK(C#) 참조: 데이터 유형

이 Amazon GameLift C# Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

- 작업
- 데이터 유형

LogParameters

이 데이터 유형은 게임 세션 중에 생성된 파일 가운데 게임 세션 종료 시 Amazon GameLift가 업로드 및 저장하도록 하려는 파일을 식별합니다. 이 정보는 ProcessReady() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

logPaths

Amazon GameLift가 향후 사용을 위해 저장하도록 하려는 게임 서버 로그 파일의 디렉터리 경로의 목록입니다. 이들 파일은 각 게임 세션 도중 서버 프로세스에 의해 생성됩니다. 파일 경로 및 이름은 게임 서버에서 정의되고 루트 게임 빌드 디렉터리에 저장됩니다. 로그 경로는 절대값이어야 합니다. 예를 들어, 게임 빌드가 `MyGame\sessionlogs\`와 같은 경로에 게임 세션 로그를 저장할 경우 로그 경로는 `c:\game\MyGame\sessionLogs`(Windows 인스턴스) 또는 `/local/game/MyGame/sessionLogs`(Linux 인스턴스)가 됩니다.

유형: List<String>

필수 항목 여부: 아니요

DescribePlayerSessionsRequest

이 데이터 형식은 검색할 플레이어 세션을 지정하는 데 사용됩니다. 이 형식은 다음과 같이 여러 방식으로 사용될 수 있습니다. (1) 특정 플레이어 세션을 요청하는 PlayerSessionId를 제공, (2) 지정한 게임 세션에서 모든 플레이어 세션을 요청하는 GameSessionId를 제공 또는 (3) 지정한 플레이어의 모든 플레이어 세션을 요청하는 PlayerId를 제공. 플레이어 세션이 대량일 경우 페이지 매김 파라미터를 사용하여 결과를 순차 페이지로 검색합니다.

목차

GameSessionId

고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. <ID string> 값은 사용자 지정 ID 문자열(게임 세션을 만들 때 지정한 경우) 또는 생성 문자열입니다.

유형: 문자열

필수 항목 여부: 아니요

Limit

반환할 최대 결과 수입니다. 결과를 순차적인 일련의 페이지로 가져오려면 이 파라미터를 NextToken과 함께 사용합니다. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 정수

필수 항목 여부: 아니요

NextToken

결과의 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마십시오. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerId

사용자의 고유 식별자입니다. 플레이어 ID는 개발자에 의해 정의됩니다. 플레이어 ID 생성 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionId

플레이어 세션의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionStatusFilter

결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다.

- RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다.

- ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 현재 연결되어 있습니다.
- COMPLETED - 플레이어 연결이 끊어졌습니다.
- TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다.

유형: 문자열

필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 ProcessReady() 호출 시 Amazon GameLift 서비스로 보낸 파라미터 집합이 포함됩니다.

목차

port

서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다.

유형: 정수

필수 항목 여부: 예

logParameters

게임 세션 로그 파일의 디렉터리 경로 목록을 포함하는 객체입니다.

유형: `Aws::GameLift::Server::LogParameters`

필수 항목 여부: 예

onStartGameSession

Amazon GameLift가 새 게임 세션을 활성화하기 위해 호출하는 콜백 함수 이름입니다. Amazon GameLift는 클라이언트 요청 CreateGameSession에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 GameSession 객체를 사용합니다(Amazon GameLift Service API 참조에서 정의함).

유형: `void OnStartGameSessionDelegate(GameSession gameSession)`

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift 서비스가 서버 프로세스를 강제로 종료하기 위해 호출하는 콜백 함수 이름입니다. 이 함수를 호출한 후 Amazon GameLift는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답한 후 서버 프로세스를 종료합니다.

유형: `void OnProcessTerminateDelegate()`

필수 항목 여부: 예

onHealthCheck

Amazon GameLift 서비스가 서버 프로세스에 상태 보고서를 요청하기 위해 호출하는 콜백 함수의 이름입니다. Amazon GameLift는 60초마다 이 함수를 호출합니다. 이 함수를 호출한 후 Amazon GameLift는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상으로 기록합니다.

유형: `bool OnHealthCheckDelegate()`

필수 항목 여부: 예

onUpdateGameSession

업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon GameLift 서비스가 호출하는 콜백 함수 이름입니다. Amazon GameLift는 업데이트된 매치메이커 데이터를 제공하기 위해 매치 채우기 요청을 처리할 때 이 함수를 호출합니다. GameSession 객체, 상태 업데이트 (`updateReason`) 및 매치 채우기 티켓 ID를 전달합니다.

유형: `void OnUpdateGameSessionDelegate ( UpdateGameSession updateGameSession )`

필수 항목 여부: 아니요

StartMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 이 정보는 StartMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

고유한 게임 세션 식별자입니다. GetGameSessionId() SDK 메서드는 ARN 형식의 식별자를 반환합니다.

유형: 문자열

필수 항목 여부: 예

MatchmakingConfigurationArn

매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션을 만드는 데 사용된 매치메이커를 찾으려면 게임 세션 객체에서 매치메이커 데이터 속성을 확인합니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요.

유형: 문자열

필수 항목 여부: 예

Players

현재 게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 플레이어 객체 형식에 대한 자세한 내용은 Amazon GameLift API 참조 가이드를 참조하세요. 플레이어 속성, ID 및 팀 배정을 찾으려면 매치메이커 데이터 속성에서 게임 세션 객체를 확인합니다. 매치메이커에서 지연 시간을 사용하는 경우 현재 리전에 대한 업데이트 지연 시간을 수집하여 각 플레이어의 데이터에 포함합니다.

유형: 플레이어[ ]

필수 항목 여부: 예

TicketId

매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 여기에 값이 제공되지 않으면 Amazon GameLift는 UUID 형식으로 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다.

유형: 문자열

필수 항목 여부: 아니요

StopMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 취소하는 데 사용됩니다. 이 정보는 StopMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

취소 중인 요청과 연결된 고유한 게임 세션 식별자입니다.

유형: 문자열

필수 항목 여부: 예

MatchmakingConfigurationArn

이 요청을 보낸 매치메이커의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 예

TicketId

취소할 채우기 요청 티켓의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 예

Go용 Amazon GameLift Server SDK 참조

이 Amazon GameLift Go Server SDK 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- 아마존 GameLift 서버 SDK (Go) 참조: 액션
- 아마존 GameLift 서버 SDK (Go) 참조: 데이터 유형

아마존 GameLift 서버 SDK (Go) 참조: 액션

이 Amazon GameLift Go 서버 SDK 참조를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

`GameLiftServerAPI.go`는 Go 서버 SDK 작업을 정의합니다.

작업

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
func GetSdkVersion() (string, error)
```

반환 값

성공하면 현재 SDK 버전을 문자열로 반환합니다. 반환된 문자열에는 버전 번호(예: `5.0.0`)가 포함됩니다. 실패하면 오류 메시지(예: `common.SdkVersionDetectionFailed`)를 반환합니다.

예

```
version, err := server.GetSdkVersion()
```

InitSDK()

아마존 GameLift SDK를 초기화합니다. GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 메서드는 서버와 Amazon GameLift 서비스 간의 통신을 설정합니다.

명령문

```
func InitSDK(params ServerParameters) error
```

파라미터

ServerParameters

Amazon GameLift Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 생성하십시오.

- 게임 서버에 연결하는 데 WebSocket 사용되는 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Anywhere 컴퓨팅을 포함하는 아마존 GameLift 플릿의 ID입니다.
- Amazon GameLift 작업에서 생성된 인증 토큰입니다.

Amazon GameLift 관리형 EC2 플릿에서 게임 서버를 초기화하려면 파라미터 없이 `ServerParameters` 객체를 생성하십시오. 이 호출을 통해 Amazon GameLift 에이전트는 컴퓨팅 환경을 설정하고 자동으로 Amazon GameLift 서비스에 연결합니다.

반환 값

성공하는 경우, 서버 프로세스가 ProcessReady()를 호출할 준비가 되었음을 나타내는 `nil` 오류를 반환합니다.

> **Note**
>
> Anywhere 플릿에 배포된 게임 빌드에 대한 `InitSDK()`로의 호출이 실패하는 경우 빌드 리소스를 생성할 때 사용된 `ServerSdkVersion` 파라미터를 확인합니다. 명시적으로 이 값을 사용 중인 Server SDK 버전으로 설정해야 합니다. 이 파라미터의 기본값은 4.x이며 호환되지 않습니다. 이 문제를 해결하려면 새 빌드를 생성하여 새 플릿에 배포해야 합니다.

예

아마존 GameLift Anywhere 예시

```
//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)
```

아마존 GameLift 관리형 EC2 예제

```
//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()

서버 프로세스가 게임 세션을 호스팅할 준비가 GameLift 되었음을 Amazon에 알립니다. InitSDK() 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

명령문

```
func ProcessReady(param ProcessParameters) error
```

파라미터

ProcessParameters

다음 서버 프로세스 관련 정보를 전달하는 ProcessParameters 객체입니다.

- Amazon GameLift 서비스가 서버 프로세스와 통신하기 위해 호출하는 게임 서버 코드에 구현된 콜백 메서드의 이름.
- 서버 프로세스가 수신하는 포트 번호입니다.
- GameLift Amazon에서 캡처하여 저장하려는 게임 세션별 파일의 경로가 포함된 LogParameters 데이터 유형입니다.

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다. 메서드가 성공한 경우 `nil`를 반환합니다.

예

이 예에서는 ProcessReady() 호출 및 위임 함수 구현을 모두 보여줍니다.

```
// 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()

서버 프로세스가 종료되고 GameLift 있음을 Amazon에 알립니다. 다른 모든 정리 작업 (활성 게임 세션 종료 포함) 이 끝난 후 프로세스를 종료하기 전에 이 메서드를 호출합니다. `ProcessEnding()`의 결과에 따라 프로세스가 성공(0) 또는 오류(-1)로 종료되고 플릿 이벤트가 생성됩니다. 오류가 발생하여 프로세스가 종료되는 경우 생성되는 플릿 이벤트는 입니다. `SERVER_PROCESS_TERMINATED_UNHEALTHY`

명령문

```
func ProcessEnding() error
```

반환 값

0 오류 코드 또는 정의된 오류 코드를 반환합니다.

예

```
// 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()

서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 GameLift 되었음을 Amazon에 알립니다. 이 작업은 모든 게임 세션 초기화 후 onStartGameSession() 콜백 함수의 일부로 호출됩니다.

명령문

```
func ActivateGameSession() error
```

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다.

예

이 예에서는 onStartGameSession() 위임 함수의 일부로 ActivateGameSession()이 호출되는 것을 보여줍니다.

```
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()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다.

명령문

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

파라미터

playerSessionCreation정책

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다.

유효한 값으로는 다음이 포함됩니다.

- **`model.AcceptAll`** - 모든 새 플레이어 세션을 수락합니다.
- **`model.DenyAll`** - 모든 새 플레이어 세션을 거부합니다.

반환 값

오류가 발생하면 오류 메시지와 함께 오류를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

활성 서버 프로세스가 호스팅된 게임 세션의 ID를 가져옵니다.

명령문

```
func GetGameSessionID() (string, error)
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID 및 nil 오류를 반환합니다. 게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 빈 문자열과 `nil` 오류를 반환합니다.

예

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 GameLift Amazon으로부터 `onProcessTerminate()` 콜백을 받은 후 이 작업을 수행합니다. Amazon에서 GameLift 전화를 거는 `onProcessTerminate()` 이유는 다음과 같습니다.

- 서버 프로세스가 상태가 좋지 않다고 보고되었거나 GameLift Amazon에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- 스팟 인스턴스 중단으로 인해 인스턴스가 종료되는 경우

명령문

```
func GetTerminationTime() (int64, error)
```

반환 값

성공하면 서버 프로세스가 종료될 예정인 타임스탬프(Epoch 초) 및 `nil` 오류 종료를 반환합니다. 값은 종료 시간이며, `0001 00:00:00`에서 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값 `2020-09-13 12:26:40 -000Z`는 `637355968000000000` 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되었으며 검증이 GameLift 필요함을 Amazon에 알립니다. Amazon은 플레이어 세션 ID가 유효한지 GameLift 확인합니다. 플레이어 세션이 검증된 후 Amazon은 플레이어 슬롯의 상태를 에서 로 GameLift `RESERVED` 변경합니다. `ACTIVE`

명령문

```
func AcceptPlayerSession(playerSessionID string) error
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제는 유효하지 않은 플레이어 세션 ID의 검증 및 거부를 포함하는 연결 요청을 처리합니다.

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {
    err := server.AcceptPlayerSession(playerSessionID)
    if err != nil {
        connection.Accept()
    } else {
        connection.Reject(err.Error())
    }
}
```

RemovePlayerSession()

플레이어가 서버 프로세스와의 연결이 GameLift 끊겼음을 Amazon에 알립니다. 이에 대해 Amazon은 플레이어 슬롯을 사용 가능한 것으로 GameLift 변경합니다.

명령문

```
func RemovePlayerSession(playerSessionID string) error
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 메서드를 사용하여 다음에 대한 정보를 얻을 수 있습니다.

- 단일 플레이어 세션
- 게임 세션의 모든 플레이어 세션
- 단일 플레이어 ID와 연결된 모든 플레이어 세션

명령문

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)
 (result.DescribePlayerSessionsResult, error) {
 return srv.describePlayerSessions(&req)
}
```

파라미터

DescribePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 `DescribePlayerSessionsRequest` 객체입니다.

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 `DescribePlayerSessionsResult` 객체를 반환합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. 제한 값을 NextToken생략하고 10으로 설정하면 Amazon은 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 GameLift 반환합니다.

```
// 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()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. 자세한 내용은 FlexMatch 백필 기능을 참조하십시오.

이 작업은 비동기식입니다. 신규 플레이어가 매칭되면 Amazon은 콜백 함수를 사용하여 업데이트된 매치메이커 데이터를 GameLift 제공합니다. `OnUpdateGameSession()`

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

명령문

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
 (result.StartMatchBackfillResult, error)
```

파라미터

StartMatchBackfillRequest

StartMatchBackfillRequest 객체는 다음 정보를 전달합니다.

- 채우기 요청에 할당할 티켓 ID. 이 정보는 선택 사항입니다. ID를 제공하지 않으면 Amazon에서 GameLift 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에 있습니다.
- 채울 게임 세션의 ID입니다.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터입니다.

반환 값

매치 채우기 티켓 ID와 함께 StartMatchBackfillResult 객체나 오류 메시지를 포함한 결함을 반환합니다.

예

```
// 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()

활성 매치 채우기 요청을 취소합니다. 자세한 내용은 FlexMatch백필 기능을 참조하십시오.

명령문

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

파라미터

StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체:

- 채우기 요청에 할당된 티켓 ID

- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
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()

게임 서버 및 게임 클라이언트 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. Amazon GameLift Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은 을 참조하십시오 RegisterCompute.

명령문

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

반환 값

다음을 포함하는 `GetComputeCertificateResult` 객체를 반환합니다.

- CertificatePath: 컴퓨팅 리소스의 TLS 인증서 경로. Amazon GameLift 관리형 플릿을 사용하는 경우 이 경로에는 다음이 포함됩니다.
  - `certificate.pem`: 최종 사용자 인증서입니다. 전체 인증서 체인은 이 인증서에 추가된 `certificateChain.pem`의 조합입니다.
  - `certificateChain.pem`: 루트 인증서와 중간 인증서를 포함하는 인증서 체인입니다.
  - `rootCertificate.pem`: 루트 인증서입니다.

- `privateKey.pem`: 최종 사용자 인증서의 프라이빗 키입니다.
- ComputeName: 컴퓨팅 리소스의 이름.

예

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

GameLiftAmazon에 대한 다른 사용자의 권한을 확장하기 위해 생성한 서비스 역할 자격 증명을 AWS 서비스 검색합니다. 이러한 자격 증명을 통해 게임 서버는 AWS 리소스를 사용할 수 있습니다. 자세한 설명은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요.

구문

```
func GetFleetRoleCredentials(
  req request.GetFleetRoleCredentialsRequest,
) (result.GetFleetRoleCredentialsResult, error) {
  return srv.getFleetRoleCredentials(&req)
}
```

파라미터

GetFleetRoleCredentialsRequest

AWS 리소스에 대한 제한된 액세스를 게임 서버까지 확장하는 역할 자격 증명입니다.

반환 값

다음을 포함하는 `GetFleetRoleCredentialsResult` 객체를 반환합니다.

- AssumedRoleUserArn - 서비스 역할이 속한 사용자의 Amazon 리소스 이름 (ARN)
- AssumedRoleId - 서비스 역할이 속한 사용자의 ID.
- AccessKeyId - AWS 리소스에 대한 액세스를 인증하고 제공하기 위한 액세스 키 ID입니다.
- SecretAccessKey - 인증을 위한 비밀 액세스 키 ID.
- SessionToken - AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하는 토큰입니다.
- Expiration - 세션 자격 증명이 만료될 때까지 남은 시간입니다.

예

```
// form the customer credentials request
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction"

credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Destroy()

Amazon GameLift 게임 서버 SDK를 메모리에서 비웁니다. `ProcessEnding()` 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 매 게임 세션 이후에 서버 프로세스를 종료하지 않는 경우, Amazon에 게임 세션을 호스팅할 준비가 되었음을 `InitSDK()` 알리기 전에 `Destroy()` GameLift 호출한 다음 다시 초기화하십시오. `ProcessReady()`

명령문

```
func Destroy() error {
 return srv.destroy()
}
```

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다.

예

```
// 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)
  }
}
```

아마존 GameLift 서버 SDK (Go) 참조: 데이터 유형

이 Amazon GameLift Go 서버 SDK 참조를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

데이터 타입

- LogParameters
- ProcessParameters
- UpdateGameSession
- GameSession
- ServerParameters
- StartMatchBackfillRequest
- Player
- DescribePlayerSessionsRequest
- StopMatchBackfillRequest
- GetFleetRoleCredentialsRequest

LogParameters

게임 세션 중에 생성된 파일을 식별하는 객체로, Amazon에서 게임 세션 종료 후 업로드하고 GameLift 저장하도록 합니다. 게임 서버는 ProcessReady() 호출 시 `ProcessParameters` 객체의 GameLift 일부로 `LogParameters` Amazon에 제공합니다.

속성	설명
`LogPaths`	Amazon에서 나중에 액세스할 수 있도록 GameLift 저장하려는 게임 서버 로그 파일의 디렉터리 경로 목록입니다. 서버 프로세스는 각 게임 세션 중에 이러한 파일을 생성합니다. 게임 서버에서 파일 경로와 이름을 정의하고 이를 루트 게임 빌드 디렉터리에 저장합니다. 로그 경로는 절대값이어야 합니다. 예를 들어, 게임 빌드가 `MyGame\sessionLogs\` 와 같은 경로에서 게임 세션 로그를 저장하면 이 경로는 Windows 인스턴스에서 `c:\game\MyGame\sessionLogs` 가 됩니다.

	유형: `[]string` 필수 항목 여부: 아니요

ProcessParameters

서버 프로세스와 Amazon GameLift 간의 통신을 설명하는 객체입니다. 서버 프로세스는 를 GameLift 호출하여 Amazon에 이 정보를 제공합니다ProcessReady().

속성	설명
`LogParameters`	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon은 나중에 액세스할 수 있도록 파일을 GameLift 복사하고 저장합니다. 유형: `LogParameters` 필수 항목 여부: 아니요
`OnHealthCheck`	Amazon이 서버 프로세스에 상태 보고서를 요청하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 60초마다 이 함수를 GameLift 호출하고 응답을 받을 때까지 60초 동안 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon은 서버 프로세스를 정상이 아닌 것으로 GameLift 기록합니다. 유형: `OnHealthCheck func() bool` 필수 항목 여부: 아니요
`OnProcessTerminate`	Amazon이 서버 프로세스를 강제로 종료하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 이 함수를 호출한 후 서버 프로세스가 종료될 때까지 5분간 GameLift 기다린 후 ProcessEnding() 호출에 응답한 후 서버 프로세스를 종료합니다. 유형: `OnProcessTerminate func()` 필수 항목 여부: 예
`OnStartGameSession`	업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 업데이트된 매치메이커 데이터를 제공하기 위해 매치 백필 요청이 처리되면 이 함수를 GameLift 호출합니

	다. GameSession객체, 상태 업데이트 (updateReason), 매치 백필 티켓 ID를 전달합니다. 유형: `OnStartGameSession func (model.GameSession )` 필수 항목 여부: 예
`OnUpdateGameSession`	업데이트된 게임 세션 정보를 서버 프로세스에 전달하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 업데이트된 매치메이커 데이터를 제공하기 위해 매치 백필 요청을 처리한 후 이 함수를 GameLift 호출합니다. 유형: `OnUpdateGameSession func (model.UpdateGameSession)` 필수 항목 여부: 아니요
`Port`	서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다. 유형: `int` 필수 항목 여부: 예

UpdateGameSession

게임 세션 객체의 업데이트로, 여기에는 게임 세션이 업데이트된 이유와 게임 세션의 플레이어 세션을 채우기 위해 backfill을 사용하는 경우 관련된 채우기 티켓 ID가 포함됩니다.

