



호스팅 가이드

Amazon GameLift Servers



Amazon GameLift Servers: 호스팅 가이드

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

Table of Contents

Amazon GameLift Servers란 무엇인가요?	1
주요 기능	1
Amazon GameLift Servers 작업 방법	4
게임 호스팅 옵션	4
관리형 EC2	5
관리형 컨테이너	5
Anywhere 호스팅	6
하이브리드 호스팅	7
작동 방식	8
핵심 구성 요소	8
구성 요소가 함께 작동하는 방식	14
플레이어 경험	15
전 세계 플레이어에게 빠른 게임 플레이 제공	15
플레이어가 신속하게 게임에 참여할 수 있도록 지원	16
게임 세션을 위해 플레이어를 그룹화하는 창의적인 방법 구축	16
게임 플레이 전반에 걸쳐 안정적인 연결 제공	17
실제 데이터를 기반으로 플레이어 경험 개선	17
향상된 게임 기능을 다른 AWS 서비스와 통합	18
프로덕션에서 게임 호스팅	18
호스팅 상태 및 성능 모니터링	18
게임 서버 업데이트 및 패치 관리	19
성능 최적화 및 크기 조정	19
문제 해결 및 실시간 운영	20
도구 및 통합	21
관련 리소스	21
주요 용어	21
시작	29
시작하기 전에	29
경로 선택	29
Amazon GameLift Servers 빠른 탐색	29
사용자 지정 호스팅 솔루션 구축	30
권장 학습 경로	30
AWS 계정 설정	31
에 가입 AWS 계정	32

관리자 액세스 권한이 있는 사용자 생성	33
Amazon GameLift Servers의 사용자 권한 설정	34
사용자의 프로그래밍 방식 액세스 설정	34
게임의 프로그래밍 방식 액세스 설정	36
IAM 권한 예제	37
IAM 서비스 역할 설정	42
개발 도구 가져오기	46
게임 서버	46
게임 클라이언트의 경우	49
게임 백엔드 서비스의 경우	49
리소스 관리의 경우	49
Amazon GameLift Servers Realtime의 경우	50
자습서: Amazon GameLift Servers 래퍼를 사용한 빠른 온보딩	51
사전 조건	52
개요	52
1단계: 게임 서버 래퍼 가져오기 및 빌드	52
2단계: 게임 서버 빌드 준비	53
3단계: 플릿의 래퍼 구성	54
4단계: 게임 서버 빌드 업로드	55
5단계: 관리형 EC2 플릿 생성	56
6단계: 게임 세션 생성 및 연결	57
7단계: 플릿 관리 및 모니터링	58
8단계: 게임 서버 확장	58
일반적인 문제 해결	59
다음 단계	61
Amazon GameLift Servers 예제	61
사용자 지정 게임 서버 예제	61
게임 호스팅 솔루션 구축	63
개발 로드맵	63
관리형 EC2 개발 로드맵	64
관리형 컨테이너 개발 로드맵	69
Anywhere 개발 로드맵	76
하이브리드 호스팅 개발 로드맵	81
호스팅할 게임 준비	88
게임 엔진 플러그인으로 준비	88
게임 서버 통합	171

게임 서버 빌드 패키징	198
Amazon GameLift Servers 상호 작용	211
백엔드 서비스 구축	216
플레이어 인증	216
게임 클라이언트 기능 통합	217
서버리스 백엔드	222
웹소켓 기반 백엔드	223
호스팅 플릿 배포	225
플릿 특성	225
호스팅 옵션 선택	226
관리형 EC2 플릿	228
관리형 컨테이너 플릿	238
Anywhere 플릿	258
하이브리드 호스팅 솔루션 구축	268
게임 세션 배치 구성	271
대기열 특성	272
모범 사례	273
대기열 생성	273
이벤트 알림 설정	277
게임 호스팅 솔루션 사용자 지정	281
게임 서버 빌드	282
플레이어 세션 및 매치메이킹	294
게임 세션 배치	295
호스팅 리소스	310
게임 호스팅 솔루션 관리	336
게임 호스팅 리소스 관리	336
Amazon GameLift Servers 콘솔 기능	336
콘솔 대시보드	337
빌드	338
플릿	341
에일리어스	361
게임 세션 대기열	364
게임 출시 준비	366
준비	367
테스트 준비	367
출시 준비	368