속성	설명
GameSession	Amazon GameLift API에서 정의한 GameSession객체입니다. `GameSession` 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 유형: `GameSession GameSession()` 필수 항목 여부: 예

속성	설명
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: `UpdateReason UpdateReason()` 필수 항목 여부: 예
BackfillTicketId	게임 세션 업데이트를 시도하는 채우기 티켓의 ID입니다. 유형: `String` 필수 항목 여부: 아니요

GameSession

게임 세션의 세부 정보입니다.

속성	설명
GameSessionId	게임 세션에 대한 고유 식별자입니다. 게임 세션 Amazon 리소스 이름(ARN)은 다음과 같은 형식을 갖습니다. `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 유형: `String` 필수 항목 여부: 아니요
명칭	게임 세션을 설명하는 레이블입니다. 유형: `String` 필수 항목 여부: 아니요
FleetId	게임 세션이 실행 중인 플릿의 고유 식별자입니다. 유형: `String` 필수 항목 여부: 아니요

속성	설명
MaximumPlayerSessionCount	게임 세션에 연결된 최대 플레이어 수입니다. 유형: `Integer` 필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `Integer` 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `String` 필수 항목 여부: 아니요
GameSessionData	단일 문자열 값으로 포맷된 사용자 지정 게임 세션 속성의 집합입니다. 유형: `String` 필수 항목 여부: 아니요
MatchmakerData	게임 세션을 생성하는 데 사용된 매치메이킹 프로세스에 대한 정보(JSON 구문 사용, 문자열 형식). 사용된 매치메이킹 구성 외에도 플레이어 속성 및 팀 배정을 포함하여 경기에 배정된 모든 플레이어에 대한 데이터가 포함됩니다. 유형: `String` 필수 항목 여부: 아니요
GameProperties	키:값 페어로 포맷된 게임 세션에 대한 사용자 정의 속성 집합입니다. 이러한 속성은 새 게임 세션을 시작하라는 요청과 함께 전달됩니다. 유형: `map[string] string` 필수 항목 여부: 아니요

속성	설명
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. • TLS 지원 플릿: `<unique identifier>.<region identifier>.amazongamelift.com` • TLS 미지원 플릿: `ec2-<unique identifier>.compute.amazonaws.com` TLS 지원 플릿에서 실행되는 게임 세션에 연결할 때는 IP 주소가 아닌 DNS 이름을 사용해야 합니다. 유형: `String` 필수 항목 여부: 아니요

ServerParameters

Amazon GameLift Anywhere 서버와 Amazon GameLift 서비스 간의 연결을 유지하는 데 사용되는 정보. 이 정보는 InitSDK()에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용하십시오.

속성	설명
`WebSocketURL`	`GameLiftServerSdkEndpoint` Amazon GameLift Anywhere 컴퓨팅 리소스를 `RegisterCompute` 요청하면 Amazon이 GameLift 반품합니다. 유형: `string` 필수 항목 여부: 예
`ProcessID`	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: `string` 필수 항목 여부: 예
`HostID`	새 서버 프로세스를 호스팅하는 컴퓨팅 리소스의 고유 식별자입니다.

속성	설명
	HostID는 컴퓨터를 등록할 때 사용되는 ComputeName 입니다. 자세한 내용은 을 참조하십시오 RegisterCompute. 유형: string 필수 항목 여부: 예
FleetID	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은 을 참조하십시오 RegisterCompute. 유형: string 필수 항목 여부: 예
AuthToken	Amazon에서 생성한 인증 토큰으로, Amazon에서 GameLift 서버를 GameLift Amazon에 인증합니다. 자세한 내용은 을 참조하십시오 GetComputeAuthToken. 유형: string 필수 항목 여부: 예

StartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 StartMatchBackfill() 통화를 통해 이 정보를 GameLift Amazon에 전달합니다.

속성	설명
GameSessionArn	고유한 게임 세션 식별자입니다. API 작업 GetGameSessionId 는 ARN 형식의 식별자를 반환합니다. 유형: String 필수 항목 여부: 예

속성	설명
MatchmakingConfigurationArn	매치메이커가 이 요청에 사용할 (ARN 형식의) 고유 식별자입니다. 원본 게임 세션에 대한 매치메이커 ARN은 매치메이커 데이터 속성의 게임 세션 객체에 있습니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요. 유형: `String` 필수 항목 여부: 예
Players	현재 게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 유형: `[]model.Player` 필수 항목 여부: 예
TicketId	매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 값을 제공하지 않으면 Amazon에서 값을 GameLift 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다. 유형: `String` 필수 항목 여부: 아니요

Player

매치메이킹 중인 플레이어를 나타내는 객체입니다. 매치메이킹 요청을 시작할 때 플레이어는 플레이어 ID, 속성, 지연 시간 데이터를 가지고 있을 수 있습니다. Amazon은 경기가 이루어진 후 팀 정보를 GameLift 추가합니다.

속성	설명
LatencyInMS	플레이어가 특정 위치에 연결되었을 때 발생하는 지연 시간을 나타내는 값 집합으로, 밀리초 단위로 표시됩니다.

속성	설명
	이 속성을 사용하는 경우 플레이어는 나열된 위치에서만 매칭됩니다. 매치메이커에 플레이어 지연 시간을 평가하는 규칙이 있는 경우 플레이어는 지연 시간을 보고해야 매칭됩니다. 유형: `map[string] int` 필수 항목 여부: 아니요
PlayerAttributes	매치메이킹에 사용할 플레이어 정보가 포함된 키:값 페어의 모음입니다. 플레이어 속성 키는 매치메이킹 규칙 PlayerAttributes 세트에 사용된 것과 일치해야 합니다. 플레이어 속성에 대한 자세한 내용은 을 참조하십시오. AttributeValue 유형: `map[string] AttributeValue` 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: `String` 필수 항목 여부: 아니요
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: `String` 필수 항목 여부: 아니요

DescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon에 대한 DescribePlayerSessions() 호출을 통해 이 정보를 제공합니다 GameLift.

속성	설명
GameSessionID	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string> 입니다. GameSessionID 는 사용자 지정 ID 문자열 또는 생성된 문자열입니다. 유형: String 필수 항목 여부: 아니요
PlayerSessionID	플레이어 세션의 고유 식별자입니다. 이 파라미터를 사용하여 단일 특정 플레이어 세션을 요청합니다. 유형: String 필수 항목 여부: 아니요
PlayerID	플레이어의 고유 식별자입니다. 이 파라미터를 사용하여 특정 플레이어에 대한 모든 플레이어 세션을 요청합니다. 플레이어 ID 생성 섹션을 참조하십시오. 유형: String 필수 항목 여부: 아니요
PlayerSessionStatusFilter	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다. • RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다. • ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 연결되어 있습니다. • COMPLETED - 플레이어 연결이 끊어졌습니다. • TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다. 유형: String

속성	설명
	필수 항목 여부: 아니요
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: String 필수 항목 여부: 아니요
Limit	반환할 최대 결과 수입니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: int 필수 항목 여부: 아니요

StopMatchBackfillRequest

매치메이킹 채우기 요청을 취소하는 데 사용되는 정보입니다. 게임 서버는 StopMatchBackfill() 통화를 통해 이 정보를 Amazon GameLift 서비스에 전달합니다.

속성	설명
GameSessionArn	취소 중인 요청의 고유한 게임 세션 식별자입니다. 유형: string 필수 항목 여부: 아니요
MatchmakingConfigurationArn	이 요청을 보낸 매치메이커의 고유 식별자입니다. 유형: string 필수 항목 여부: 아니요
TicketId	취소할 채우기 요청 티켓의 고유 식별자입니다. 유형: string

속성	설명
	필수 항목 여부: 아니요

GetFleetRoleCredentialsRequest

AWS 리소스에 대한 제한된 액세스를 게임 서버까지 확장하는 역할 자격 증명입니다. 자세한 내용은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요.

속성	설명
RoleArn	AWS 리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 ARN입니다. 유형: `string` 필수 항목 여부: 예
RoleSessionName	역할 자격 증명의 사용을 설명하는 세션 이름입니다. 유형: `string` 필수 항목 여부: 예

Unreal Engine용 Amazon GameLift Server SDK 참조

이 Amazon GameLift Server API 참조는 Amazon GameLift와 함께 사용할 Unreal Engine 게임 프로젝트를 준비하는 데 유용합니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

이 API는 `GameLiftServerSDK.h` 및 `GameLiftServerSDKModels.h`에 정의되어 있습니다.

Unreal Engine 플러그인을 설치하려면 코드 샘플 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

주제

- Amazon GameLift Unreal Engine Server SDK 5.x 참조
- Amazon GameLift Unreal Engine Server SDK 3.x 참조

Amazon GameLift Unreal Engine Server SDK 5.x 참조

이 Amazon GameLift Unreal Engine Server SDK 5.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하고, Unreal SDK Server 플러그인 사용에 대한 자세한 내용은 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

주제

- 아마존 GameLift 서버 SDK (언리얼) 5.x 레퍼런스: 액션
- Amazon GameLift 서버 SDK (언리얼) 참조: 데이터 유형

아마존 GameLift 서버 SDK (언리얼) 5.x 레퍼런스: 액션

이 Amazon GameLift Unreal 서버 SDK 레퍼런스를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하고, Unreal SDK Server 플러그인 사용에 대한 자세한 내용은 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

작업

- GetSdkVersion()
- InitSDK()
- InitSDK()
- ProcessReady()
- ProcessEnding()
- ActivateGameSession()
- UpdatePlayerSessionCreationPolicy()
- GetGameSessionId()
- GetTerminationTime()
- AcceptPlayerSession()
- RemovePlayerSession()
- DescribePlayerSessions()
- StartMatchBackfill()
- StopMatchBackfill()

- GetComputeCertificate()
- GetFleetRoleCredentials()

> **Note**
>
> 이 주제에서는 언리얼 엔진용으로 빌드할 때 사용할 수 있는 Amazon GameLift C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DBUILD_FOR_UNREAL=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
FGameLiftStringOutcome GetSdkVersion();
```

반환 값

성공하면 현재 SDK 버전을 the section called “F GameLiftStringOutcome” 객체로 반환합니다. 반환된 객체는 버전 번호(예: `5.0.0`)를 포함합니다. 실패하면 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
 Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

관리형 EC2 플릿을 위한 Amazon GameLift SDK를 초기화합니다. GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 메서드는 호스트 환경에서 서버 파라미터를 읽어 서버와 Amazon GameLift 서비스 간의 통신을 설정합니다.

명령문

```
FGameLiftGenericOutcome InitSDK()
```

반환 값

성공하는 경우, 서버 프로세스가 ProcessReady()를 호출할 준비가 되었음을 나타내는 `InitSdkOutcome` 객체를 반환합니다.

예

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
 further communication.
FGameLiftGenericOutcome initSdkOutcome = gameLiftSdkModule->InitSDK();
```

InitSDK()

플릿의 Amazon GameLift SDK를 초기화합니다. Anywhere GameLift Amazon과 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출하십시오. 이 방법을 사용하려면 서버와 Amazon GameLift 서비스 간의 통신을 설정하기 위한 명시적인 서버 파라미터가 필요합니다.

명령문

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

파라미터

F ServerParameters

Amazon GameLift Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 생성하십시오.

- 게임 서버에 연결하는 데 WebSocket 사용되는 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Anywhere 컴퓨팅을 포함하는 아마존 GameLift 플릿의 ID입니다.
- Amazon GameLift 작업에서 생성된 인증 토큰입니다.

반환 값

성공하는 경우, 서버 프로세스가 ProcessReady()를 호출할 준비가 되었음을 나타내는 `InitSdkOutcome` 객체를 반환합니다.

> **Note**
>
> Anywhere 플릿에 배포된 게임 빌드에 대한 `InitSDK()`로의 호출이 실패하는 경우 빌드 리소스를 생성할 때 사용된 `ServerSdkVersion` 파라미터를 확인합니다. 명시적으로 이 값을 사용 중인 Server SDK 버전으로 설정해야 합니다. 이 파라미터의 기본값은 4.x이며 호환되지 않습니다. 이 문제를 해결하려면 새 빌드를 생성하여 새 플릿에 배포해야 합니다.

예

```
//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()

서버 프로세스가 게임 세션을 호스팅할 준비가 GameLift 되었음을 Amazon에 알립니다. InitSDK() 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

명령문

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

파라미터

processParameters

다음 서버 프로세스 관련 정보를 전달하는 F. ProcessParameters 객체입니다.

- Amazon GameLift 서비스가 서버 프로세스와 통신하기 위해 호출하는 게임 서버 코드에 구현된 콜백 메서드의 이름입니다.

- 서버 프로세스가 수신하는 포트 번호입니다.
- GameLift Amazon에서 캡처하여 저장하려는 게임 세션별 파일의 경로입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 ProcessReady() 호출 및 위임 함수 구현을 모두 보여줍니다.

```
//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()

서버 프로세스가 종료되고 GameLift 있음을 Amazon에 알립니다. 다른 모든 정리 작업 (활성 게임 세션 종료 포함) 이 끝난 후 프로세스를 종료하기 전에 이 메서드를 호출합니다. ProcessEnding( )의 결과에 따라 프로세스가 성공(0) 또는 오류(-1)로 종료되고 플릿 이벤트가 생성됩니다. 오류가 발생하여 프로세스가 종료되는 경우 생성되는 플릿 이벤트는 SERVER_PROCESS_TERMINATED_UNHEALTHY입니다.

명령문

```
FGameLiftGenericOutcome ProcessEnding()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
//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()

서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 GameLift 되었음을 Amazon에 알립니다. 이 작업은 모든 게임 세션 초기화 후 `onStartGameSession()` 콜백 함수의 일부로 호출되어야 합니다.

명령문

```
FGameLiftGenericOutcome ActivateGameSession()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예에서는 `onStartGameSession()` 위임 함수의 일부로 `ActivateGameSession()`이 호출되는 것을 보여줍니다.

```
//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()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다.

명령문

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
 policy)
```

파라미터

playerCreationSession정책

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다.

유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
FGameLiftGenericOutcome outcome = gameLiftSdkModule-
>UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

활성 서버 프로세스가 호스팅된 게임 세션의 ID를 가져옵니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 the section called “F GameLiftError”를 반환합니다.

명령문

```
FGameLiftStringOutcome GetGameSessionId()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 the section called “F GameLiftStringOutcome” 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

게임 세션으로 활성화되지 않은 유휴 프로세스의 경우 호출은 `Success=True` 및 `GameSessionId=""`를 반환합니다.

예

```
//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()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 GameLift Amazon으로부터 `onProcessTerminate()` 콜백을 받은 후 조치를 취합니다. Amazon에서 GameLift 전화를 거는 `onProcessTerminate()` 이유는 다음과 같습니다.

- 서버 프로세스가 상태가 좋지 않다고 보고되었거나 GameLift Amazon에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- 스팟 인스턴스 중단으로 인해 인스턴스가 종료되는 경우

명령문

```
AwsDateTimeOutcome GetTerminationTime()
```

반환 값

성공하면 종료 시간을 `AwsDateTimeOutcome` 객체로 반환합니다. 값은 종료 시간이며, `0001 00:00:00` 이후 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값 `2020-09-13 12:26:40`

-000Z는 637355968000000000 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

프로세스가 ProcessParameters.OnProcessTerminate() 콜백을 받지 못한 경우 오류 메시지가 반환됩니다. 서버 프로세스 종료에 대한 자세한 내용은 서버 프로세스 종료 알림에 응답 섹션을 참조하세요.

예

```
AwsDateTimeOutcome TermTimeOutcome = gameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되었으며 검증이 GameLift 필요함을 Amazon에 알립니다. Amazon은 플레이어 세션 ID가 유효한지 GameLift 확인합니다. 플레이어 세션이 검증된 후 Amazon은 플레이어 슬롯의 상태를 예약됨에서 ACTIVE로 GameLift 변경합니다.

명령문

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

이 예제는 유효하지 않은 플레이어 세션 ID의 검증 및 거부를 포함하는 연결 요청을 처리합니다.

```
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()

플레이어가 서버 프로세스와의 연결이 GameLift 끊겼음을 Amazon에 알립니다. 이에 대해 Amazon은 플레이어 슬롯을 사용 가능한 것으로 GameLift 변경합니다.

명령문

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

파라미터

playerSessionId

새 플레이어 세션이 GameLift 생성될 때 Amazon에서 발급하는 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
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()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 메서드를 사용하여 다음에 대한 정보를 얻을 수 있습니다.

- 단일 플레이어 세션
- 게임 세션의 모든 플레이어 세션
- 단일 플레이어 ID와 연결된 모든 플레이어 세션

명령문

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
 FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

파라미터

F GameLiftDescribePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 the section called “F GameLiftDescribePlayerSessionsRequest” 객체입니다.

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 the section called “F GameLiftDescribePlayerSessionsOutcome” 객체를 반환합니다.

예

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. 제한 값을 NextToken생략하고 10으로 설정하면 Amazon은 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 GameLift 반환합니다.

```
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()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. 자세한 내용은 FlexMatch 백필 기능을 참조하십시오.

이 작업은 비동기식입니다. 신규 플레이어가 매칭되면 Amazon은 콜백 함수를 사용하여 업데이트된 매치메이커 데이터를 GameLift 제공합니다. `OnUpdateGameSession()`

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

명령문

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
 &startBackfillRequest);
```

파라미터

F StartMatchBackfillRequest

다음 StartMatchBackfillRequest 정보를 전달하는 객체:

- 채우기 요청에 할당할 티켓 ID. 이 정보는 선택 사항입니다. ID를 제공하지 GameLift 않으면 Amazon에서 ID를 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에 있습니다.
- 채울 게임 세션의 ID입니다.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터입니다.

반환 값

매치 채우기 티켓 ID와 함께 `StartMatchBackfillOutcome` 객체나 오류 메시지를 포함한 결함을 반환합니다.

예

```
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()

활성 매치 채우기 요청을 취소합니다. 자세한 내용은 FlexMatch백필 기능을 참조하십시오.

명령문

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

파라미터

F. StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체:

- 채우기 요청에 할당된 티켓 ID
- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

```
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()

Amazon GameLift Anywhere 컴퓨팅 리소스와 Amazon 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. GameLift Amazon GameLift Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은, 을 참조하십시오 RegisterCompute.

명령문

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

반환 값

다음을 포함하는 `GetComputeCertificateResponse` 객체를 반환합니다.

- CertificatePath: 컴퓨팅 리소스의 TLS 인증서 경로.
- HostName: 컴퓨팅 리소스의 호스트 이름.

예

```
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()

GameLift Amazon이 다른 사람과 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. AWS 서비스 자세한 설명은 플릿에서 다른 AWS 리소스와 통신 섹션을 참조하세요.

구문

```
FGameLiftGetFleetRoleCredentialsOutcome
 FGameLiftServerSDKModule::GetFleetRoleCredentials(const
 FGameLiftGetFleetRoleCredentialsRequest &request)
```

파라미터

F GameLiftGetFleetRoleCredentialsRequest

반환 값

the section called “F GameLiftGetFleetRoleCredentialsOutcome” 객체를 반환합니다.

예

```
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 서버 SDK (언리얼) 참조: 데이터 유형

이 Amazon GameLift Unreal 서버 SDK 레퍼런스를 사용하면 Amazon에서 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. GameLift 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하고, Unreal SDK Server 플러그인 사용에 대한 자세한 내용은 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

데이터 타입

- F. ProcessParameters
- UpdateGameSession
- GameSession
- F ServerParameters
- F StartMatchBackfillRequest
- FPlayer
- F GameLiftDescribePlayerSessionsRequest
- F. StopMatchBackfillRequest
- F. AttributeValue
- F GameLiftGetFleetRoleCredentialsRequest

- F GameLiftLongOutcome
- F GameLiftStringOutcome
- F GameLiftDescribePlayerSessionsOutcome
- F GameLiftDescribePlayerSessionsResult
- F GenericOutcome
- F GameLiftPlayerSession
- F GameLiftGetComputeCertificateOutcome
- F GameLiftGetComputeCertificateResult
- F GameLiftGetFleetRoleCredentialsOutcome
- F GetFleetRoleCredentialsResult
- F GameLiftError
- Enums

> **Note**
>
> 이 주제에서는 언리얼 엔진용으로 빌드할 때 사용할 수 있는 Amazon GameLift C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DBUILD_FOR_UNREAL=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

F. ProcessParameters

이 데이터 유형에는 a로 GameLift Amazon으로 전송되는 매개변수 세트가 포함되어 ProcessReady() 있습니다.

속성	설명
LogParameters	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon은 나중에 액세스할 수 있도록 파일을 GameLift 복사하고 저장합니다. 유형: `TArray<FString>` 필수 항목 여부: 아니요

OnHealthCheck	Amazon이 서버 프로세스에 상태 보고서를 요청하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 60초마다 이 함수를 GameLift 호출하고 응답을 받을 때까지 60초 동안 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon은 서버 프로세스를 정상이 아닌 것으로 GameLift 기록합니다. 이 속성은 다음과 같이 정의된 대리자 함수입니다. `DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)` 유형: `FOnHealthCheck` 필수 항목 여부: 아니요
OnProcessTerminate	Amazon이 서버 프로세스를 강제로 종료하기 위해 GameLift 호출하는 콜백 함수입니다. Amazon은 이 함수를 호출한 후 서버 프로세스가 종료될 때까지 5분간 GameLift 기다린 후 ProcessEnding() 호출에 응답한 후 서버 프로세스를 종료합니다. 유형: `FSimpleDelegate` 필수 항목 여부: 예

OnStartGameSession	새 게임 세션을 활성화하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 클라이언트 요청에 대한 응답으로 이 함수를 GameLift CreateGameSession호출합니다. 콜백 함수는 Amazon GameLift API 참조에 정의된 대로 GameSession객체를 전달합니다. 이 속성은 다음과 같이 정의된 대리자 함수입니다. `DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);` 유형: `FOnStartGameSession` 필수 항목 여부: 예
OnUpdateGameSession	업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon이 GameLift 호출하는 콜백 함수입니다. Amazon은 업데이트된 매치메이커 데이터를 제공하기 위해 매치 백필 요청이 처리되면 이 함수를 GameLift 호출합니다. GameSession객체, 상태 업데이트 (`updateReason`), 매치 백필 티켓 ID를 전달합니다. 이 속성은 다음과 같이 정의된 대리자 함수입니다. `DECLARE_DELEGATE_OneParam(FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession);` 유형: `FOnUpdateGameSession` 필수 항목 여부: 아니요

Port	서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다. 유형: `int` 필수 항목 여부: 예

UpdateGameSession

이 데이터 유형은 게임 세션 객체로 업데이트되며, 여기에는 게임 세션이 업데이트된 이유와 게임 세션의 플레이어 세션을 채우기 위해 backfill을 사용하는 경우 관련된 백필 티켓 ID가 포함됩니다.

속성	설명
GameSession	Amazon GameLift API에서 정의한 GameSession객체입니다. `GameSession` 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 유형: `Aws::GameLift::Server::GameSession` 필수 항목 여부: 아니요
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: `enum class UpdateReason` • MATCHMAKING_DATA_UPDATED • BACKFILL_FAILED • BACKFILL_TIMED_OUT • BACKFILL_CANCELLED

속성	설명
	필수 항목 여부: 아니요
BackfillTicketId	게임 세션 업데이트를 시도하는 채우기 티켓의 ID입니다. 유형: `char[]` 필수 항목 여부: 아니요

GameSession

이 데이터 유형은 게임 세션의 세부 정보를 제공합니다.

속성	설명
GameSessionId	게임 세션에 대한 고유 식별자입니다. 게임 세션 ARN의 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token>` 유형: `char[]` 필수 항목 여부: 아니요
명칭	게임 세션을 설명하는 레이블입니다. 유형: `char[]` 필수 항목 여부: 아니요
FleetId	게임 세션이 실행 중인 플릿의 고유 식별자입니다. 유형: `char[]` 필수 항목 여부: 아니요
MaximumPlayerSessionCount	게임 세션에 연결된 최대 플레이어 수입니다.

속성	설명
	유형: `int` 필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `int` 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: `char[]` 필수 항목 여부: 아니요
GameSessionData	단일 문자열 값으로 포맷된 사용자 지정 게임 세션 속성의 집합입니다. 유형: `char[]` 필수 항목 여부: 아니요
MatchmakerData	게임 세션을 생성하는 데 사용된 매치메이킹 프로세스에 대한 정보(JSON 구문 사용, 문자열 형식). 사용된 매치메이킹 구성 외에도 플레이어 속성 및 팀 배정을 포함하여 경기에 배정된 모든 플레이어에 대한 데이터가 포함됩니다. 유형: `char[]` 필수 항목 여부: 아니요

속성	설명
GameProperties	키:값 페어로 포맷된 게임 세션에 대한 사용자 정의 속성 집합입니다. 이러한 속성은 새 게임 세션을 시작하라는 요청과 함께 전달됩니다. 유형: `GameProperty[]` 필수 항목 여부: 아니요
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. • TLS 지원 플릿: `<unique identifier>.<region identifier>.amazongamelift.com` • TLS 미지원 플릿: `ec2-<unique identifier>.compute.amazonaws.com` TLS 지원 플릿에서 실행되는 게임 세션에 연결할 때는 IP 주소가 아닌 DNS 이름을 사용해야 합니다. 유형: `char[]` 필수 항목 여부: 아니요

F ServerParameters

Amazon GameLift Anywhere 서버와 Amazon GameLift 서비스 간의 연결을 유지하는 데 사용되는 정보. 이 정보는 InitSDK()에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용하십시오.

속성	설명
webSocketUrl	`GameLiftServerSdkEndpoint` Amazon GameLift Anywhere 컴퓨팅 리소스를

속성	설명
	`RegisterCompute` 요청하면 Amazon이 GameLift 반품합니다. 유형: `char[]` 필수 항목 여부: 예
processId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: `char[]` 필수 항목 여부: 예
hostId	`HostID`는 컴퓨터를 등록할 때 사용되는 `ComputeName` 입니다. 자세한 내용은, 을 참조하십시오 RegisterCompute. 유형: `char[]` 필수 항목 여부: 예
fleetId	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은, 을 참조하십시오 RegisterCompute. 유형: `char[]` 필수 항목 여부: 예
authToken	Amazon에서 생성한 인증 토큰으로, Amazon에서 GameLift 서버를 GameLift Amazon에 인증합니다. 자세한 내용은, GetComputeAuthToken을 참조하십시오. 유형: `char[]` 필수 항목 여부: 예

F StartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 StartMatchBackfill() 통화를 통해 이 정보를 GameLift Amazon에 전달합니다.

속성	설명
GameSessionArn	고유한 게임 세션 식별자입니다. API 작업 `GetGameSessionId` 는 ARN 형식의 식별자를 반환합니다. 유형: `char[]` 필수 항목 여부: 예
MatchmakingConfigurationArn	매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션에 대한 매치메이커 ARN은 매치메이커 데이터 속성의 게임 세션 객체에 있습니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요. 유형: `char[]` 필수 항목 여부: 예
Players	게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 유형: `TArray<FPlayer>` 필수 항목 여부: 예
TicketId	매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 값을 제공하지 않으면 Amazon에서 값을 GameLift 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다.

속성	설명
	유형: `char[]` 필수 항목 여부: 아니요

FPlayer

이 데이터 유형은 매치메이킹 중인 플레이어를 나타냅니다. 매치메이킹 요청을 시작할 때 플레이어는 플레이어 ID, 속성, 지연 시간 데이터를 가지고 있을 수 있습니다. Amazon은 경기가 이루어진 후 팀 정보를 GameLift 추가합니다.

속성	설명
LatencyInMS	플레이어가 특정 위치에 연결되었을 때 발생하는 지연 시간을 나타내는 값 집합으로, 밀리초 단위로 표시됩니다. 이 속성을 사용하는 경우 플레이어는 나열된 위치에서만 매칭됩니다. 매치메이커에 플레이어 지연 시간을 평가하는 규칙이 있는 경우 플레이어는 지연 시간을 보고해야 매칭됩니다. 유형: `TMap>FString, int32<` 필수 항목 여부: 아니요
PlayerAttributes	매치메이킹에 사용할 플레이어 정보가 포함된 키:값 페어의 모음입니다. 플레이어 속성 키는 매치메이킹 규칙 PlayerAttributes 세트에 사용된 것과 일치해야 합니다. 플레이어 속성에 대한 자세한 내용은 을 참조하십시오. AttributeValue 유형: `TMap>FString, FAttribut eValue<` 필수 항목 여부: 아니요

속성	설명
PlayerId	플레이어의 고유 식별자입니다. 유형: `std::string` 필수 항목 여부: 아니요
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: `FString` 필수 항목 여부: 아니요

F GameLiftDescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon에 대한 DescribePlayerSessions() 호출을 통해 이 정보를 제공합니다 GameLift.

속성	설명
GameSessionId	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 `FString` 입니다. `GameSessionID` 는 다음입니다. 사용자 지정 ID 문자열 또는 유형: `std::string` 필수 항목 여부: 아니요
PlayerSessionId	플레이어 세션의 고유 식별자입니다. 이 파라미터를 사용하여 단일 특정 플레이어 세션을 요청합니다. 유형: `FString`

속성	설명
	필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 이 파라미터를 사용하여 특정 플레이어에 대한 모든 플레이어 세션을 요청합니다. 플레이어 ID 생성 섹션을 참조하십시오. 유형: `FString` 필수 항목 여부: 아니요
PlayerSessionStatusFilter	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다. • RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다. • ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 연결되어 있습니다. • COMPLETED - 플레이어 연결이 끊어졌습니다. • TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다. 유형: `FString` 필수 항목 여부: 아니요

속성	설명
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `FString` 필수 항목 여부: 아니요
Limit	반환할 최대 결과 수입니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `int` 필수 항목 여부: 아니요

F. StopMatchBackfillRequest

매치메이킹 채우기 요청을 취소하는 데 사용되는 정보입니다. 게임 서버는 StopMatchBackfill() 통화를 통해 이 정보를 Amazon GameLift 서비스에 전달합니다.

속성	설명
GameSessionArn	취소 중인 요청의 고유한 게임 세션 식별자입니다. 유형: `FString` 필수 항목 여부: 예
MatchmakingConfigurationArn	이 요청을 보낸 매치메이커의 고유 식별자입니다. 유형: `FString` 필수 항목 여부: 예
TicketId	취소할 채우기 요청 티켓의 고유 식별자입니다.

속성	설명
	유형: `FString` 필수 항목 여부: 예

F. AttributeValue

FPlayer 속성 키-값 페어에 이러한 값을 사용합니다. 이 객체를 사용하면 문자열, 숫자, 문자열 배열 또는 데이터 맵과 같은 유효한 데이터 유형을 사용하여 속성 값을 지정할 수 있습니다. 각 `AttributeValue` 객체는 사용 가능한 속성 중 하나만 사용할 수 있습니다.

속성	설명
attrType	속성 값의 유형을 지정합니다. 형식: `FAttributeType` 열거형 값. 필수 항목 여부: 아니요
S	문자열 속성 값을 나타냅니다. 유형: `FString` 필수 항목 여부: 아니요
N	숫자 속성 값을 나타냅니다. 유형: `double` 필수 항목 여부: 아니요
SL	문자열 속성 값의 배열을 나타냅니다. 유형: `TArray<FString>` 필수 항목 여부: 아니요
SDM	문자열 키와 이중 값의 사전을 나타냅니다. 유형: `TMap<FString, double>`

속성	설명
	필수 항목 여부: 아니요

F GameLiftGetFleetRoleCredentialsRequest

이 데이터 유형은 AWS 리소스에 대한 제한된 액세스를 게임 서버까지 확장하는 역할 자격 증명을 제공합니다. 자세한 내용은 Amazon에 대한 IAM 서비스 역할 설정 GameLift 섹션을 참조하세요.

속성	설명
RoleArn	AWS 리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `FString` 필수 항목 여부: 아니요
RoleSessionName	역할 자격 증명의 사용을 설명하는 세션 이름입니다. 유형: `FString` 필수 항목 여부: 아니요

F GameLiftLongOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `long` 필수 항목 여부: 아니요

속성	설명
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `long&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GameLiftStringOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: `FString` 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `FString&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다.

속성	설명
	유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GameLiftDescribePlayerSessionsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “F GameLiftDescribePlayerSessionsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `FGameLiftDescribePlayerSessionsResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다.

속성	설명
	유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GameLiftDescribePlayerSessionsResult

속성	설명
PlayerSessions	유형: `TArray<FGameLiftPlayerSession>` 필수 항목 여부: 예
NextToken	결과의 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: `FString` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GenericOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GameLiftPlayerSession

속성	설명
CreationTime	유형: `long` 필수 항목 여부: 예
FleetId	유형: `FString` 필수 항목 여부: 예
GameSessionId	유형: `FString` 필수 항목 여부: 예
IpAddress	유형: `FString` 필수 항목 여부: 예
PlayerData	유형: `FString` 필수 항목 여부: 예
PlayerId	유형: `FString`

속성	설명
	필수 항목 여부: 예
PlayerSessionId	유형: `FString` 필수 항목 여부: 예
Port	유형: `int` 필수 항목 여부: 예
Status	유형: `PlayerSessionStatus` 열거형. 필수 항목 여부: 예
TerminationTime	유형: `long` 필수 항목 여부: 예
DnsName	유형: `FString` 필수 항목 여부: 예

F GameLiftGetComputeCertificateOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “F GameLiftGetComputeCertificateResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다.

속성	설명
	유형: `FGameLiftGetComputeCertificateResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GameLiftGetComputeCertificateResult

컴퓨팅의 TLS 인증서 경로 및 컴퓨팅 호스트 이름입니다.

속성	설명
CertificatePath	유형: `FString` 필수 항목 여부: 예
ComputeName	유형: `FString` 필수 항목 여부: 예

F GameLiftGetFleetRoleCredentialsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다.

속성	설명
	유형: the section called “F GetFleetRoleCredentialsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: `FGameLiftGetFleetRoleCredentialsResult&&` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “F GameLiftError” 필수 항목 여부: 아니요

F GetFleetRoleCredentialsResult

속성	설명
AccessKeyId	인증하고 AWS 리소스에 액세스를 제공하기 위한 액세스 키 ID입니다. 유형: `FString` 필수 항목 여부: 아니요
AssumedRoleId	서비스 역할이 속한 사용자의 ID입니다. 유형: `FString`

속성	설명
	필수 항목 여부: 아니요
AssumedRoleUserArn	사용자가 맡을 서비스 역할의 Amazon 리소스 이름(ARN)입니다. 유형: `FString` 필수 항목 여부: 아니요
Expiration	세션 자격 증명이 만료될 때까지 남은 시간입니다. 유형: `FDateTime` 필수 항목 여부: 아니요
SecretAccessKey	인증에 대한 비밀 액세스 키 ID를 지정합니다. 유형: `FString` 필수 항목 여부: 아니요
SessionToken	AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하는 토큰입니다. 유형: `FString` 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: `bool` 필수 항목 여부: 예
Error	작업이 실패한 경우 발생되는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

F GameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: `GameLiftErrorType` 열거형. 필수 항목 여부: 아니요
ErrorName	오류 유형 이름입니다. 유형: `std::string` 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: `std::string` 필수 항목 여부: 아니요

Enums

Amazon GameLift 서버 SDK (언리얼) 에 정의된 열거형은 다음과 같이 정의됩니다.

F. AttributeType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

오류 유형을 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- SERVICE_CALL_FAILED - AWS 서비스 호출이 실패했습니다.
- LOCAL_CONNECTION_FAILED — 아마존에 대한 로컬 연결에 실패했습니다. GameLift
- NETWORK_NOT_INITIALIZED - 네트워크가 초기화되지 않았습니다.

- GAMESESSION_ID_NOT_SET - 게임 세션 ID가 설정되지 않았습니다.
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION
- ALREADY_INITIALEZED — GameLift 아마존 서버 또는 클라이언트가 이미 Initialize () 를 사용하여 초기화되었습니다.
- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALEZED — 아마존 클라이언트가 초기화되지 않았습니다. GameLift
- GAMELIFT_SERVER_NOT_INITIALIZED — 아마존 서버가 초기화되지 않았습니다. GameLift
- GAME_SESSION_ENDED_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY — 아마존 GameLift 서버 게임 세션이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 게임 세션이 준비되었다고 보고할 수 없습니다.
- INITIALIZATION_MISMATCH - 클라이언트 메서드는 Server::Initialize() 이후에 호출되었으며 그 반대의 경우도 마찬가지입니다.
- NOT_INITIALIZED — 아마존 GameLift 서버 또는 클라이언트가 Initialize () 를 사용하여 초기화되지 않았습니다.
- NO_TARGET_ALIASID_SET - 대상 aliasId가 설정되지 않았습니다.
- NO_TARGET_FLEET_SET - 대상 플릿이 설정되지 않았습니다.
- PROCESS_ENDING_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 프로세스가 종료되었음을 보고할 수 없습니다.
- PROCESS_NOT_ACTIVE — 서버 프로세스가 아직 활성화되지 않았고 a에 바인딩되지 않았으므로 수락하거나 처리할 수 없습니다. GameSession PlayerSessions
- PROCESS_NOT_READY - 서버 프로세스를 아직 활성화할 준비가 되지 않았습니다.
- PROCESS_READY_FAILED — GameLift Amazon Server SDK가 서비스에 연락하여 프로세스가 준비되었다고 보고할 수 없습니다.
- SDK_VERSION_DETECTION_FAILED - SDK 버전 감지에 실패했습니다.
- STX_CALL_FAILED - XStx 서버 백엔드 구성 요소에 대한 호출이 실패했습니다.
- STX_INITIALIZATION_FAILED - XStx 서버 백엔드 구성 요소를 초기화하지 못했습니다.
- UNEXPECTED_PLAYER_SESSION - 서버에서 등록되지 않은 플레이어 세션을 발견했습니다.

- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE — 서비스에 메시지를 보내는데 다시 실패했습니다. GameLift WebSocket
- WEBSOCKET_SEND_MESSAGE_FAILURE — 서비스에 메시지를 GameLift 보내지 못했습니다. WebSocket
- MATCH_BACKFILL_REQUEST_VALIDATION - 요청 검증에 실패했습니다.
- PLAYER_SESSION_REQUEST_VALIDATION - 요청 검증에 실패했습니다.

E. PlayerSessionCreationPolicy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.
- NOT_SET - 게임 세션이 새 플레이어 세션을 수락하거나 거부하도록 설정되지 않았습니다.

E PlayerSessionStatus

- ACTIVE
- COMPLETED
- NOT_SET
- RESERVED
- TIMEDOUT

Amazon GameLift Unreal Engine Server SDK 3.x 참조

이 Amazon GameLift Unreal Engine Server SDK 3.x 참조를 사용하면 Amazon GameLift와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

주제

- Unreal Engine용 Amazon GameLift Server SDK 참조: 작업
- Unreal Engine용 Amazon GameLift Server SDK 참조: 데이터 유형

Unreal Engine용 Amazon GameLift Server SDK 참조: 작업

이 Amazon GameLift Server API 참조는 Amazon GameLift와 함께 사용할 Unreal Engine 게임 프로젝트를 준비하는 데 유용합니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

이 API는 `GameLiftServerSDK.h` 및 `GameLiftServerSDKModels.h`에 정의되어 있습니다.

Unreal Engine 플러그인을 설치하려면 코드 샘플 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

- 작업
- 데이터 유형

AcceptPlayerSession()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift가 플레이어 세션 ID가 유효한지, 즉 플레이어 ID가 게임 세션의 플레이어 슬롯을 예약했는지 확인합니다. 검증되면 Amazon GameLift가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerSessionId)
```

파라미터

playerSessionId

AWS SDK Amazon GameLift API 작업 CreatePlayerSession에 호출에 대한 응답으로 Amazon GameLift 서비스에서 발급한 고유 ID입니다. 게임 클라이언트가 서버 프로세스에 연결할 때 이 ID를 참조합니다.

형식: FString

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

ActivateGameSession()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화가 완료된 후 `onStartGameSession()` 콜백 함수의 일부로 호출되어야 합니다.

구문

```
FGameLiftGenericOutcome ActivateGameSession()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

DescribePlayerSessions()

설정, 세션 메타데이터 및 플레이어 데이터 등의 플레이어 세션 데이터를 가져옵니다. 이 작업을 사용하면 단일 플레이어 세션 정보, 게임 세션에 있는 모든 플레이어 세션 정보 또는 단일 플레이어 ID와 관련된 모든 플레이어 세션 정보를 가져올 수 있습니다.

구문

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
 FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

파라미터

describePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 FDescribePlayerSessionsRequest 객체입니다.

필수 항목 여부: 예

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 FDescribePlayerSessionsRequest 객체를 반환합니다. 플레이어 세션 객체 구조는 AWS SDK Amazon GameLift API PlayerSession 데이터 유형과 동일합니다.

GetGameSessionId()

서버 프로세스가 활성인 경우 현재 서버 프로세스에서 호스팅하고 있는 게임 세션의 ID를 가져옵니다.

구문

```
FGameLiftStringOutcome GetGameSessionId()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 `FGameLiftStringOutcome` 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

GetInstanceCertificate()

플릿 및 해당 인스턴스와 연결된 pem 인코딩된 TLS 인증서의 파일 위치를 검색합니다. AWS Certificate Manager는 인증서 구성이 GENERATED로 설정된 새 플릿을 생성할 때 이 인증서를 생성합니다. 이 인증서를 사용하여 게임 클라이언트와 보안 연결을 설정하고 클라이언트/서버 통신을 암호화합니다

구문

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 인스턴스에 저장된 플릿의 TLS 인증서 파일과 인증서 체인 위치를 포함한 `GetInstanceCertificateOutcome` 객체를 반환합니다. 인증서 체인에서 추출한 루트 인증서 파일도 인스턴스에 저장됩니다. 실패하면 오류 메시지를 반환합니다.

인증서 및 인증서 체인 데이터에 대한 자세한 내용은 AWS Certificate Manager API 참조의 GetCertificate 응답 요소를 참조하세요.

GetSdkVersion()

현재 서버 프로세스에 빌드된 SDK 버전 번호를 반환합니다.

구문

```
FGameLiftStringOutcome GetSdkVersion();
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 현재 SDK 버전을 FGameLiftStringOutcome 객체로 반환합니다. 반환된 문자열에는 버전 번호만(예: “3.1.5”)을 포함합니다. 실패하면 오류 메시지를 반환합니다.