출시 후 계획	368
를 사용하여 게임 호스팅 리소스 관리 CloudFormation	369
모범 사례	369
CloudFormation 스택 사용	370
빌드 업데이트	373
DDoS 보호	376
주요 이점	376
플레이어 게이트웨이 작동 방식	376
트래픽 흐름	377
핵심 개념	377
플레이어 게이트웨이 모니터링	378
IPv4 및 IPv6 호환성	378
플릿에서 플레이어 게이트웨이 활성화	379
사전 조건	379
플레이어 게이트웨이 모드	379
IPv4 및 IPv6 호환성	380
플레이어 게이트웨이를 게임에 통합	380
백엔드 통합	381
클라이언트 통합	382
통합 테스트	384
모범 사례	384
테스트 및 문제 해결	386
반복 개발을 위한 설정	386
클라우드 기반 테스트 환경 빌드	388
로컬 테스트 설정	391
로컬 테스트 설정(레거시)	396
플릿 문제 디버깅	404
플릿 생성 문제	404
서버 프로세스 문제	406
플릿 삭제 문제	406
플릿 리소스에 원격으로 액세스	406
플릿 인스턴스에 연결	407
컨테이너에 연결	415
규모 조정	420
콘솔에서 플릿 용량을 관리하려면	420
호스팅 용량 제한 설정	421

용량 제한을 설정하려면	421
수동으로 플릿 용량 설정	422
Auto Scaling 일시 중지	423
수동으로 플릿 용량을 설정하려면	423
플릿 용량 Auto Scaling	424
대상 기반 Auto Scaling	425
규칙 기반 Auto Scaling	427
0으로/0에서 플릿 크기 조정 관리	431
스케일 인 동작	431
스케일 아웃 동작	432
제로로/제로에서 스케일링 구성	432
모범 사례	433
컨테이너 플릿 확장	434
모니터링	436
지표 비교	436
소스별 지표 가용성	437
주제	440
CloudWatch를 사용하여 모니터링	440
지표의 차원	441
플릿 지표	443
DDoS 보호(플레이어 게이트웨이) 지표	458
대기열 지표	460
FlexMatch 지표	464
FleetIQ 지표	467
서버 원격 측정 지표로 모니터링	469
원격 측정 지표의 이점	470
시작하기 전 준비 사항	471
구현	472
사용 가능한 지표	473
작동 방식	477
대시보드 조직 및 사용량	478
일반적인 모니터링 시나리오	487
문제 해결 가이드	492
API 직접 호출 로깅	495
CloudTrail의 Amazon GameLift Servers 정보	495
Amazon GameLift Servers 로그 파일 항목 이해	496

서버 메시지 로깅	498
EC2 대 컨테이너 로깅 비교	499
사용자 지정 서버 로깅	500
보안	504
데이터 보호	505
저장 시 암호화	506
전송 중 암호화	506
인터넷워크 트래픽 개인 정보	507
Identity and Access Management(IAM)	508
대상	508
ID를 통한 인증	508
정책을 사용하여 액세스 관리	510
Amazon GameLift Servers에서 IAM을 사용하는 방식	511
자격 증명 기반 정책 예시	517
문제 해결	522
AWS 관리형 정책	524
Amazon GameLift Servers를 사용하여 로깅 및 모니터링	526
규정 준수 확인	526
복원력	527
인프라 보안	528
구성 및 취약성 분석	528
보안 모범 사례	529
플릿 런타임 환경을 최신 상태로 유지	529
포트 구성 보호	530
추가 보안 리소스	531
가격 책정	532
요금 모델	532
예상 요금 산출	533
관리형 호스팅 예상 요금	533
비용 최적화 전략	536
모범 사례	536
리소스 비용 및 사용률 도구	538
성능 모니터링	539
참조 안내서	540
서비스 API(AWS SDK)	540
Amazon GameLift Servers 호스팅 리소스 관리	541