예

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Amazon GameLift SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift 관련 초기화가 진행되기 전, 시작 시 호출되어야 합니다.

구문

```
FGameLiftGenericOutcome InitSDK()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

ProcessEnding()

Amazon GameLift 서비스에 서버 프로세스를 중단할 준비가 되었음을 알립니다. 이 메서드는 모든 활성 게임 세션의 종료 등, 다른 모든 정리 작업 이후에 호출되어야 합니다. 이 메서드는 종료 코드 0으로

종료해야 합니다. 0이 아닌 종료 코드를 사용하면 프로세스가 정상적으로 종료되지 않았다는 이벤트 메시지가 표시됩니다.

구문

```
FGameLiftGenericOutcome ProcessEnding()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

ProcessReady()

Amazon GameLift 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 InitSDK()를 호출하고, 서버 프로세스가 게임 세션을 호스트하기 전에 수행해야 하는 설정 작업을 완료한 후에만 이 메서드를 호출할 수 있습니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

구문

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

파라미터

FProcessParameters

다음 서버 프로세스 관련 정보를 전달하는 FProcessParameters 객체입니다.

- 게임 서버 코드에 구현되어 있는 콜백 메서드의 이름으로, 서버 프로세스와 통신할 수 있도록 Amazon GameLift 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예

Unreal Engine 플러그인 사용 섹션에 있는 샘플 코드를 참조하세요.

RemovePlayerSession()

Amazon GameLift 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

파라미터

playerSessionId

AWS SDK Amazon GameLift API 작업 CreatePlayerSession에 호출에 대한 응답으로 Amazon GameLift 서비스에서 발급한 고유 ID입니다. 게임 클라이언트가 서버 프로세스에 연결할 때 이 ID를 참조합니다.

형식: FString

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

StartMatchBackfill()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. AWS SDK 작업 StartMatchBackfill()도 참조하세요. 이 작업을 사용하면 게임 세션을 호스팅하는 게임 서버 프로세스에서 매치 채우기 요청을 시작할 수 있습니다. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift 서비스는 콜백 함수 `OnUpdateGameSession()`을 사용하여 업데이트된 매치메이커 데이터를 전달합니다.

서버 프로세스는 한 번에 하나의 활성 매치 채우기 요청만 할 수 있습니다. 새 요청을 보내려면 먼저 StopMatchBackfill()을 호출하여 원본 요청을 취소해야 합니다.

구문

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
 &startBackfillRequest);
```

파라미터

FStartMatchBackfillRequest

다음 정보를 전달하는 FStartMatchBackfillRequest 객체입니다.

- 백필(backfill) 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift가 ID를 자동 생성합니다.
- 요청을 보낼 매치메이커. 전체 구성 ARN이 필요합니다. 이 값은 게임 세션의 매치메이커 데이터에서 수집할 수 있습니다.
- 백필(backfill) 중인 게임 세션의 ID.
- 게임 세션의 현재 플레이어에 대해 사용 가능한 매치메이킹 데이터.

필수 항목 여부: 예

반환 값

성공하면 매치 채우기 티켓을 `FGameLiftStringOutcome` 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다. 티켓 상태는 AWS SDK 작업 DescribeMatchmaking()을 사용하여 추적할 수 있습니다.

StopMatchBackfill()

StartMatchBackfill()을 통해 생성된 활성 매치 채우기 요청을 취소합니다. AWS SDK 작업 StopMatchmaking()도 참조하세요. FlexMatch 채우기 기능에 대해 자세히 알아보세요.

구문

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
 &stopBackfillRequest);
```

파라미터

StopMatchBackfillRequest

취소할 매치메이킹 티켓을 식별하는 FStopMatchBackfillRequest 객체입니다.

- 취소 중인 채우기 요청에 할당한 티켓 ID
- 채우기 요청을 보낸 매치메이커
- 채우기 요청과 연결된 게임 세션

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

TerminateGameSession()

버전 4.0.1 에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 ProcessEnding()을 호출해야 합니다.

Amazon GameLift 서비스에 서버 프로세스가 현재 게임 세션을 종료했음을 알립니다. 이 작업은 서버 프로세스가 활성 상태를 유지하고 새 게임 세션을 호스팅할 준비가 되면 호출됩니다. 서버 프로세스를 즉시 사용하여 새 게임 세션을 호스팅할 수 있음을 Amazon GameLift에 알리기 때문에 게임 세션 종료 절차가 완료된 후에만 호출해야 합니다.

게임 세션이 중지된 후 서버 프로세스가 종료되는 경우에는 이 작업이 호출되지 않습니다. 대신 ProcessEnding()을 호출하여 게임 세션과 서버 프로세스가 모두 종료되었음을 알립니다.

구문

```
FGameLiftGenericOutcome TerminateGameSession()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

UpdatePlayerSessionCreationPolicy()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다. (Amazon GameLift Service API 참조의 UpdateGameSession() 작업도 참조하세요.)

구문

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
 policy)
```

파라미터

Policy

게임 세션이 새 플레이어를 수락하는지 여부를 나타내는 값입니다.

유형: EPlayerSessionCreationPolicy 열거형입니다. 유효한 값으로는 다음이 포함됩니다.

- ACCEPT_ALL - 모든 새 플레이어 세션을 수락합니다.
- DENY_ALL - 모든 새 플레이어 세션을 거부합니다.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

Unreal Engine용 Amazon GameLift Server SDK 참조: 데이터 유형

이 Amazon GameLift Server API 참조는 Amazon GameLift와 함께 사용할 Unreal Engine 게임 프로젝트를 준비하는 데 유용합니다. 통합 프로세스에 대한 자세한 내용은 Amazon GameLift를 게임 서버에 추가 섹션을 참조하세요.

이 API는 GameLiftServerSDK.h 및 GameLiftServerSDKModels.h에 정의되어 있습니다.

Unreal Engine 플러그인을 설치하려면 코드 샘플 GameLift Amazon을 언리얼 엔진 프로젝트에 통합 섹션을 참조하세요.

- 작업
- 데이터 유형

FDescribePlayerSessionsRequest

이 데이터 형식은 검색할 플레이어 세션을 지정하는 데 사용됩니다. 다음과 같이 사용할 수 있습니다.

- 특정 플레이어 세션을 요청하려면 PlayerSessionId를 제공합니다.

- 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 GameSessionId를 제공합니다.
- 지정한 플레이어의 모든 플레이어 세션을 요청하려면 PlayerId를 제공합니다.

플레이어 세션이 대량일 경우 페이지 매김 파라미터를 사용하여 결과를 순차 블록으로 검색합니다.

목차

GameSessionId

고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 다음과 같습니다. `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. <ID string> 값은 사용자 지정 ID 문자열(게임 세션을 만들 때 지정한 경우) 또는 생성 문자열입니다.

유형: 문자열

필수 항목 여부: 아니요

한도

반환할 최대 결과 수입니다. 결과를 순차적인 일련의 페이지로 가져오려면 이 파라미터를 NextToken과 함께 사용합니다. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 정수

필수 항목 여부: 아니요

NextToken

결과의 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마십시오. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerId

사용자의 고유 식별자입니다. 플레이어 ID는 개발자에 의해 정의됩니다. 플레이어 ID 생성 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionId

플레이어 세션의 고유 식별자입니다.

유형: 문자열

필수 항목 여부: 아니요

PlayerSessionStatusFilter

결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. 다음과 같은 플레이어 세션 상태가 가능합니다.

- RESERVED - 플레이어 세션 요청이 수신되었지만, 플레이어가 서버 프로세스에 연결되지 않았거나 확인되지 않았습니다.
- ACTIVE - 플레이어가 서버 프로세스에 의해 확인되고 현재 연결되어 있습니다.
- COMPLETED - 플레이어 연결이 끊어졌습니다.
- TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60초) 이내에 연결하지 않았거나 확인되지 않았습니다.

유형: 문자열

필수 항목 여부: 아니요

FProcessParameters

이 데이터 유형에는 ProcessReady() 호출 시 Amazon GameLift 서비스로 보낸 파라미터 집합이 포함됩니다.

목차

port

서버 프로세스가 새 플레이어 연결을 수신 대기하는 포트 번호입니다. 이 값은 이 게임 서버 빌드를 전개하는 플릿에 대해 구성된 포트 범위에 속해야 합니다. 이 포트 번호는 게임 세션 및 플레이어 세션 객체에 포함되며, 게임 세션이 서버 프로세스에 연결할 때 이 포트 번호를 사용합니다.

유형: 정수

필수 항목 여부: 예

logParameters

게임 세션 로그 파일의 디렉터리 경로 목록을 포함하는 객체입니다.

유형: 어레이 <FString>

필수 항목 여부: 아니요

onStartGameSession

Amazon GameLift가 새 게임 세션을 활성화하기 위해 호출하는 콜백 함수 이름입니다. Amazon GameLift는 클라이언트 요청 CreateGameSession에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 GameSession 객체를 사용합니다(Amazon GameLift Service API 참조에서 정의함).

형식: FOnStartGameSession

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift 서비스가 서버 프로세스를 강제로 종료하기 위해 호출하는 콜백 함수 이름입니다. 이 함수를 호출한 후 Amazon GameLift는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답한 후 서버 프로세스를 종료합니다.

형식: FSimpleDelegate

필수 항목 여부: 아니요

onHealthCheck

Amazon GameLift 서비스가 서버 프로세스에 상태 보고서를 요청하기 위해 호출하는 콜백 함수의 이름입니다. Amazon GameLift는 60초마다 이 함수를 호출합니다. 이 함수를 호출한 후 Amazon GameLift는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상으로 기록합니다.

형식: FOnHealthCheck

필수 항목 여부: 아니요

onUpdateGameSession

업데이트된 게임 세션 객체를 서버 프로세스에 전달하기 위해 Amazon GameLift 서비스가 호출하는 콜백 함수 이름입니다. Amazon GameLift는 업데이트된 매치메이커 데이터를 제공하기

위해 매치 채우기 요청을 처리할 때 이 함수를 호출합니다. GameSession 객체, 상태 업데이트 (updateReason) 및 매치 채우기 티켓 ID를 전달합니다.

유형: FonUpdateGameSession

필수 항목 여부: 아니요

FStartMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 이 정보는 StartMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

고유한 게임 세션 식별자입니다. GetGameSessionId() API 작업은 ARN 형식의 식별자를 반환합니다.

형식: FString

필수 항목 여부: 예

MatchmakingConfigurationArn

매치메이커가 이 요청에 사용할 ARN 형식의 고유 식별자입니다. 원본 게임 세션을 만드는 데 사용된 매치메이커를 찾으려면 게임 세션 객체에서 매치메이커 데이터 속성을 확인합니다. 매치메이커 데이터에 대한 자세한 내용은 매치메이커 데이터를 사용하는 작업을 참조하세요.

형식: FString

필수 항목 여부: 예

플레이어

현재 게임 세션에 있는 모든 플레이어를 나타내는 데이터 세트입니다. 매치메이커는 이 정보를 사용하여 현재 플레이어와 적절하게 일치하는 새로운 플레이어를 검색합니다. 플레이어 객체 형식에 대한 자세한 내용은 Amazon GameLift API 참조 가이드를 참조하세요. 플레이어 속성, ID 및 팀 배정을 찾으려면 매치메이커 데이터 속성에서 게임 세션 객체를 확인합니다. 매치메이커에서 지연 시간을 사용하는 경우 현재 리전에 대한 업데이트 지연 시간을 수집하여 각 플레이어의 데이터에 포함합니다.

유형: TArray<FPlayer>

필수 항목 여부: 예

TicketId

매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 여기에 값이 제공되지 않으면 Amazon GameLift는 UUID 형식으로 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다.

형식: FString

필수 항목 여부: 아니요

FStopMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 취소하는 데 사용됩니다. 이 정보는 StopMatchBackfill() 호출을 통해 Amazon GameLift 서비스에 전달됩니다.

목차

GameSessionArn

취소 중인 요청과 연결된 고유한 게임 세션 식별자입니다.

형식: FString

필수 항목 여부: 예

MatchmakingConfigurationArn

이 요청을 보낸 매치메이커의 고유 식별자입니다.

형식: FString

필수 항목 여부: 예

TicketId

취소할 채우기 요청 티켓의 고유 식별자입니다.

형식: FString

필수 항목 여부: 예

게임 세션 배치 이벤트

Amazon은 각 게임 세션 배치 요청이 처리될 때 GameLift 해당 요청에 대해 이벤트를 내보냅니다. 게임 세션 배치의 이벤트 알림 설정에 설명된 대로 Amazon SNS 주제에 이러한 이벤트를 게시할 수 있습니다. 또한 이러한 CloudWatch 이벤트는 Amazon Events에 거의 실시간으로 그리고 최선의 노력을 기울여 내보냅니다.

이 주제에서는 게임 세션 배치 이벤트의 구조를 설명하고 각 이벤트 유형의 예를 제공합니다.
게임 세션 배치 요청 상태에 대한 자세한 내용은 Amazon GameLift API 참조를 참조하십시오 GameSessionPlacement.

배치 이벤트 구문

이벤트는 JSON 객체로 표현됩니다. 이벤트 구조는 유사한 최상위 필드 및 서비스별 세부 정보를 포함하는 CloudWatch Events 패턴을 따릅니다.

최상위 필드에는 다음이 포함됩니다(자세한 내용은 이벤트 패턴 참조).

version

이 필드는 항상 0으로 설정됩니다.

id

이벤트의 고유 추적 식별자입니다.

detail-type

값은 항상 `GameLift Queue Placement Event`입니다.

source

값은 항상 `aws.gamelift`입니다.

account

Amazon을 관리하는 데 사용되는 AWS GameLift 계정입니다.

시간

이벤트 타임스탬프입니다.

region

배치 요청이 처리되고 있는 AWS 지역. 사용 중인 게임 세션 대기열이 있는 리전입니다.

resources

배치 요청을 처리하는 게임 세션 대기열의 ARN 값입니다.

PlacementFulfilled

배치 요청이 성공적으로 처리되었습니다. 새 게임 세션이 시작되었고 게임 세션 배치 요청에 나열된 각 플레이어에 대한 새 플레이어 세션이 생성되었습니다. 플레이어 연결 정보를 확인할 수 있습니다.

상세 구문:

placementId

게임 세션 배치 요청에 할당된 고유 식별자입니다.

포트

새 게임 세션의 포트 번호입니다.

gameSessionArn

새 게임 세션에 대한 ARN 식별자입니다.

ipAddress

게임 세션의 IP 주소입니다.

dnsName

새 게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값 형식은 게임 세션을 실행하는 인스턴스가 TLS를 지원하는지 여부에 따라 달라집니다. TLS 지원 플릿에서 게임 세션에 연결할 때는 플레이어는 IP 주소가 아닌 DNS 이름을 사용해야 합니다.

TLS 지원 플릿: `<unique identifier>.<region identifier>.amazongamelift.com`.

TLS 미지원 플릿: `ec2-<unique identifier>.compute.amazonaws.com`.

startTime

이 요청이 대기열에 배치된 시간을 나타내는 타임스탬프입니다.

endTime

이 요청이 이행된 시간을 나타내는 타임스탬프입니다.

gameSessionRegion

AWS 게임 세션을 호스팅하는 플릿의 지역. 이는 의 지역 토큰에 해당합니다`GameSessionArn`.

placedPlayerSessions

게임 세션 배치 요청에서 각 플레이어에 대해 생성된 플레이어 세션 모음입니다.

예

```
{
  "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

GameLift 서비스에 전화를 걸어 배치 요청이 취소되었습니다 StopGameSessionPlacement.

세부 정보:

placementId

게임 세션 배치 요청에 할당된 고유 식별자입니다.

startTime

이 요청이 대기열에 배치된 시간을 나타내는 타임스탬프입니다.

endTime

이 요청이 취소된 시간을 나타내는 타임스탬프입니다.

예

```
{
  "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

대기열의 제한 시간이 만료되기 전에 게임 세션 배치가 성공적으로 완료되지 않았습니다. 배치 요청은 필요에 따라 다시 제출할 수 있습니다.

세부 정보:

placementId

게임 세션 배치 요청에 할당된 고유 식별자입니다.

startTime

이 요청이 대기열에 배치된 시간을 나타내는 타임스탬프입니다.

endTime

이 요청이 취소된 시간을 나타내는 타임스탬프입니다.

예

```
{
  "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

GameLift Amazon은 게임 세션 요청을 처리할 수 없었습니다. 이는 일반적으로 예상치 못한 내부 오류로 인해 발생합니다. 배치 요청은 필요에 따라 다시 제출할 수 있습니다.

세부 정보:

placementId

게임 세션 배치 요청에 할당된 고유 식별자입니다.

startTime

이 요청이 대기열에 배치된 시간을 나타내는 타임스탬프입니다.

endTime

이 요청에 실패한 시간을 나타내는 타임스탬프입니다.

예

```
{
  "version": "0",
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "252386620677",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {

    "type": "PlacementFailed",
    "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

Amazon GameLift의 추정 요금 산출

AWS Pricing Calculator를 사용하여 Amazon GameLift의 추정 요금을 생성할 수 있습니다. 계산기를 사용하기 위해 AWS 계정 또는 AWS에 대한 세부적인 지식이 필요하지 않습니다.

AWS Pricing Calculator 계산기는 서비스 비용에 영향을 미치는 결정 과정을 안내하여 Amazon GameLift의 게임 프로젝트 비용을 파악할 수 있게 해줍니다. Amazon GameLift를 어떻게 사용할 계획인지 아직 확실하지 않은 경우 기본값을 사용하여 추정치를 산출합니다. 프로덕션 사용량을 계획할 때 계산기를 사용하면 잠재적 시나리오를 테스트하고 더 정확한 추정치를 생성할 수 있습니다.

AWS Pricing Calculator를 사용하여 다음과 같은 Amazon GameLift 호스팅 옵션에 대한 추정치를 산출할 수 있습니다.

- Amazon GameLift 호스팅 추정
- Amazon GameLift 독립형 FlexMatch 추정

Amazon GameLift 호스팅 추정

이 옵션은 서버 인스턴스 사용량 및 데이터 전송 비용을 포함하여 Amazon GameLift 관리 서버에서 게임을 호스팅하는 데 드는 추정 비용을 제공합니다. FlexMatch 매치메이킹은 Amazon GameLift 관리형 호스팅 비용에 포함되어 있습니다.

둘 이상의 AWS 리전 또는 둘 이상의 인스턴스 유형에서 게임 서버를 호스팅하거나 호스팅할 계획이라면 각 리전 및 인스턴스 유형에 대한 추정치를 산출합니다.

Amazon GameLift 인스턴스

이 섹션에서는 플레이어를 위한 게임 세션을 호스팅하는 데 필요한 컴퓨팅 리소스의 유형과 수를 추정하는 데 도움이 됩니다. Amazon GameLift는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 사용하여 게임 서버를 관리합니다. Amazon GameLift에서는 특정 인스턴스 유형 및 운영 체제를 갖춘 인스턴스 플릿을 배포합니다. 플릿을 여러 개 보유하고 있거나 보유할 계획이라면 각 플릿에 대한 추정치를 산출합니다.

시작하려면 AWS Pricing Calculator의 Amazon GameLift 구성 페이지를 엽니다. 설명을 추가하고 리전을 선택한 다음 Amazon GameLift 호스팅 추정(인스턴스+데이터 전송)을 선택합니다. Amazon GameLift 인스턴스에서 다음 필드를 작성합니다.

- 최대 동시 플레이어 수(최대 CCU)

게임 서버에 동시에 연결할 수 있는 최대 플레이어 수입니다. 이 필드는 Amazon GameLift가 최대 플레이어 수요를 충족하는 데 필요한 호스팅 용량을 나타냅니다. 선택한 AWS 리전의 인스턴스를 사용하여 호스팅할 것으로 예상되는 일일 최대 플레이어 수를 입력합니다.

예를 들어 1,000명의 플레이어가 한 번에 게임에 연결할 수 있게 하려면 기본값인 **1000**을 유지하세요.

- 일일 최대 CCU를 백분율로 나타낸 시간당 평균 CCU

  24시간 동안 시간당 평균 동시 플레이어 수입니다. 이 값을 사용하여 Amazon GameLift가 플레이어를 위해 유지해야 하는 지속적인 호스팅 용량의 양을 추정합니다. 사용할 백분율 값을 잘 모르는 경우 기본값인 **50**%를 유지합니다. 플레이어 수요가 안정적인 게임의 경우 **70** 백분율 값을 입력하는 것이 좋습니다.

  예를 들어 게임의 시간당 평균 CCU가 6,000이고 최대 CCU가 10,000인 경우 **60** 백분율 값을 입력합니다.