게임 세션 시작 및 플레이어 참여	544
서버 SDK 5.x	545
Server SDK 5.x 업데이트	546
서버 SDK 5.x로 마이그레이션	547
C++ Server SDK	549
C# 서버 SDK	600
Go 서버 SDK	644
C++(Unreal) 서버 SDK: 작업	673
서버 SDK 4 이하	718
C++용 서버 SDK: 작업	719
C#용 서버 SDK: 작업	742
Unreal용 서버 SDK: 작업	763
게임 세션 배치 이벤트	778
배치 이벤트 구문	778
PlacementFulfilled	779
PlacementCancelled	782
PlacementTimedOut	783
PlacementFailed	784
AMI 버전	785
서비스 위치	787
지원되는 AWS 위치	787
관리형 호스팅 위치	791
Amazon GameLift Servers Anywhere의 위치	793
Amazon GameLift Servers FlexMatch의 위치	793
Amazon GameLift Servers 중국	793
서비스 엔드포인트 및 할당량	794
Amazon GameLift Servers API 제한	794
API 제한 참조	794
리소스 수준 스로틀링	803
Amazon GameLift Servers SDK API 제한	803
UDP 핑 비콘	807
UDP 핑 비콘 작동 방식	807
UDP 핑 비콘의 일반적인 사용 사례	807
비콘 엔드포인트 가져오기	808
지연 시간 측정 구현	809
코드 예제	810

릴리스 정보 및 SDK 버전	818
SDK 버전	818
릴리스 노트	830
.....	dccclxxviii

Amazon GameLift Servers이란?

Amazon GameLift Servers를 사용하여 세션 기반 멀티플레이어 게임을 위해 클라우드에서 저비용 전용 서버를 배포, 운영, 확장할 수 있습니다. AWS 글로벌 컴퓨팅 인프라를 기반으로 구축되는 전 세계 플레이어 수요에 맞게 리소스 사용량을 동적으로 조정하면서 고성능, 고신뢰도 게임 서버를 제공하는 데 Amazon GameLift Servers 도움이 됩니다.

주요 기능

글로벌 도달 범위 및 짧은 지연 시간

- 전 세계 AWS 리전 및 로컬 영역에 호스팅을 배포하여 새로운 플레이어에게 게임을 제공합니다. [Amazon GameLift Servers에서 지원하는 위치](#)의 전체 목록을 참조하세요.
- 플레이어가 어디에 위치하든 지연 시간을 최소화하고 뛰어난 플레이어 경험을 제공합니다.
- 다중 위치 배치 대기열을 사용하여 플레이어를 최적의 플레이어 경험으로 게임 세션에 동적으로 배치합니다.

관리형 플릿의 자동 조정

- 서비스 품질을 유지하면서 비용 효율성을 위해 플릿 용량을 최적화합니다.
- 대상 기반 Auto Scaling을 사용하여 플레이어 수요에 따라 용량 조정
- 용량 버퍼를 유지하여 갑작스러운 플레이어 유입을 처리합니다.

유연한 호스팅 옵션

자세한 내용은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

- Amazon GameLift Servers에 의해 관리되는 클라우드 게임 호스팅
 - Amazon Elastic Compute Cloud(Amazon EC2) 인프라에서 게임 서버를 호스팅합니다. 컴퓨팅 최적화(C 패밀리), 메모리 최적화(R 패밀리), 비용 효율적인 ARM 기반 Graviton 인스턴스를 포함한 리소스 유형 중에서 선택합니다.
 - Amazon GameLift Servers가 게임 서버를 배포하고 게임 세션을 시작하는 것부터 플레이어를 매칭하여 함께 게임에 참여하는 것에 이르기까지 게임 호스팅 솔루션의 모든 측면을 관리하는 방법을 사용자 지정합니다.