- 인스턴스당 게임 세션

  각 게임 서버 인스턴스가 동시에 호스팅할 수 있는 게임 세션 수입니다. 이 번호에 영향을 미칠 수 있는 요인으로는 게임 서버의 리소스 요구사항, 각 게임 세션에서 호스팅할 플레이어 수, 플레이어의 기대 성능 등이 있습니다. 게임의 동시 게임 세션 수를 알고 있다면 그 값을 입력합니다. 또는 기본값 **20**을 그대로 유지합니다.

- 게임 세션당 플레이어 수

  게임 디자인에 정의된 대로 게임 세션에 연결하는 평균 플레이어 수입니다. 플레이어 수가 다른 게임 모드가 있는 경우 게임 전체의 게임 세션당 평균 플레이어 수를 추정합니다. 기본값은 **8**입니다.

- 인스턴스 유휴 버퍼%

  갑작스럽게 급증하는 플레이어 수요를 처리하기 위해 비축해 두어야 하는 미사용 호스팅 용량의 비율입니다. 버퍼 크기는 플릿에 대한 총 인스턴스 수의 비율입니다. 기본값은 **10**%입니다.

  예를 들어 유휴 버퍼가 20%인 경우 활성 인스턴스가 100개인 플레이어를 지원하는 플릿은 20개의 유휴 인스턴스를 유지합니다.

- 스팟 인스턴스 %

  Amazon GameLift 플릿은 온디맨드 인스턴스 및 스팟 인스턴스의 조합을 사용할 수 있습니다. 온디맨드 인스턴스는 보다 안정적인 가용성을 제공하지만 스팟 인스턴스는 비용 효율성이 매우 높은 대안을 제공합니다. 비용 절감과 가용성을 모두 최적화하려면 이 조합을 사용하는 것이 좋습니다.

Amazon GameLift에서 스팟 인스턴스를 사용하는 방법에 대한 자세한 내용은 온디맨드 인스턴스 및 스팟 인스턴스 비교 섹션을 참조하세요.

이 필드에는 플릿에서 유지할 스팟 인스턴스의 비율을 입력합니다. 스팟 인스턴스 비율을 50~85% 사이로 설정하는 것이 좋습니다. 기본값은 **50**%입니다.

예를 들어 100개의 인스턴스가 포함된 플릿을 배포하고 **40** 백분율을 지정하는 경우 Amazon GameLift는 60개의 온디맨드 인스턴스 및 40개의 스팟 인스턴스를 유지합니다.

- 인스턴스 유형

  Amazon GameLift 플릿은 컴퓨팅 성능, 메모리, 스토리지 및 네트워킹 기능이 서로 다른 다양한 Amazon EC2 인스턴스 유형을 사용할 수 있습니다. Amazon GameLift 플릿을 구성할 때 게임 요구 사항에 가장 적합한 인스턴스 유형을 선택합니다. Amazon GameLift에서 인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 Amazon GameLift 컴퓨팅 리소스 선택 섹션을 참조하세요.

  Amazon GameLift 플릿에서 사용 중이거나 사용할 예정인 인스턴스 유형을 알고 있는 경우 해당 유형을 선택합니다. 어떤 유형을 선택해야 할지 잘 모르겠으면 c5.large를 선택하는 것이 좋습니다. 이 유형은 평균 크기와 성능을 갖춘 고가용성 유형입니다.

- 운영 체제

  이 필드는 게임 서버가 실행되는 운영 체제(Linux 또는 Windows)를 지정합니다. 기본값은 Linux입니다.

데이터 전송(DTO)

이 섹션에서는 게임 클라이언트와 게임 서버 간의 트래픽 비용을 추정할 수 있습니다. 데이터 전송 요금은 아웃바운드 트래픽에만 적용됩니다. 인바운드 데이터 전송에는 요금이 부과되지 않습니다.

AWS Pricing Calculator의 Amazon GameLift 구성 페이지에서 데이터 전송을 확장한 후 다음 필드를 작성합니다.

- DTO 추정 유형

  게임의 데이터 전송을 추적하는 방법에 따라 다음 두 가지 방법 중 하나로 DTO를 추정할 수 있습니다.

  - 매월(GB당) - 게임 서버의 월간 트래픽을 추적하는 경우 이 유형을 선택합니다.
  - 플레이어당 - 플레이어당 데이터 전송을 추적하는 경우 이 유형을 선택합니다. 이는 기본 유형입니다.

다음 필드에서는 이전 섹션에서 계산한 플레이어 시간을 기반으로 플레이어당 DTO를 추정합니다.

- 월별 DTO(GB당)

  매월(GB당) DTO 추정 유형을 선택한 경우 각 인스턴스에서 리전별 예상 월간 DTO 사용량을 GB 단위로 입력합니다.
- 플레이어당 DTO

  플레이어당 DTO 추정 유형을 선택한 경우 플레이어당 게임 추정 DTO 사용량을 KB/초 단위로 입력합니다. 기본값은 **4**입니다.

Amazon GameLift 추정 요금을 구성했으면 내 추정치에 추가를 선택합니다. AWS Pricing Calculator에서 추정치를 생성하고 관리하는 방법에 대한 자세한 내용은 AWS Pricing Calculator 사용 설명서의 추정치 생성, 서비스 구성 및 기타 서비스 추가를 참조하세요.

Amazon GameLift 독립형 FlexMatch 추정

이 옵션은 다른 게임 서버 솔루션으로 게임을 호스팅하는 동안 FlexMatch 매치메이킹을 독립형 서비스로 사용하는 데 드는 추정 비용을 제공합니다. 여기에는 FleetIQ를 포함한 Amazon GameLift 자체 관리형 호스팅과 온프레미스 호스팅, 피어 투 피어 또는 클라우드 컴퓨팅 기본 데이터 유형이 포함됩니다. 독립형 FlexMatch 비용은 사용된 컴퓨팅 성능을 기준으로 합니다.

여러 AWS 리전에 매치메이커가 두 개 이상 있거나 운영할 계획인 경우 각 리전에 대한 추정치를 산출합니다.

> **Note**
>
> Amazon GameLift FlexMatch는 미국 동부(버지니아 북부), 미국 서부(오레곤), 아시아 태평양(서울), 아시아 태평양(시드니), 아시아 태평양(도쿄), 유럽(프랑크푸르트), 유럽(아일랜드)과 같은 리전에서 사용할 수 있습니다.

시작하려면 AWS Pricing Calculator의 Amazon GameLift 구성 페이지를 엽니다. 설명을 추가하고 리전을 선택한 다음 Amazon GameLift 독립형 FlexMatch 추정치를 선택합니다. Amazon GameLift FlexMatch에서 다음 필드를 작성합니다.

- 최대 동시 플레이어 수(최대 CCU)

게임 서버에 동시에 연결하고 매치메이킹을 요청할 수 있는 최대 플레이어 수입니다. 선택한 리전의 게임 세션에 매칭될 것으로 예상되는 일일 최대 플레이어 수를 입력합니다.

예를 들어 1,000명의 플레이어가 한 번에 매칭할 수 있게 하려면 기본값인 **1000**을 유지하세요.

- 일일 최대 CCU를 백분율로 나타낸 시간당 평균 CCU

  24시간 동안 시간당 평균 동시 플레이어 수입니다. 이 값은 매치메이킹 요청량을 추정하는 데 도움이 됩니다. 사용할 백분율 값을 잘 모르는 경우 기본값인 **50**%를 유지합니다. 플레이어 수요가 안정적인 게임의 경우 **70** 백분율 값을 입력하는 것이 좋습니다.

  예를 들어 게임의 시간당 평균 CCU가 6,000이고 최대 CCU가 10,000인 경우 **60** 백분율 값을 입력합니다.

- 매치당 플레이어 수

  게임 디자인에 정의된 대로 게임 세션에 매칭하는 평균 플레이어 수입니다. 플레이어 수가 다른 게임 모드가 있는 경우 게임 전체의 게임 세션당 평균 플레이어 수를 추정합니다. 기본값은 **8**입니다.

- 게임 시간(분)

  플레이어가 게임 세션을 처음부터 끝까지 플레이하는 평균 시간입니다. 이 값은 플레이어들이 새 매치를 요구하는 빈도를 결정하는 데 도움이 됩니다. 플레이어의 평균 게임 시간을 분 단위로 입력합니다. 기본값은 **1**입니다.

- 매치메이킹 규칙의 복잡성

  매치메이킹 규칙의 복잡성이란 플레이어를 매칭하는 데 사용하는 규칙의 수와 유형을 의미합니다. 규칙 세트의 복잡성 수준은 각 매치에 필요한 컴퓨팅 성능을 결정하는 데 도움이 됩니다.

  - 복잡성 감소 - 매치메이킹 규칙 세트에 포함된 규칙이 적고, 더 간단한 규칙 유형(예: 비교 규칙)을 사용하며, 적은 시도로 성공적인 매치를 형성하는 규칙이 있는 경우 이 옵션을 선택합니다.
  - 복잡성 증가 - 매치메이킹 규칙 세트에 여러 규칙이 포함되어 있고, 더 복잡한 규칙 유형(예: 거리 또는 지연 규칙)을 사용하며, 제한적인 규칙으로 인해 오류가 더 많이 발생하고 더 많은 매칭 시도가 필요한 경우 이 옵션을 선택합니다.

  규칙의 복잡성 및 요금에 대한 자세한 내용은 Amazon GameLift 요금 페이지에서 Amazon GameLift FlexMatch를 참조하세요.

Amazon GameLift FlexMatch 추정 요금을 구성했으면 내 추정치에 추가를 선택합니다. AWS Pricing Calculator에서 추정치를 생성하고 관리하는 방법에 대한 자세한 내용은 AWS Pricing Calculator 사용 설명서의 추정치 생성, 서비스 구성 및 기타 서비스 추가를 참조하세요.

할당량 및 지원 리전

AWS Amazon GameLift Service Quotas에 대해서는 Amazon GameLift 할당량을 참조하세요.

AWS 리소스에 대한 할당량 증가 요청에 대한 자세한 정보는 AWS Service Quotas를 참조하세요.

AWS 리전 지원되는 Amazon GameLift의 목록에 대해서는 Amazon GameLift 리전을 참조하세요.

아마존 GameLift 릴리스 노트

Amazon GameLift 릴리스 노트는 서비스와 관련된 새로운 기능, 업데이트 및 수정 사항에 대한 세부 정보를 제공합니다.

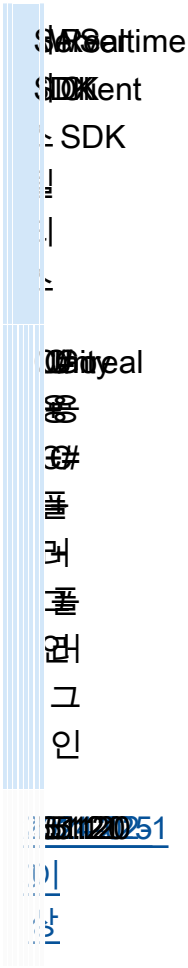
SDK 버전

다음 표에는 SDK 버전 정보가 포함된 모든 Amazon GameLift 릴리스가 나열되어 있습니다. 게임 서버 및 클라이언트 통합에 유사한 SDK를 사용할 필요는 없습니다. 하지만 한 SDK의 이전 버전은 다른 SDK의 최신 기능을 완전히 지원하지 않을 수 있습니다.

Amazon GameLift SDK에 대한 자세한 내용은 을 참조하십시오아마존을 통한 개발 지원 GameLift.

최신 Amazon GameLift SDK를 다운로드하려면 Amazon GameLift SDK 다운로드 사이트를 참조하십시오.

현재 버전



이전 버전

서비스 릴리스	AWS SDK	Server SDK				Realti Client SDK
		Unity용 C# 플러그인	C++	Unreal용 C++ 플러그인	Go	
2023-12-14	1.11.225 이상	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-11-02	1.11.193 이상	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-09-28	1.11.144 이상	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-08-17	1.11.144 이상	5.1.0	5.1.1	5.1.0	5.0.0	1.2.0
2023-07-27	1.11.111 이상	5.1.0	5.1.0	5.0.2	5.0.0	1.2.0
2023-06-29	1.11.111 이상		5.0.4	5.0.2	5.0.0	1.2.0
2023-06-15	1.11.87 이상		5.0.4	5.0.2	5.0.0	1.2.0

서비스 릴리스	AWS SDK	Server SDK				Realti Client SDK
		Unity용 C# 플러그인	C++	Unreal용 C++ 플러그인	Go	
2023-05-25	1.11.87 이상		5.0.3	5.0.2	5.0.0	1.2.0
2023-04-20	1.11.63 이상				5.0.0	1.2.0
2023-04-13	1.10.21 이상			5.0.0	5.0.0	1.2.0
2023-02-09	1.10.21 이상			3.4.0	5.0.0	1.2.0
2023-01-31	1.10.21 이상		5.0.0	3.4.0	5.0.0	1.2.0
2022-12-01	1.10.21 이상		5.0.0	3.4.0		1.2.0
2022-08-25	1.9.333 이상		3.4.2	3.4.0		1.2.0
2021-10-28	1.9.133 이상		3.4.2	3.4.0		1.2.0
2021-06-03	1.8.168 이상		3.4.2	3.4.0		1.2.0
2021-03-23	1.8.168 이상		3.4.1	3.3.3		1.1.0
2021-03-16	1.8.163 이상		3.4.1	3.3.3		1.1.0
2021-02-09	1.8.139 이상		3.4.1	3.3.3		1.1.0
2020-12-22	1.8.95 이상		3.4.1	3.3.3		1.1.0

서비스 릴리스	AWS SDK	Server SDK				Realti Client SDK
		Unity용 C# 플러그인	C++	Unreal용 C++ 플러그인	Go	
2020-11-24	1.8.95 이상		3.4.1	3.3.2		1.1.0
2020-11-11	1.8.36 이상		3.4.1	3.3.2		1.1.0
2020-09-17	1.8.36 이상		3.4.1	3.3.2		1.1.0
2020-08-27	1.7.310 이상		3.4.0	3.3.1		1.1.0
2020-04-16	1.7.310 이상		3.4.0	3.3.1		1.1.0
2020년 4월 2일	1.7.310 이상		3.4.0			1.1.0
2019년 12월 19일	1.7.249 이상		3.4.0			1.1.0
2019-11-14	1.7.210 이상		3.4.0			1.1.0
2019-10-24	1.7.210 이상		3.4.0			1.1.0
2019-09-03	1.7.175 이상		3.4.0			1.1.0
2019-07-09	1.7.140 이상		3.3.0			1.0.0
2019-04-25	1.7.91 이상		3.3.0			1.0.0
2019-03-07	1.7.65 이상		3.3.0			
2019-02-07	1.7.45 이상		3.3.0			
2018-12-14	1.6.20 이상		3.3.0			
2018-09-27	1.6.20 이상		3.2.1			

서비스 릴리스	AWS SDK	Server SDK				Realti Client SDK
		Unity용 C# 플러그인	C++	Unreal용 C++ 플러그인	Go	
2018-06-14	1.4.47 이상		3.2.1			
2018-05-10	1.4.47 이상		3.2.1			
2018-02-15	1.3.58 이상		3.2.1			
2018-02-08	1.3.52 이상		3.2.0			
2017-09-01	1.1.43 이상		3.1.7			
2017-08-16	1.1.31 이상		3.1.7			
2017-05-16	1.0.122 이상		3.1.5			
2017-04-11	1.0.103 이상		3.1.5			
2017-02-21	1.0.72 이상		3.1.5			
2016-11-18	1.0.31 이상		3.1.0			
2016-10-13	1.0.17 이상		3.1.0			
2016-09-01	0.14.9 이상		3.1.0			
2016-08-04	0.12.16 이상		3.0.7			

릴리스 정보

다음 릴리스 정보는 시간 순서로 나열되어 있으며 최신 업데이트가 먼저 나열됩니다. GameLift 아마존은 2016년에 처음 출시되었습니다. 여기에 나열된 릴리스 정보보다 이전 릴리스 정보는 SDK 버전에서 릴리스 날짜 링크를 참조하십시오.

2024년 4월 24일: 아마존, 컨테이너 GameLift 플릿 출시

GameLift Amazon은 이제 향상된 휴대성, 확장성, 내결함성 및 민첩성을 제공하는 컨테이너 플릿 미리 보기를 제공합니다.

컨테이너 플릿에서 Amazon EC2 인스턴스는 하나 이상의 컨테이너를 호스팅합니다. 이러한 컨테이너에는 종속성 및 구성을 포함하여 게임 서버와 필요한 모든 것이 포함됩니다. 종속성의 예로는 SDK와 소프트웨어 패키지가 있습니다. 프라이빗 Amazon Elastic Container 레지스트리에 컨테이너를 업로드하면 Amazon에서 해당 컨테이너로 GameLift 플릿을 채웁니다.

컨테이너 플릿에서 작동하려면 게임 서버를 Linux에서 실행하고 Server SDK 5.x와 통합해야 합니다. 컨테이너 플릿에서는 호스팅 리소스를 세밀하게 제어할 수 있으므로 CPU 유닛 및 메모리와 같은 리소스 사용을 최적화할 수 있습니다. 또한 컨테이너에 여러 게임 서버를 호스팅하여 리소스 사용을 줄일 수 있습니다.

컨테이너 플릿에서는 온디맨드 인스턴스 유형, 조정 (자동 및 수동), 대기열, 매치메이킹 등 다른 유형의 플릿이 제공하는 것과 동일한 많은 이점을 얻을 수 있습니다. 또한 다른 플릿 유형과 동일한 메트릭을 사용할 수 있으며 컨테이너에 대한 몇 가지 새로운 지표도 제공됩니다. 컨테이너 플릿을 사용하면 다음 위치 지역의 플레이어에게 전 세계 도달 범위를 넓힐 수 있습니다.

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

더 많은 지역과 로컬 존에 도달하려면 여러 위치에 있는 컨테이너 플릿을 생성하세요.

자세히 알아보기:

- Amazon GameLift 컨테이너를 사용한 호스팅 관리, Amazon GameLift 개발자 가이드
- CreateContainerGroupDefinition, 아마존 GameLift API 레퍼런스

2024년 2월 13일: 아마존, SDK 개선 GameLift 출시 및 언리얼 엔진용 Amazon GameLift 플러그인 설치 간소화

SDK 버전 업데이트:

- 고 서버 SDK, 버전 5.1.0
- C# 서버 SDK, 버전 5.1.2
- C++ 서버 SDK, 버전 5.1.2

다음과 같이 개선되었습니다.

- 네트워크 중단 시 자동 재연결을 추가하여 SDK의 안정성을 개선했습니다.
- [Go] 이제 서버 매개변수를 사용하거나 사용하지 않고 `InitSDK()` 호출할 수 있습니다. Amazon GameLift 관리형 EC2 플릿에서 실행되는 게임 서버는 환경 변수에서 직접 서버 파라미터를 읽습니다. Amazon GameLift Anywhere 플릿의 게임 서버는 서버 매개변수를 `InitSDK()` 사용하여 호출해야 합니다.

업데이트된 플러그인 버전:

- 언리얼 엔진용 아마존 GameLift 플러그인, 버전 1.1.0
- 유니티용 아마존 GameLift 플러그인, 버전 2.1.0
- 언리얼용 C++ 서버 SDK 플러그인, 버전 5.1.1
- Unity용 C# 서버 SDK 플러그인, 버전 5.1.2

다음과 같은 개선 사항이 적용되었습니다.

- [언리얼 엔진용 Amazon GameLift 플러그인] 설치 지침이 업데이트되고 패키징이 간소화되었습니다. 이제 이 플러그인에 최신 버전의 언리얼용 C++ Server SDK가 포함되어 있습니다.
- 최신 버전의 Server SDK를 지원하도록 플러그인을 업그레이드했습니다. GameLift

자세히 알아보기:

- 언리얼 엔진용 Amazon GameLift 플러그인과 게임 통합, Amazon 개발자 가이드 GameLift
- 아마존 GameLift 플러그인 및 SDK 다운로드

2023년 12월 14일: Amazon, 활성 게임 세션의 게임 속성 업데이트 기능 GameLift 추가

이미 게임 세션을 생성할 때 게임 속성을 설정하고 게임 세션에서 지정된 속성을 검색할 수 있었습니다. 이제 활성 게임 세션에서 이러한 속성을 추가하고 업데이트할 수도 있습니다.

예를 들어 플레이어가 플레이하고 싶은 맵에 투표하는 경우를 예로 들 수 있습니다. 게임 클라이언트가 `GameProperty` 값을 `UpdateGameSession` 수정하기 위해 `{"Key": "map", "Value":"jungle"}` 호출합니다. 그러면 게임이 게임 세션의 플레이어를 위해 새 맵을 구현합니다.

게임 관리자는 `SearchGameSessions` 작업을 사용하여 게임 속성에서 유용한 데이터를 검색할 수도 있습니다. 예를 들어 관리자는 `Status` 값이 다음과 같은 게임 세션을 나열할 수 `{"Key": "map", "Value":"desert"}` 있습니다. `ACTIVE`

자세히 알아보기:

- the section called “ GameLift Amazon을 게임 클라이언트에 추가”, 아마존 GameLift 개발자 가이드
- GameProperty, 아마존 GameLift API 레퍼런스
- UpdateGameSession, 아마존 GameLift API 레퍼런스
- SearchGameSessions, 아마존 GameLift API 레퍼런스

2023년 11월 21일: Amazon, Terraform 및 Pulumi와 같은 코드형 인프라 도구에 대한 지원 GameLift 시작: AWS Cloud Control API

이제 코드형 인프라 (IaC) 도구를 사용하여 전체 Amazon GameLift 리소스 스택을 관리할 수 있습니다. 이러한 도구에는 Terraform 및 Pulumi와 같은 타사 도구도 포함됩니다 AWS CloudFormation. 이 추가 지원을 통해 이제 게임 구축에 집중하고 DevOps 전략을 활용하여 리소스 관리, CI/CD 및 고객 배포를 처리할 수 있습니다.

또한 이제 AWS 클라우드 제어 API를 사용하여 모든 Amazon GameLift 리소스 유형을 프로비저닝하고 구성할 수 있습니다. Amazon GameLift API 또는 Amazon용 AWS CloudFormation 템플릿을 사용하여 리소스를 계속 사용할 수 있습니다. GameLift

IaC를 통해 사용할 수 있는 Amazon GameLift 리소스에 대한 자세한 내용은 Amazon GameLift 리소스 유형 참조 Amazon GameLift 리소스 유형 참조를 참조하십시오.

또한 이제 새 플릿 속성을 사용하여 AWS CloudFormation 템플릿 또는 AWS Cloud Control API를 사용하여 플릿을 자동으로 확장할 수 있습니다. `ScalingPolicies`

Cloud Control API는 개발자에게 수백 개의 AWS 서비스와 Terraform 및 Pulumi와 같은 여러 타사 도구에서 리소스 (CRUDL) 를 생성, 읽기, 업데이트, 삭제 및 나열할 수 있는 표준 API 세트를 제공합니다.

자세히 알아보기:

- AWS CloudFormation
- AWS 클라우드 제어 API
- AWS CC 테라폼 프로바이더
- 풀루미

2023년 11월 16일: 아마존, 유니티용 스탠드얼론 플러그인 GameLift 업데이트

SDK 버전 업데이트: 유니티용 아마존 GameLift 플러그인, 버전 2.0.0

Unity용 Amazon GameLift 플러그인은 Amazon에서 클라우드 호스팅을 위해 Unity 게임을 설치하고 실행하는 단계를 간소화하는 도구와 워크플로를 제공합니다. GameLift GameLift Amazon은 게임 개발자가 세션 기반 멀티플레이어 게임 전용 게임 서버를 관리하고 확장할 수 있는 완전 관리형 서비스입니다.

이번 버전에서는 Unity 플러그인이 서버 SDK 버전 5.x 및 Amazon Anywhere를 통한 로컬 테스트 지원 등 Amazon의 최신 GameLift 기능을 사용하도록 업데이트되었습니다. GameLift 플러그인은 유니티 버전 유니티 2021.3 LTS 및 2022.3 LTS와 호환됩니다.

주요 플러그인 기능은 다음과 같습니다.

- 다음 시나리오에 대한 Unity 에디터의 가이드 UI 워크플로:
  - 로컬 워크스테이션을 호스트로 GameLift 사용하여 Amazon과의 게임 통합을 테스트하십시오. 이 워크플로는 로컬 머신용 Amazon GameLift Anywhere 플릿을 설정하고, 게임 서버 및 클라이언트의 인스턴스를 시작하고, Amazon을 통해 게임 세션을 요청하고 GameLift, 게임에 참여하는 데 도움이 됩니다.
  - Amazon GameLift 관리형 EC2 및 지원 AWS 리소스를 사용하여 통합 게임 서버용 클라우드 호스팅 솔루션을 배포하십시오. 이 워크플로는 게임을 클라우드 호스팅용으로 구성하는 데 도움이 되며 세 가지 배포 시나리오를 제공합니다.
    - 게임 서버를 단일 플릿에 배포하십시오.
    - 여러 AWS 지역의 저렴한 스팟 플릿 세트에 게임 서버를 배포하십시오.
    - 매치메이커와 함께 게임 서버를 배포하세요. FlexMatch
- AWS 계정 사용자와 연결되는 사용자 프로필을 설정하고 기본 AWS 지역을 설정할 수 있습니다. 여러 프로필을 유지 관리하여 여러 AWS 계정, 계정 사용자, 지역에서 작업할 수 있습니다.
- Amazon GameLift 통합 및 배포 프로세스를 간소화하는 데 도움이 되는 특별한 편의 기능은 다음과 같습니다.

  - 각 호스팅 솔루션에는 고유한 플레이어 ID와 플레이어 검증을 제공하는 Amazon Cognito 사용자 풀을 비롯한 지원 AWS 리소스가 포함되어 있습니다. 솔루션에는 스토리지용 Amazon S3 버킷, Amazon SNS 이벤트 알림, AWS Lambda 함수 및 기타 리소스도 포함됩니다.
  - Anywhere워크플로의 경우 플러그인은 필수 서버 파라미터 설정을 자동화합니다.
  - Amazon EC2 워크플로의 경우, 각 배포 솔루션은 Lambda 함수를 사용하여 내장된 클라이언트 백엔드 서비스를 제공합니다. 백엔드 서비스는 게임 클라이언트와 Amazon 서비스 사이에 위치하며 Amazon GameLift 서비스에 대한 모든 직접 호출을 관리합니다. GameLift
- 게임 서버 및 게임 클라이언트 통합을 설명하기 위한 간단한 샘플 멀티플레이어 게임의 에셋과 코드를 포함한 통합 테스트용 콘텐츠.
- 자세한 통합 지침 및 샘플 코드가 포함된 플러그인 설명서.

for Anywhere 및 Amazon EC2 플릿을 포함한 모든 배포 시나리오에서는 AWS CloudFormation 템플릿을 사용하여 게임 솔루션용 AWS 리소스를 설명하고 배포합니다. 이러한 템플릿은 Amazon GameLift 플러그인 다운로드에 포함되어 있습니다. 그대로 사용하거나 게임에 맞게 사용자 지정할 수 있습니다.

자세히 알아보기:

- 서버 SDK 5.x용 유니티용 아마존 GameLift 플러그인 가이드, 아마존 GameLift 개발자 가이드
- 에서 플러그인을 다운로드하십시오. GitHub
- 아마존 GameLift 호스팅에 대해
- 아마존 GameLift 포럼

2023년 11월 2일: 아마존, 공유 자격 증명에 대한 지원 GameLift 추가

SDK 버전 업데이트: AWS SDK 1.11.193

새로운 Amazon GameLift 공유 자격 증명 기능을 사용하면 관리형 EC2 플릿에 배포된 애플리케이션이 다른 AWS 리소스와 상호 작용할 수 있습니다. 이 업데이트는 서버 SDK 버전 5.x 이상과 통합된 게임 서버 바이너리와 함께 번들링하여 배포하는 애플리케이션에 영향을 미칩니다. (게임 서버 실행 파일은 이미 Server SDK 5.x `GetFleetRoleCredentials()` 작업을 사용하여 자격 증명을 요청할 수 있습니다.)

예를 들어 Amazon CloudWatch 에이전트를 사용하여 게임 서버 빌드를 배포하여 EC2 인스턴스 지표 및 기타 데이터를 수집하려는 경우 에이전트는 CloudWatch 리소스와 상호 작용할 수 있는 권한이 필요합니다. 이렇게 하려면 먼저 CloudWatch 리소스를 사용할 권한이 있는 AWS Identity and Access Management IAM () 역할을 설정한 다음 IAM 역할 및 공유 자격 증명이 활성화된 플릿을 구성해야 합

니다. Amazon은 게임 서버 빌드를 각 EC2 인스턴스에 GameLift 배포할 때 공유 자격 증명 파일을 생성하여 인스턴스에 저장합니다. 인스턴스의 모든 애플리케이션은 공유 자격 증명을 사용할 수 있습니다. Amazon은 인스턴스의 수명 주기 동안 임시 자격 증명을 GameLift 자동으로 새로 고칩니다.

다음 방법을 사용하여 관리형 EC2 플릿을 생성할 때 공유 자격 증명을 활성화할 수 있습니다.

- Amazon GameLift 콘솔 플릿 생성 워크플로에서
- 새 파라미터를 `CreateFleet InstanceRoleCredentialsProvider` 사용하여 Amazon GameLift 서비스 API 작업을 호출하는 경우
- 파라미터를 사용하여 AWS CLI 작업을 `aws gamelift create-fleet` 호출하는 경우 `instance-role-credentials-provider`

자세히 알아보기:

- 플릿의 다른 AWS 리소스와 소통하기, Amazon GameLift 개발자 가이드
- CreateFleet, InstanceRoleCredentialsProvider, 아마존 GameLift API 레퍼런스
- IAM 서비스 역할 설정, Amazon GameLift 개발자 가이드

2023년 9월 28일: 아마존, 언리얼 엔진용 새 독립형 플러그인 GameLift 출시

SDK 버전 업데이트: 언리얼 엔진 버전 1.0.0용 Amazon GameLift 플러그인

Unreal Engine용 Amazon GameLift 플러그인은 클라우드 호스팅용 Amazon에서 게임을 시작하고 실행하는 단계를 간소화하는 도구와 GameLift 워크플로를 제공합니다. GameLift Amazon은 게임 개발자가 세션 기반 멀티플레이어 게임 전용 게임 서버를 관리하고 확장할 수 있는 완전 관리형 서비스입니다. 플러그인은 UE 버전 5.0, 5.1 및 5.2를 지원합니다. 이 기능은 다음과 같습니다.

- Unreal 편집기의 안내식 UI 워크플로는 다음 경로를 따라 진행됩니다.
  - 로컬 워크스테이션을 호스트로 GameLift 사용하여 Amazon과의 게임 통합을 테스트하십시오. 이 워크플로는 로컬 머신용 Amazon GameLift Anywhere 플릿을 설정하고, 게임 서버 및 클라이언트의 인스턴스를 시작하고, Amazon을 통해 게임 세션을 요청하고 GameLift, 새 게임 세션에 대한 연결 정보를 가져오는 데 도움이 됩니다.
  - 통합된 게임 서버를 위한 Amazon EC2 클라우드 호스팅 솔루션을 배포합니다. 이 워크플로는 게임을 클라우드 호스팅용으로 구성하는 데 도움이 되며, 세 가지 배포 시나리오를 제공합니다. 즉, 단일 플릿에 배포, 여러 지역의 스팟 플릿 세트에 배포 또는 매치메이커를 사용하여 플릿 세트에 배포하는 것입니다. FlexMatch 각 배포 시나리오의 솔루션에는 Amazon GameLift 리소스 및 지원 AWS 리소스가 포함됩니다.

- AWS 계정 사용자에게 연결되는 사용자 프로필을 설정하고 기본 AWS 지역을 정의할 수 있습니다. 여러 프로필을 유지 관리하여 여러 AWS 계정, 계정 사용자, 지역에서 작업할 수 있습니다.
- Amazon GameLift 통합 및 배포 프로세스를 간소화하는 데 도움이 되는 특별한 편의 기능은 다음과 같습니다.
  - 각 호스팅 솔루션에는 고유한 플레이어 ID를 제공하는 기본 Amazon Cognito 사용자 풀, 스토리지용 Amazon S3 버킷, Amazon SNS 이벤트 알림 및 AWS Lambda 기능을 비롯한 지원 AWS 리소스가 포함되어 있습니다.
  - Anywhere 워크플로의 경우 플러그인은 명령줄 인수를 사용하여 필요한 서버 파라미터 설정을 자동화합니다.
  - Amazon EC2 워크플로의 경우, 각 배포 솔루션은 Lambda 함수를 사용하여 내장된 클라이언트 백엔드 서비스를 제공합니다. 백엔드 서비스는 게임 클라이언트로부터 요청을 받아 Amazon GameLift 서비스에 전달합니다.
- 기본 블루프린트와 UI 요소가 포함된 스타트업 게임 맵과 테스트 맵 2개를 포함한 통합 테스트용 콘텐츠.
- 자세한 통합 지침 및 샘플 코드가 포함된 플러그인 설명서.

for Anywhere 및 Amazon EC2 플릿을 포함한 모든 배포 시나리오에서는 AWS CloudFormation 템플릿을 사용하여 솔루션을 설명합니다. 플러그인은 게임에 Amazon GameLift 리소스를 배포할 때 이러한 템플릿을 사용합니다. 이러한 템플릿은 Amazon GameLift 플러그인 다운로드에 포함되어 있으며 편집할 수 있습니다. 그대로 사용하거나 게임에 맞게 수정할 수 있습니다.

자세히 알아보기:

- 언리얼 엔진용 Amazon GameLift 플러그인과 게임 통합, 아마존 GameLift 개발자 가이드
- 에서 플러그인을 다운로드하십시오. GitHub
- 아마존 GameLift 호스팅에 대해
- 아마존 GameLift 포럼

2023년 8월 17일: GameLift Amazon은 AWS 그래비톤 프로세서를 사용한 게임 서버 호스팅을 제공합니다.

SDK 버전 업데이트: SDK 1.11.144 AWS

GameLift Amazon에서는 이제 AWS Graviton 프로세서가 장착된 EC2 인스턴스를 사용하여 클라우드에서 게임을 호스팅할 수 있습니다. ARM64 기반 프로세서로 설계된 Graviton 인스턴스는 유사한 x86

기반 인스턴스보다 최대 40% AWS 개선된 EC2를 사용하는 클라우드 워크로드에 대해 최고의 가격 대비 성능을 제공합니다. 최신 Graviton3 프로세서는 이전 버전보다 최대 25% 더 나은 컴퓨팅 성능을 제공합니다.

GameLiftAmazon에서는 이제 AWS Graviton 제품군의 다음과 같은 새 인스턴스 중에서 선택할 수 있습니다.

- Graviton2 기반 인스턴스: c6g, c6gn, r6g, m6g, g5g
- Graviton3 기반 인스턴스스: c7g, r7g, m7g

자세히 알아보기:

- AWS Graviton 프로세서: Graviton 기반 EC2 인스턴스의 이점과 실제 사용에 대해 알아보십시오.
- Graviton 시작하기: Graviton 기반 인스턴스에 대한 개요와 운영 체제, 언어 및 실행 시간에 따라 Graviton 기반 인스턴스에서 애플리케이션이 실행되는 방식에 대한 통찰력을 얻으십시오.

> **Note**
>
> Graviton Arm 인스턴스는 리눅스 OS에 빌드된 아마존 GameLift 서버가 필요합니다. C++ 및 C#에는 Server SDK 5.1.1 이상이 필요합니다. Go에는 Server SDK 5.0 이상이 필요합니다. 이러한 인스턴스는 아마존 리눅스 2023 (AL2023) 또는 아마존 리눅스 2 (AL2) 에서의 모노 설치를 out-of-the-box 지원하지 않습니다.

2023년 7월 27일: 아마존, 유니티 개발에 대한 지원이 추가된 서버 SDK 5.1.0 GameLift 출시

업데이트된 SDK 버전: C++, C#/Unity, Unreal 5.1.0용 Server SDK

Amazon GameLift 서버 SDK의 최신 릴리스는 C++, C#, 언리얼 플러그인에 대한 업데이트와 Unity 게임 엔진과 함께 사용할 수 있는 새로운 플러그인을 제공합니다. 게임 개발자는 Amazon GameLift 서버 SDK를 Amazon에서 호스팅하기 위해 배포하는 게임 서버에 통합합니다. GameLift

최신 서버 SDK 버전에는 여러 고객 요청을 포함하는 다음과 같은 업데이트가 포함되어 있습니다.

- 언어별 SDK 패키지 다운로드 — 업데이트된 Amazon GameLift 다운로드 사이트에는 각 언어에 대한 SDK 패키지가 포함되어 있습니다. 현재 또는 이전 버전을 다운로드할 수 있습니다.

- Unity용 새 C# Server SDK 플러그인 - Unity용 새 Server SDK 패키지에는 Unity 편집기에서 패키지 관리자를 사용하여 설치할 수 있는 빌드된 C# 라이브러리가 포함되어 있습니다(새 Unity 통합 안내서 참조). 이러한 라이브러리에는 를 통한 필수 종속성이 포함되어 있습니다. UnityNuGet 이 플러그인은 Windows 및 Mac OS용 Unity 2020.3 LTS, 2021.3 LTS 및 2022.3 LTS와 함께 사용할 수 있습니다. NET Standard 2.1 및 .NET 4.x가 있는 Unity의 .NET Framework 및 .NET Standard 프로필을 지원합니다.
- C#용 통합 .NET 솔루션 - C#용 Server SDK는 이제 단일 솔루션에서.NET Framework 4.6.2(4.6.1에서 업그레이드됨) 및 .NET 6.0을 지원합니다. .NET Standard 2.1은 Unity 빌드된 라이브러리와 함께 사용할 수 있습니다.
- Server SDK 5.1.0 업데이트
  - [C++, C#, Unreal] 이제 서버 파라미터를 사용하거나 사용하지 않고 `InitSDK()`를 호출할 수 있습니다. Amazon GameLift 관리형 EC2 플릿에서 실행되는 게임 서버는 환경 변수에서 직접 서버 파라미터를 읽습니다. Amazon GameLift Anywhere 플릿의 게임 서버는 서버 매개변수를 `InitSDK()` 사용하여 호출해야 합니다.
  - [C++, C#, Unreal] Server SDK 호출을 통해 오류 메시지를 개선했습니다.
  - [C++ SDK] Server SDK 빌드 시간을 개선하기 위해 `-DRUN_CLANG_FORMAT` 빌드 플래그가 기본적으로 비활성화되어 있습니다. `-DRUN_CLANG_FORMAT=1`을 사용하여 활성화할 수 있습니다.
  - [C++ SDK] 표준 라이브러리(`-DGAMELIFT_USE_STD=0`)없이 라이브러리를 빌드하면 `InitSDK()`는 더 이상 `std::` 데이터 유형을 사용하지 않습니다.
- 확장된 Server SDK 5.x 설명서
  - 모든 데이터 유형에 대한 확장된 적용 범위를 포함한 C++, C#/Unity, 및 Unreal용 Server SDK 참조 가이드가 업데이트되었습니다.
    - C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조
    - C++용 Amazon GameLift Server SDK 5.x 참조
    - Amazon GameLift Unreal Engine Server SDK 5.x 참조
  - Unity 및 Unreal 플러그인을 위한 Server SDK 5 통합 안내서의 새 버전
    - Amazon GameLift를 Unity 프로젝트에 통합
    - GameLift Amazon을 언리얼 엔진 프로젝트에 통합
- 추가 설명서 업데이트
  - 사용 중인 Amazon GameLift 서버 SDK 버전을 기반으로 원격 액세스 절차를 명확히 GetComputeAccess하고 GetInstanceAccessAmazon GameLift 서비스 API 작업에 대한 설명서를 수정했습니다.

- 배치가 “보류 중” 상태일 때 게임 세션 정보가 어떻게 일시적인지 GameSessionPlacement설명하도록 설명을 수정했습니다.

2023년 7월 13일: 아마존, 플릿 하드웨어 지표 GameLift 추가

이제 Amazon GameLift 관리형 EC2 플릿의 하드웨어 성능 지표를 추적할 수 있습니다. 지표에는 CPU 사용률, 네트워크 트래픽 볼륨, 디스크 읽기/쓰기 활동에 대한 EC2 인스턴스 지표가 포함됩니다. Amazon의 GameLift 경우 이러한 지표는 플릿 위치의 모든 활성 인스턴스를 설명합니다. 의 Amazon CloudWatch 대시보드를 사용하여 이러한 플릿 하드웨어 지표를 볼 수 AWS Management Console있습니다. Amazon GameLift 콘솔의 플릿 세부 정보에서도 이를 확인할 수 있습니다.

자세히 알아보기:

- Amazon CloudWatch를 사용한 Amazon GameLift 모니터링(플릿용 지표), Amazon GameLift 개발자 가이드

2023년 6월 29일: 아마존, 아마존 리눅스 2023에 대한 지원 GameLift 시작

SDK 버전 업데이트: AWS SDK 1.11.111

Amazon GameLift 고객은 이제 Amazon Linux 2023 운영 체제를 사용하여 게임 서버를 호스팅할 수 있습니다. AL2023은 보안을 포함하여 AL2에 비해 몇 가지 향상된 기능을 제공합니다. 이 운영 체제는 중국 지역을 제외한 모든 AWS 리전 지역에서 사용할 수 있습니다.

고객은 최신 Linux 운영 체제를 사용할 수 있으며 2023년 12월 Amazon Linux(AL1)에 대한 지원이 종료되면 중요한 보안 업데이트를 계속 받을 수 있습니다. Amazon Linux 2에 대한 지원은 2025년까지 계속됩니다.

자세히 알아보기:

- 아마존 GameLift 리눅스 서버 FAQ
- Amazon Linux 2 및 Amazon Linux 2023 비교
- 아마존 GameLift API 참조 링크:
  - AWS SDK 액션 `CreateBuild`
  - CLI 명령 `upload-build`
  - CLI 명령 `create-build`

2023년 5월 25일: Amazon GameLift FleetiQ는 드레인 인스턴스의 게임 세션 배치를 제외하는 필터를 추가했습니다.

SDK 버전 업데이트: SDK 1.11.87 AWS

Amazon GameLift FleetiQ를 게임 호스팅에 사용하는 경우, 이제 현재 드레이닝 중인 인스턴스에 게임 세션이 배치되는 것을 방지할 수 있습니다. 드레이닝 인스턴스는 종료 플래그가 지정되지만 사용 가능한 다른 호스팅 리소스가 없는 경우 새 게임 세션을 호스팅하도록 선택할 수 있습니다. 이 새로운 기능을 사용하면 드레이닝 인스턴스 사용을 완전히 배제할 수 있습니다.

사용 가능한 게임 서버를 찾기 위해 `ClaimGameServer`를 호출할 때 이 기능을 사용합니다. 새 `FilterOption` 파라미터를 추가하고 허용된 인스턴스 상태를 ACTIVE로만 설정합니다. 이에 대해 Amazon GameLift FleetiQ는 사용 가능한 게임 서버를 검색하고 요청할 때 활성 인스턴스만 살펴봅니다.

자세히 알아보기:

- ClaimGameServer아마존 GameLift API 레퍼런스에서
- 아마존 GameLift FleetiQ 개발자 안내서에서 FleetiQ가 작동하는 방식

2023년 5월 16일: Amazon은 플릿에 대한 비용 할당 태깅을 GameLift 지원합니다.

Amazon GameLift 고객은 이제 AWS Billing 비용 할당 태그를 사용하여 게임 호스팅 비용을 정리할 수 있습니다. 개별 Amazon GameLift EC2 플릿 리소스에 비용 할당 태그를 할당하여 플릿이 전체 호스팅 비용에 어떻게 기여하는지 추적할 수 있습니다.

자세히 알아보기:

- 게임 호스팅 비용 관리
- AWS Billing 사용 설명서의 AWS 비용 할당 태그 사용

2023년 4월 20일: 아마존 GameLift , 윈도우 서버 2016 지원 시작

SDK 버전 업데이트: AWS SDK 1.11.63

아마존 GameLift 고객은 이제 Windows Server 2016 운영 체제를 사용하여 게임 서버를 호스팅할 수 있습니다. 이 운영 체제는 모두 사용할 수 AWS 리전있습니다. Microsoft가 2023년 10월에 Windows Server 2012에 대한 지원을 종료함에 따라 고객은 최신 Windows 운영 체제를 사용하고 중요한 보안 업데이트를 계속 받을 수 있습니다.

오늘부터 Windows 런타임 환경이 필요한 신규 고객은 호스팅용 새 게임 서버 빌드를 만들 때 Windows Server 2016을 지정해야 합니다. 기존 고객은 Windows Server 2012를 사용하여 새로운 빌드와 플릿을 계속 만들 수 있지만, Microsoft의 지원 종료일인 2023년 10월 10일 이전에 Windows Server 2016으로 마이그레이션을 완료해야 합니다.

다음은 이 업데이트의 서비스 변경 사항입니다.

- Amazon GameLift SDK 또는 CLI 명령을 사용하여 게임 서버 빌드를 생성할 때 이제 운영 체제를 명시적으로 설정해야 합니다. 더 이상 기본값이 없습니다. Windows Server 2016에 게임 서버를 배포하려면 `WINDOWS_2016` 값을 사용합니다.
- Amazon GameLift 콘솔을 사용하여 게임 서버 빌드를 생성할 때는 사용 가능한 값에서 운영 체제를 선택해야 합니다. 활성 Windows Server 2012 플릿을 사용하는 기존 고객인 경우 `WINDOWS_2012` 또는 `WINDOWS_2016` 둘 중 하나를 선택할 수 있습니다.

자세히 알아보기:

- 아마존 GameLift API 참조 링크:
  - CLI 명령 `upload-build`
  - CLI 명령 `create-build`
  - AWS SDK 액션 `CreateBuild`
- 윈도우 2012용 아마존 GameLift FAQ

2023년 4월 13일: 아마존, 언리얼용 서버 SDK 5.x GameLift 출시

업데이트된 SDK 버전: Unreal용 Server SDK 5.0.0

언리얼 엔진용 Amazon GameLift 경량 플러그인 최신 버전은 이제 Amazon GameLift 서버 SDK 5.x를 기반으로 합니다. 언리얼 엔진 환경을 Amazon과 통합하려면 다음 GameLift 링크를 참조하십시오.

자세히 알아보기:

- GameLift Amazon을 언리얼 엔진 프로젝트에 통합
- Amazon GameLift를 게임 서버에 추가
- C++용 Amazon GameLift Server SDK 5.x 참조

2023년 3월 14일: 아마존 GameLift , 새로운 콘솔 경험 출시

새 Amazon GameLift 콘솔에는 다음과 같은 개선 사항이 포함되어 있습니다.

- 향상된 탐색 — 새 탐색 창을 통해 Amazon GameLift 리소스 간 탐색이 용이해집니다.
- Amazon GameLift 랜딩 페이지 — 새 랜딩 페이지는 유용한 설명서로 연결되는 링크를 제공하고, Amazon에 대한 개괄적인 개요를 표시하고 GameLift, 설명서, 자주 묻는 질문 등에 대한 링크를 통해 지원을 제공합니다. AWS re:Post
- Amazon CloudWatch 지표 개선 — Amazon GameLift 메트릭은 이제 Amazon GameLift 콘솔과 CloudWatch 대시보드에서 모두 사용할 수 있습니다. 이 업데이트에는 성능, 사용률, 플레이어 세션에 대한 새로운 지표도 포함됩니다.

자세히 알아보기:

- 콘솔에서 게임 데이터 보기
- Amazon GameLift 호스팅 리소스 관리
- 매치메이커 구축 FlexMatch

2023년 2월 14일: Amazon은 GameLift 이제 Amazon SNS 주제에 대한 서버 측 암호화를 지원합니다.

SNS 주제의 서버 측 암호화(SSE)는 민감한 저장 데이터를 암호화합니다. SSE는 AWS Key Management Service (AWS KMS) 키를 사용하여 SNS 주제의 콘텐츠를 보호합니다.

자세히 알아보기:

- 게임 세션 배치의 이벤트 알림 설정
- FlexMatch매치메이킹 이벤트
- 저장된 데이터 암호화

2023년 2월 9일: 아마존 GameLift 서버 SDK는 C #10 기반으로.NET 6을 지원합니다.

업데이트된 SDK 버전: .NET 6용 Server SDK 5.0.0. SDK 업데이트는 필요하지 않습니다.

Unity 실시간 개발 플랫폼을 사용하는 경우 Amazon GameLift 서버 SDK 5.0.0을.NET 4.6과 함께 계속 사용하십시오. Unity는 .NET 6을 지원하지 않습니다.

자세히 알아보기:

- Amazon에서 최신 버전의 Amazon GameLift 서버 SDK를 다운로드하여 시작하십시오 GameLift .
- C# 및 Unity용 Amazon GameLift Server SDK 5.x 참조

2023년 1월 31일: 아마존 GameLift 서버 SDK는 Go 언어를 지원합니다.

업데이트된 SDK 버전: Go용 Server SDK 5.0.0

자세히 알아보기:

- Amazon에서 최신 버전의 Amazon GameLift 서버 SDK를 다운로드하여 시작하십시오 GameLift .
- Go용 Amazon GameLift Server SDK 참조

2022년 12월 1일: 아마존 GameLift Anywhere 및 아마존 GameLift 서버 SDK 5.0 GameLift 출시

SDK 버전 업데이트: AWS SDK 1.10.21, C++ 및 C #용 서버 SDK 5.0.0

GameLift AnywhereAmazon은 사용자의 게임 서버 리소스를 사용하여 Amazon GameLift 게임 서버를 호스팅합니다. GameLift AnywhereAmazon을 사용하여 자체 컴퓨팅 리소스를 Amazon GameLift 관리형 EC2 컴퓨팅과 통합하여 게임 서버를 여러 컴퓨팅 유형에 분산할 수 있습니다. 또한 GameLift Anywhere Amazon을 사용하면 반복할 때마다 Amazon에 빌드를 업로드하지 않고도 게임 서버를 반복적으로 테스트할 수 있습니다. GameLift

주요 내용:

- 새로운 Amazon GameLift Anywhere 플릿 및 컴퓨팅 유형
- Amazon GameLift Anywhere 컴퓨팅 리소스 등록
- 테스트 반복 주기 개선

Amazon GameLift Server SDK 5.0.0은 기존 서버 SDK와 새로운 리소스 유형인 컴퓨팅을 개선했습니다. Server SDK 5.0.0은 Amazon GameLift Anywhere 및 게임 서버 호스팅을 위한 자체 컴퓨팅 리소스 사용을 지원합니다.

자세히 알아보기:

- Amazon GameLift Server SDK 참조

- 플릿 위치
- Amazon GameLift 컴퓨팅 리소스 선택
- 아마존 GameLift Anywhere 플릿 생성

2022년 8월 25일: 아마존, Local Zones에 대한 지원 GameLift 시작

SDK 버전 업데이트: AWS SDK 1.9.333

GameLift Amazon은 이제 미국의 8개 Local Zone에서 사용할 수 있으므로 플레이어들과 더 가까운 곳에 플릿을 배치할 수 있습니다. 플릿에 Local Zones를 추가하여 모든 관리형 Amazon GameLift 기능을 Local Zones와 함께 사용할 수 있습니다.

Local Zone은 대규모 인구, 산업 및 정보 기술 (IT) 센터 근처의 클라우드 에지까지 AWS 리소스와 서비스를 확장합니다. 즉, 한 자릿수 밀리초 지연 시간이 필요한 애플리케이션을 최종 사용자나 온프레미스 데이터 센터 가까이에 배포할 수 있게 되었습니다.

자세히 알아보기:

- 로컬 영역
- 플릿 위치
- Amazon GameLift 관리형 플릿 생성

2022년 6월 28일: 아마존 GameLift , 새로운 옵트인 콘솔 환경 출시

새 Amazon GameLift 콘솔에는 다음과 같은 개선 사항이 포함되어 있습니다.

- 향상된 탐색 — 새 탐색 창을 통해 Amazon GameLift 리소스 간 탐색이 용이해집니다.
- Amazon GameLift 랜딩 페이지 — 새 랜딩 페이지는 유용한 설명서로 연결되는 링크를 제공하고, Amazon에 대한 개괄적인 개요를 표시하고 GameLift, 설명서, 자주 묻는 질문 등에 대한 링크를 통해 지원을 제공합니다. AWS re:Post
- Amazon CloudWatch 지표 개선 — Amazon GameLift 메트릭은 이제 Amazon GameLift 콘솔과 CloudWatch 대시보드에서 모두 사용할 수 있습니다. 이 업데이트에는 성능, 사용률, 플레이어 세션에 대한 새로운 지표도 포함됩니다.

자세히 알아보기:

- 콘솔에서 게임 데이터 보기

- Amazon GameLift 호스팅 리소스 관리
- 매치메이커 구축 FlexMatch

2022년 2월 15일: 복합 규칙 FlexMatch 추가 및 추가 개선 사항

FlexMatch 이제 사용자는 다음 기능을 이용할 수 있습니다.

- 복합 규칙 - 플레이어 40명 이하의 매치에 대한 복합 매치메이킹 규칙에 대한 지원이 추가되었습니다. 이제 논리적 문을 사용하여 매치를 구성하도록 복합 규칙을 생성할 수 있습니다. 매치를 구성하는 데 규칙 세트에 복합 규칙이 없는 경우 규칙 세트의 모든 규칙이 True여야 합니다. 복합 규칙의 경우, 다음과 같은 논리 연산자를 사용하여 적용할 규칙을 선택할 수 있습니다. `and`, `or`, `not`, `xor`.
- 유연한 팀 선택 - 사용 가능한 모든 팀의 하위 집합을 선택할 수 있도록 매치메이킹 속성 표현식이 업데이트되었습니다.
- 더 긴 문자열 목록 - 플레이어 속성 값의 문자열 목록에서 최대 문자열 수를 10개에서 100개로 늘렸습니다.

자세히 알아보기:

- Amazon GameLift FlexMatch 개발자 가이드:
  - FlexMatch 규칙 유형
  - FlexMatch 속성 표현식
- AttributeValue: SL

2021년 10월 28일: 아마존은 아시아 태평양 (오사카) 지역의 멀티 리전 플릿에 대한 지원을 GameLift 추가하고, Amazon GameLift FleetiQ는 Graviton2 프로세서에 대한 지원을 추가합니다. AWS

SDK 버전 업데이트 AWS : SDK 1.9.133

GameLift Amazon은 이제 아시아 태평양 (오사카) 지역에서 사용할 수 있습니다. 이제 게임 개발자는 GameLift 다중 지역 플릿을 사용하여 오사카에 인스턴스를 배포할 수 있습니다.

이제 ARM 기반 프로세서 아키텍처를 기반으로 하는 Graviton2 호스팅 게임 서버를 사용하여 동급 인텔 기반 컴퓨팅 옵션에 비해 저렴한 비용으로 향상된 성능을 달성할 수 있습니다.

주요 내용:

- GameLift Amazon은 이제 아시아 태평양 (오사카) 지역에서 사용할 수 있습니다.
- 이제 Amazon GameLift FleetiQ 게임 서버 그룹을 구성하여 Graviton2 인스턴스 제품군 c6g, m6g 및 r6g를 관리할 수 있습니다.

자세히 알아보기:

- Amazon GameLift 멀티 리전 플릿
- CreateGameServerGroup
- AWS 그래비톤 프로세서

2021년 9월 20일: 아마존, 유니티용 플러그인 GameLift 출시

Unity 버전 1.0.0용 Amazon GameLift 플러그인에는 Amazon GameLift 리소스에 쉽게 액세스하고 Amazon을 Unity 게임에 GameLift 통합할 수 있는 라이브러리와 네이티브 UI가 포함되어 있습니다. Unity용 Amazon GameLift 플러그인을 사용하여 Amazon GameLift API에 액세스하고 일반적인 게임 시나리오에 맞는 AWS CloudFormation 템플릿을 배포할 수 있습니다. 플러그인에는 샘플 시나리오와 함께 작동하는 샘플 게임도 포함되어 있습니다. Amazon GameLift Local을 사용하여 게임 클라이언트와 게임 서버 간에 전달되는 메시지를 보고 일반적인 게임이 GameLift Amazon과 상호 작용하는 방식을 알아볼 수 있습니다.

Unity용 플러그인은 Unity 2019.4 LTS 및 2020.3 LTS를 지원합니다.

주요 내용:

- 다양한 시나리오로 샘플 게임을 빌드, 실행, 수정하거나 직접 만들 수 있습니다.
- 인증 전용, 단일 지역 플릿, 대기열 및 사용자 지정 매치메이커가 있는 다중 지역 플릿, 대기열 및 사용자 지정 매치메이커가 있는 스팟 플릿 등을 포함한 일반적인 게임 시나리오에 대한 샘플 AWS CloudFormation 시나리오를 배포하십시오. FlexMatch

자세히 알아보기:

- Unity용 Amazon GameLift 플러그인과 게임 통합

2021년 6월 30일: 배치/디스턴스 FlexMatch 규칙 추가

batchDistance 규칙 유형을 사용하여 문자열 또는 숫자 속성을 지정할 수 있으므로 각 문자열에 많은 이점을 제공할 수 있습니다.

주요 내용:

- 대규모 매치(플레이어 40명 이상)의 경우, 이제 스킬만으로 플레이어를 균등하게 밸런싱하는 대신, 스킬, 모드, 맵에 따라 동일한 밸런스를 맞출 수 있습니다. 매치에 참여하는 모든 사용자가 스킬 밴드에 속해 있는지 확인하고, 리그 또는 플레이 스타일과 같은 여러 숫자 속성을 밴드로 묶고, 맵이나 게임 모드와 같은 문자열 속성에 따라 그룹을 구성합니다. 시간이 지남에 따라 확장을 생성할 수도 있습니다. 예를 들어 매치에 참여하는 데 플레이어의 대기 시간이 더 길수록 더 훌륭한 스킬 레벨 범위를 허용하는 확장판을 만들 수 있습니다.

  플레이어 수가 40명 미만인 매치의 경우 새롭게 단순화된 규칙 표현식을 사용할 수 있습니다.

2021년 6월 3일: 아마존 GameLift 실시간 클라이언트 SDK 및 서버 SDK 업데이트

업데이트된 SDK 버전: Realtime Client SDK 1.2.0, Unreal용 Server SDK 3.4.0

이번 최신 SDK 업데이트를 통해, 이제 RTS Client SDK를 사용하는 모바일 애플리케이션에 IL2CPP를 통합하고 프레임워크의 모범 사례를 따를 수 있습니다. 이제 언리얼 버전 4.26용 Amazon GameLift Server SDK를 빌드할 수도 있습니다. 이 업데이트에는 C++ 및 C# 버전의 Amazon Server SDK, Amazon GameLift Local, 언리얼 엔진 플러그인 등 Windows 또는 Linux 게임 GameLift 서버와 통합되는 컴포넌트가 포함되어 있습니다.

주요 내용:

- 최신 모바일 디바이스용 RTS 클라이언트를 구축할 수 있도록 RTS Client SDK에 IL2CPP 지원 및 네이티브 라이브러리를 프레임워크로 빌드하기 위한 지원이 추가되었습니다.
- DescribePlayerSessions()을 사용하여 단일 플레이어 세션 정보, 게임 세션에 있는 모든 플레이어 세션 정보 또는 단일 플레이어 ID와 관련된 모든 플레이어 세션 정보를 가져올 수 있습니다.
- GetInstanceCertificate()을 사용하여 플릿 및 플릿의 인스턴스와 연결된 PEM 인코딩된 TLS 인증서의 파일 위치를 검색할 수 있습니다.
- Unreal 버전 4.26에 대한 Server SDK 지원을 만들었습니다.
- 기존 C# SDK 버전 4.0.2는 Unity 2020.3과 호환되는 것으로 검증되었습니다. SDK 업데이트는 필요하지 않습니다.

자세히 알아보기:

- Amazon GameLift 개발자 가이드:
  - DescribePlayerSessions()
  - GetInstanceCertificate()

2021년 3월 23일: Amazon은 게임 세션 배치에 알림을 GameLift 추가합니다.

SDK 버전 업데이트: AWS SDK 1.8.168

이제 이벤트를 사용하여 게임 세션 대기열의 게임 세션 배치 활동을 모니터링할 수 있습니다. Amazon Simple Notification Service (Amazon SNS) 주제를 생성하여 이벤트 알림을 게시하거나 이벤트를 사용하여 이벤트 추적을 CloudWatch 설정합니다.

주요 내용:

- 각 대기열에 대해 모든 이벤트 메시지에 포함되도록 사용자 지정 텍스트 문자열을 설정할 수 있습니다.
- Amazon SNS 주제를 사용하는 경우 게시를 특정 대기열로 제한하는 추가 액세스 조건을 설정할 수 있습니다.

자세히 알아보기:

- Amazon GameLift 개발자 가이드:
  - 게임 세션 배치의 이벤트 알림 설정 (new)
  - 게임 세션 배치 이벤트 (new)
- API 참조(AWS SDK)
  - 새 게임 세션 대기열 매개 변수 `NotificationTarget` 및`CustomEventData`: GameSessionQueue, CreateGameSessionQueue, UpdateGameSessionQueue
- 아마존 GameLift 포럼

2021년 3월 16일: 아마존, 다중 지역 플릿, 6개 신규 지역 GameLift 추가

SDK 버전 업데이트: SDK 1.8.163 AWS

Amazon GameLift 관리형 호스팅은 이제 21개 AWS 지역에서 사용할 수 있습니다. 새 리전은 케이프타운(af-south-1), 바레인(me-south-1), 홍콩(ap-east-1), 밀라노(eu-south-1), 파리(eu-west-3), 스톡홀름(eu-north-1)입니다.

새로운 Amazon GameLift 다중 위치 플릿 기능을 사용하면 이제 20개 Amazon GameLift 지원 지역 중 일부 또는 전체에 게임 서버를 호스팅하도록 단일 플릿을 설정할 수 있습니다 (베이징 지역 제외). 이 기능은 전 세계적으로 Amazon GameLift 호스팅 리소스를 설정하고 유지 관리하는 데 필요한 작업을 크게 줄이는 것을 목표로 합니다. (버지니아 북부), (오레곤), us-east-1 (프랑크푸르트), (아일랜드), us-west-2 (시드니), eu-central-1 (도쿄), eu-west-1 (서울) AWS 지역에서 다중 위치 플릿을 생성할 수 있습니다. ap-southeast-2 ap-northeast-1 ap-northeast-2 다른 모든 리전에서는 필요에 따라 단일 위치 플릿을 계속 설정할 수 있습니다. 이번 릴리스 이전에 생성된 모든 플릿은 단일 위치 플릿입니다. 다중 위치 플릿을 사용해도 호스팅 비용에는 영향을 미치지 않습니다. Amazon GameLift 요금은 사용하는 인스턴스의 유형, 위치 및 볼륨을 기준으로 책정됩니다. (자세한 내용은 Amazon GameLift 요금을 참조하십시오.) AWS CloudFormation 다중 위치 플릿에 대한 지원이 곧 제공될 예정입니다.

> **Note**
>
> 중국 리전에서 다중 위치 플릿을 사용할 수 없습니다. 중국 지역에 있는 Amazon GameLift 리소스는 다른 Amazon GameLift 지역의 리소스와 상호 작용하거나 해당 리소스에서 사용할 수 없습니다.

주요 내용:

- 다중 위치 플릿의 경우 원격 위치 목록을 명시적으로 추가합니다. Amazon은 빌드 및 런타임 구성을 포함하여 동일한 유형 및 구성의 인스턴스를 플릿의 홈 지역 및 추가된 모든 위치에 GameLift 배포합니다.
- 각 위치의 용량 설정 및 크기 조정을 개별적으로 조정합니다. 자동 크기 조정 정책은 전체 플릿에 적용되지만 위치별로 켜거나 끌 수 있습니다.
- 특정 플릿 위치에서 새 게임 세션을 시작합니다. 게임 세션 대기열이나 매치메이킹을 사용하여 게임 세션을 배치할 때 이제 위치, 호스팅 비용, 플레이어 지연 시간을 기준으로 새 게임 세션이 시작되는 위치의 우선 순위를 지정할 수 있습니다.
- Amazon GameLift 콘솔에서 호스팅 메트릭을 가져오세요. 플릿 내 모든 위치에 대해 집계되거나 플릿 위치별로 분류된 호스팅 메트릭을 확인할 수 있습니다.

자세히 알아보기:

- Amazon Game Tech 블로그
- API 참조(AWS SDK)
  - 새로운 플릿 위치 운영: CreateFleetLocations, DescribeFleetLocationAttributes, DescribeFleetLocationCapacity, DescribeFleetLocationUtilizationDeleteFleetLocations
  - 새로운 다중 위치 지원을 통한 플릿 운영 업데이트: CreateFleet,, DescribeEC2 UpdateFleetCapacityInstanceLimits,, DescribeInstancesStopFleetActionsStartFleetActions
  - 새로운 우선 순위 및 필터링 기능으로 게임 세션 배치 작업 업데이트:,, CreateGameSessionQueueDescribeGameSessionQueuesUpdateGameSessionQueue
  - 새로운 위치 지원과 함께 게임 세션 생성 작업 업데이트: CreateGameSessionDescribeGameSessions, DescribeGameSessionDetails, SearchGameSessions
- Amazon GameLift 개발자 가이드:
  - 아마존 GameLift 호스팅 위치 (업데이트됨)
  - Amazon GameLift 플릿 설계 가이드 (new)

    Amazon GameLift 호스팅 용량 확장 (업데이트됨)
  - 게임 세션 대기열 설계 (new)
  - 플릿 세부 정보 보기 (업데이트됨)
- 아마존 GameLift 포럼

2021년 2월 9일: 아마존, 독립형 AMD 인스턴스 지원 GameLift 확대 FlexMatch

SDK 버전 업데이트: SDK 1.8.139 AWS

다음은 이 릴리스의 업데이트 사항입니다.

- 이제 Amazon GameLift FleetiQ 게임 서버 그룹을 구성하여 AMD 인스턴스 제품군 C5a, M5a 및 R5a를 관리할 수 있습니다. 에 대해 나열된 대로 지원되는 Amazon EC2 인스턴스 유형에는 이제 다음이 포함됩니다. GameServerGroup InstanceDefinition
  - c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
  - m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge, m5a.24xlarge
  - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

참고: FleetiQ용 AMD 인스턴스는 현재 중국 (베이징) 지역에서 사용할 수 없습니다. AWS 중국의 기능 가용성 및 구현 차이를 참조하세요.

- Amazon GameLift 관리형 게임 호스팅은 이제 Sinnet에서 운영하는 중국 (베이징) 지역의 AMD 인스턴스를 지원합니다. 새 AMD 인스턴스 패밀리에는 M5a 및 R5a가 포함됩니다. 플릿 InstanceType목록에 따라 지원되는 EC2 인스턴스 유형에는 이제 다음이 포함됩니다.
  - 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
- Amazon은 이제 Sinnet에서 운영하는 중국 (베이징) 지역에서 독립형 매치메이킹 솔루션으로 사용할 GameLift FlexMatch 수 있습니다. 고객은 베이징 지역에서 FlexMatch 매치메이커를 생성하고 파라미터를 독립형으로 구성할 수 있습니다. FlexMatchMode Amazon GameLift 관리형 호스팅 또는 Amazon 외 GameLift 호스팅 솔루션을 사용하는 방법에 대한 FlexMatch 자세한 내용은 Amazon GameLift FlexMatch 개발자 안내서를 참조하십시오.
- Amazon에 대한 이벤트 알림을 설정할 때 이제 Amazon GameLift FlexMatch SNS FIFO 주제를 알림 대상으로 지정할 수 있습니다. 자세한 내용은 다음을 참조하세요.
  - MatchmakingConfiguration NotificationTarget, 아마존 GameLift API 레퍼런스
  - FlexMatch 이벤트 알림 설정, Amazon GameLift FlexMatch 개발자 가이드
  - Amazon SNS FIFO 소개 — F irst-in-first-out 게시/구독 메시징, 뉴스 블로그AWS

2020년 12월 22일: 아마존 GameLift 서버 SDK는 언리얼 엔진 4.25와 유니티 2020을 지원합니다.

SDK 버전 업데이트: 아마존 GameLift 서버 SDK 4.0.2, 언리얼 플러그인 버전 3.3.3

Amazon GameLift Server SDK의 최신 버전에는 다음과 같은 구성 요소가 포함되어 있습니다.

- 업데이트된 Unreal 플러그인은 Unreal Engine 4.25와의 호환성을 위해 업데이트되었습니다. API는 변경되지 않았습니다.
- 기존 C# SDK 버전 4.0.2는 Unity 2020과 호환되는 것으로 검증되었습니다. SDK 업데이트는 필요하지 않습니다.

Amazon에서 최신 버전의 Amazon GameLift Server SDK를 다운로드하여 GameLift 시작하십시오.

2020년 11월 24일: GameLift FlexMatch 이제 어디서든 호스팅되는 게임을 Amazon에서 이용할 수 있습니다.

SDK 버전 업데이트: AWS SDK 1.8.95

GameLift FlexMatch Amazon은 멀티플레이어 게임을 위한 맞춤형 매치메이킹 서비스입니다. 처음에는 Amazon GameLift 관리형 호스팅 사용자를 위해 설계되었지만 이제는 독점적인 온프레미스 컴퓨팅 및 클라우드 컴퓨팅 프리미티브 유형을 비롯한 peer-to-peer 다른 호스팅 시스템을 사용하는 게임에 통합할 FlexMatch 수 있습니다. Amazon EC2에서 게임을 호스팅하기 위해 Amazon GameLift FleetiQ를 사용하는 게임은 이제 매치메이킹을 사용하여 구현할 수 있습니다. FlexMatch

FlexMatch 강력한 매치메이킹 알고리즘과 규칙 언어를 제공하여 플레이어가 주요 플레이어 특성과 보고된 지연 시간을 기반으로 매칭되도록 매치메이킹 프로세스를 사용자 지정할 수 있는 폭넓은 권한을 제공합니다. 또한 플레이어 파티, 플레이어 수락, 매치 보충과 같은 기능을 지원하는 매치메이킹 요청 워크플로를 FlexMatch 제공합니다. Amazon GameLift 관리 호스팅 또는 실시간 FlexMatch 서버와 함께 사용하는 경우 매치메이커는 자동으로 Amazon을 사용하여 호스팅 리소스를 찾고 새로 구성된 매치를 위한 새 게임 세션을 시작합니다. GameLift 독립형 FlexMatch 서비스로 사용하는 경우 매치메이커는 경기 결과를 게임에 전달하고, 그러면 호스팅 솔루션을 사용하여 새 게임 세션을 시작할 수 있습니다.

에 대한 API 작업은 아마존 GameLift 서비스 API의 FlexMatch 일부이며, AWS SDK와 AWS Command Line Interface (AWS CLI) 에 포함되어 있습니다. 이번 릴리스에는 독립형 매치메이킹을 지원하는 다음과 같은 업데이트가 포함되어 있습니다.

- API 리소스 `MatchmakingConfiguration`에는 다음 변경 사항이 있습니다.
  - 새 속성은 매치메이커가 Amazon GameLift 관리형 호스팅과 함께 사용되는지 아니면 독립형 매치메이킹으로 사용되는지를 `FlexMatchMode` 나타냅니다.
  - `FlexMatchMode`가 독립형으로 설정되는 경우 `GameSessionQueueArns` 속성은 필요하지 않습니다.
  - 이러한 속성은 다음과 같은 독립형 매치메이킹에는 사용되지 않습니다. `AdditionalPlayerCount`, `BackfillMode`, `GameProperties`, `GameSessionData`.
- 독립형 매치메이킹에서는 자동 채우기 기능을 사용할 수 없습니다.

2020년 11월 24일: 이제 아마존에서 AMD 인스턴스를 사용할 수 있습니다 GameLift

SDK 버전 업데이트: AWS SDK 1.8.95

Amazon에서 지원하는 Amazon EC2 인스턴스 유형 목록에는 GameLift 이제 C5a, M5a, R5a의 세 가지 새로운 인스턴스 패밀리가 포함됩니다. 이러한 패밀리는 최대 3.3의 주파수에서 실행되는 AMD EPYC 프로세서를 기반으로 하는 AMD 컴퓨팅 최적화 인스턴스로 구성됩니다. 사용합니다. AMD 인스턴스는 x86과 호환되므로 현재 Amazon에서 실행 중인 게임을 변경 없이 AMD 인스턴스 유형에 GameLift 배포할 수 있습니다. 새 인스턴스를 사용할 수 있는 AWS 지역은 미국 동부 (버지니아 북부 및 오하이오), 미국 서부 (오레곤 및 캘리포니아 북부), 캐나다 중부 (몬트리올), 남아메리카 (상파울루), EU 중부 (프랑크푸르트), EU 서부 (런던 및 아일랜드), 아시아 태평양 남부 (뭄바이), 아시아 태평양 북동부 (서울 및 시드니) 입니다.

새 AMD 인스턴스에는 다음이 포함됩니다.

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
- m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge, m5a.24xlarge
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge, r5a.24xlarge

자세히 알아보기:

- Amazon Game Tech 블로그
- 아마존 GameLift 인스턴스 요금
- Amazon EC2 인스턴스 기능 AMD EPYC 프로세서
- 아마존 GameLift 포럼

2020년 11월 11일: 아마존 GameLift 서버 SDK 버전 업데이트

SDK 버전 업데이트: 아마존 GameLift 서버 SDK 4.0.2

새 서버 SDK 버전 4.0.2는 API 작업 `StartMatchBackfill()`과 관련된 알려진 문제를 수정합니다. 이제 이 작업은 매치 채우기 요청에 대한 올바른 응답을 반환합니다.

이 문제는 매치 채우기 프로세스에 영향을 주지 않았으며 이 기능의 작동 방식에는 변화가 없습니다. 이 문제는 매치 채우기 요청의 로그 메시지 및 오류 처리에 영향을 미쳤을 수 있습니다.

Amazon에서 최신 버전의 Amazon GameLift Server SDK를 다운로드하여 GameLift 시작하십시오.

2020년 11월 5일: 새로운 FlexMatch 알고리즘 사용자 지정

FlexMatch 이제 사용자는 매치메이킹 프로세스에서 다음과 같은 기본 동작을 조정할 수 있습니다. 이러한 사용자 지정은 매치메이킹 규칙 세트에서 설정됩니다. Amazon GameLift SDK에는 변경 사항이 없습니다.

- 채우기 티켓의 우선 순위 지정: 적합한 매치를 검색할 때 매치 채우기 티켓의 우선 순위를 높이거나 낮출 수 있습니다. 자동 채우기 기능이 활성화된 경우 채우기 티켓의 우선 순위를 정하는 것이 유용합니다. 알고리즘 속성 `backfillPriority`를 사용합니다.
- 사전 정렬을 통한 매치 일관성 및 효율성의 최적화: 평가를 위해 티켓을 일괄 처리하기 전에 티켓 풀을 사전 정렬하도록 매치메이커를 구성합니다. 주요 플레이어 속성에 따라 티켓을 사전 정렬하면 해당 속성이 더 비슷한 플레이어가 매치에 나오는 경향이 있습니다. 매치 규칙에 사용되는 것과 동일한 속성을 기준으로 사전 정렬하여 평가 프로세스의 효율성을 높일 수도 있습니다. `strategy` 속성을 “Sorted”로 설정한 알고리즘 속성 `sortByAttributes`를 사용합니다.
- 확장 대기 시간이 트리거되는 방식 조정: 매치가 완료되지 않은 경우 기간이 가장 최근(기본)인 티켓을 기준으로 확장을 트리거할지, 기간이 가장 오래된 티켓을 기준으로 확장을 트리거할지 선택합니다. 가장 오래된 티켓으로 트리거하면 매치를 더 빨리 완료하는 경향이 있고, 가장 최근 티켓으로 트리거하면 매치의 질이 높아집니다. 알고리즘 속성 `expansionAgeSelection`을 사용합니다.

2020년 9월 17일: 아마존, 서버 SDK GameLift 업데이트

SDK 버전 업데이트: 아마존 GameLift 서버 SDK 4.0.1

새 Server SDK에는 다음 업데이트를 포함합니다.

- C# API 버전 4.0.1
  - API 작업 TerminateGameSession()은 더 이상 지원되지 않습니다. ProcessEnding()으로 호출을 교체하여 게임 세션과 서버 프로세스를 모두 종료합니다.
  - GetInstanceCertificate() 작업과 관련된 알려진 문제가 수정되었습니다.
  - GetTerminationTime()이제 이 작업은 데이터 유형의 값을 반환합니다. AwsDateTimeOutcome
- C++ API 버전 3.4.1
  - 작업 TerminateGameSession()은 더 이상 지원되지 않습니다. ProcessEnding()으로 호출을 교체하여 게임 세션과 서버 프로세스를 모두 종료합니다.
- Unreal Engine 플러그인 버전 3.3.2
  - 작업 TerminateGameSession()은 더 이상 지원되지 않습니다. ProcessEnding()으로 호출을 교체하여 게임 세션과 서버 프로세스를 모두 종료합니다.

- 매치 채우기를 지원하도록 콜백 작업 `OnUpdateGameSession`이 FProcessParameters에 추가되었습니다.

Amazon에서 최신 버전의 Amazon GameLift Server SDK를 다운로드하여 GameLift 시작하십시오.

2020년 8월 27일: 아마존 EC2를 GameLift 사용한 게임 호스팅용 Amazon FleetiQ (일반 공급)

SDK 버전 업데이트: SDK 1.8.36AWS

Amazon GameLift EC2에서의 저렴한 클라우드 기반 게임 호스팅을 위한 Amazon FleetiQ 솔루션이 이제 정식 출시되었습니다. Amazon GameLift FleetiQ는 개발자가 게임 호스팅에 대한 실행 가능성을 최적화하여 Amazon EC2 스팟 인스턴스에서 직접 게임 서버를 호스팅할 수 있는 기능을 제공합니다. 게임 개발자는 Amazon GameLift FleetiQ를 새 게임에 사용하거나 기존 게임의 용량을 보충하는 데 사용할 수 있습니다. 이 솔루션은 컨테이너 또는 AWS Shield 및 Amazon Elastic Container Service (Amazon ECS) 와 같은 기타 AWS 서비스의 사용을 지원합니다.

이 일반 공급 릴리스에는 Amazon GameLift FleetiQ 솔루션에 대한 다음 업데이트가 포함됩니다.

- 새 API 작업은 Amazon GameLift FleetiQ 게임 서버 그룹의 모든 활성 인스턴스에 대한 상태 등의 정보를 `DescribeGameServerInstances` 반환합니다.
- 새로운 밸런싱 전략 `ON_DEMAND_ONLY`는 게임 서버 그룹이 온디맨드 인스턴스만 사용하도록 구성합니다. 게임 서버 그룹의 밸런싱 전략을 언제든지 업데이트하여 필요에 따라 스팟 인스턴스와 온디맨드 인스턴스 사용을 전환할 수 있습니다.
- 정식 출시를 위해 다음과 같은 미리 보기 요소가 삭제되었습니다.
  - 게임 서버 리소스의 사용자 지정 정렬 키 사용. 등록 타임스탬프를 기준으로 게임 서버를 정렬할 수 있습니다.
  - 게임 서버 리소스에 대한 태깅.

2020년 4월 16일: 아마존, Unity 및 언리얼 엔진용 서버 SDK GameLift 업데이트

SDK 버전 업데이트: 아마존 GameLift 서버 SDK 4.0.0, 아마존 로컬 1.0.5 GameLift

Amazon GameLift Server SDK의 최신 버전에는 다음과 같은 업데이트된 구성 요소가 포함되어 있습니다.

- Unity 2019에 대해 업데이트된 C# SDK 버전 4.0.0.

- Unreal Engine 버전 4.22, 4.23, 4.24에 대해 업데이트된 Unreal 플러그인 버전 3.3.1.
- Amazon GameLift Local 버전 1.0.5는 C# 서버 SDK 버전 4.0.0을 사용하는 통합을 테스트하기 위해 업데이트되었습니다.

Amazon에서 최신 버전의 Amazon GameLift Server SDK를 다운로드하여 GameLift 시작하십시오.

2020년 4월 2일: Amazon GameLift FleetiQ를 EC2에서 게임 호스팅으로 사용 가능 (공개 미리 보기)

SDK 버전 업데이트: SDK 1.7.310AWS

Amazon GameLift FleetiQ 기능은 게임 호스팅과 함께 사용할 수 있도록 저렴한 스팟 인스턴스의 실행 가능성을 최적화합니다. 이제 이 기능은 관리형 Amazon GameLift 서비스를 통하지 않고 호스팅 리소스를 직접 관리하려는 고객을 위해 확장되었습니다. 이 솔루션은 컨테이너 또는 AWS Shield 및 Amazon Elastic Container Service (Amazon ECS) 와 같은 기타 AWS 서비스의 사용을 지원합니다.

자세히 알아보기:

GameTech 아마존 GameLift FleetiQ의 블로그 게시물

2019년 12월 19일: Amazon AWS GameLift 리소스의 리소스 관리 개선

SDK 버전 업데이트: AWS SDK 1.7.249

이제 Amazon AWS GameLift 리소스에서 리소스 관리 도구를 활용할 수 있습니다. 특히 GameLift 빌드, 스크립트, 플릿, 게임 세션 대기열, 매치메이킹 구성, 매치메이킹 규칙 세트 등 모든 주요 Amazon 리소스에는 이제 Amazon Resource Name (ARN) 값이 할당됩니다. 리소스 ARN은 모든 AWS 지역에서 고유한 일관된 식별자를 제공합니다. 이를 사용하여 리소스별 AWS Identity and Access Management (IAM) 권한 정책을 생성할 수 있습니다. 이제 리소스에는 ARN과 리전별로 고유하지 않은 기존 리소스 식별자가 할당됩니다.

또한 Amazon GameLift 리소스는 이제 태깅을 지원합니다. 태그를 사용하여 리소스를 구성하고, 리소스 그룹에 대한 액세스를 관리하기 위한 IAM 권한 정책을 생성하고, AWS 비용 분석을 사용자 지정하는 등의 작업을 수행할 수 있습니다. Amazon GameLift 리소스의 태그를 관리할 때는 Amazon GameLift API 작업 `TagResource()``UntagResource()`, 및 을 사용하십시오`ListTagsForResource()`.

자세히 알아보기:

- TagResource아마존 GameLift API 레퍼런스에서

- AWS 일반 참조의 AWS 리소스 태깅
- AWS AWS General Reference의 Amazon 리소스 이름

2019년 11월 14일: 중국(베이징) 리전에서 새로운 AWS CloudFormation 템플릿 업데이트

SDK 버전 업데이트: AWS SDK 1.7.210

AWS CloudFormation 아마존용 템플릿 GameLift

이제 Amazon GameLift 리소스를 생성하고 통해 관리할 수 AWS CloudFormation있습니다. 기존 AWS CloudFormation 빌드 및 플릿 템플릿이 현재 리소스에 맞게 업데이트되었으며, 이제 스크립트, 큐, 매치메이킹 구성 및 매치메이킹 규칙 세트에 새 템플릿을 사용할 수 있습니다. AWS CloudFormation 템플릿을 사용하면 특히 여러 지역에 게임을 배포할 때 관련 AWS 리소스 그룹을 관리하는 작업이 크게 단순해집니다.

자세히 알아보기:

- 사용 AWS CloudFormation 설명서의 Amazon GameLift 리소스 유형 참조
- AWS CloudFormation을 사용하여 리소스 관리Amazon GameLift 개발자 가이드에서

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.