- 다중 위치 플릿을 사용하여 글로벌 게임 서버 배포를 간소화합니다.
- [Amazon EC2 스팟 인스턴스](#)를 사용하여 저렴한 호스팅을 활용합니다.
- 관리형 EC2 호스팅을 사용하여 Windows 또는 Linux 게임 서버를 배포합니다.
- 관리형 컨테이너 호스팅을 사용하여 게임 서버와 함께 Linux 기반 컨테이너를 배포합니다.
- Amazon GameLift Servers Anywhere를 사용하여 사용자가 관리하는 게임 호스팅
 - Amazon GameLift Servers 게임 세션 배치 및 세션 관리 기능을 활용합니다.
 - 자체 하드웨어, 온프레미스 인프라 또는 기타 클라우드 제공업체에 게임 서버 배포 및 조정을 직접 관리합니다.
- 하이브리드 게임 호스팅
 - Amazon GameLift Servers 관리형 플릿과 Anywhere 플릿의 조합을 사용하여 기존 솔루션으로 작업하거나 마이그레이션을 지원하는 게임 호스팅 솔루션을 생성합니다.

게임 세션 관리

- Amazon GameLift Servers에서 게임 서버 프로세스를 관리하고, 게임 서버 상태 및 가용성을 추적하며, 요청 시 새 게임 세션을 배치할 수 있습니다.
- 게임 세션 배치 기능을 사용하면 플레이어가 최상의 플레이어 경험을 제공할 게임 세션을 찾고 참여할 수 있습니다. Amazon GameLift Servers 의사 결정에 의존하거나 호스팅, 비용, 플레이어 지연 시간, 지리적 위치와 같은 기준을 사용자 지정할 수 있습니다.
- 게임 세션 배치 대기열을 생성하여 새 게임 세션 요청을 효율적으로 처리합니다. 기능을 사용하여 게임 및 플레이어 기대치에 맞는 배치 품질과 속도 간의 균형을 설정합니다.
- 선택적인 플레이어 세션 기능을 활용하여 게임 세션에서 플레이어 슬롯을 예약하고, 연결 시 플레이어를 검증하며, 슬롯 가용성을 모니터링할 수 있습니다. 기존 게임 세션을 검색하거나 필터링하여 사용 가능한 슬롯을 찾습니다.

FlexMatch를 사용한 고급 매치메이킹

- Amazon GameLift Servers 호스팅 솔루션과 완전히 통합된 강력한 매치메이킹을 추가합니다.
- 스킬 수준 또는 캐릭터 선택과 같은 플레이어 속성 및 기본 설정을 기반으로 사용자 지정 매치메이킹 규칙을 구축하고 필요에 따라 복잡한 로직을 사용합니다.
- 경쟁적이거나 협력적인 게임 플레이를 위해 팀 기반 매치를 설정합니다. 팀 및/또는 매치 구성에 걸쳐 플레이어들의 균형을 맞춥니다.
- 매치메이킹 알고리즘을 최적화하여 매치 품질과 대기 시간의 균형을 맞춥니다.

- 동시 플레이어 2~200명의 매치를 지원합니다.
- 매치 채우기를 지원하여 사용 가능한 플레이어 슬롯을 채우고 게임 세션을 가득 채웁니다.

포괄적인 모니터링

- 게임 세션 및 플레이어 활동의 사용량과 성능, 관리형 플릿의 호스팅 성능 및 상태에 대한 실시간 지표를 가져옵니다.
- 알림 및 대시보드를 위해 Amazon CloudWatch와 통합합니다.
- 문제 해결을 위해 서버 로그 및 이벤트 추적에 액세스합니다.
- 성능 및 사용률 데이터를 분석하여 규모 조정(관리형 플릿) 또는 리소스 할당(자체 관리형 플릿)을 최적화합니다.

사용자 지정 및 확장성

- 지연 시간, 비용, 위치에 대한 우선 순위를 지정하는 등 게임 세션 배치 로직을 사용자 지정합니다.
- 관리형 호스팅의 경우 게임 호스팅 리소스를 선택하고 각 인스턴스에서 게임 서버를 시작하고 실행하기 위한 지침을 제공합니다. 관리형 컨테이너의 경우 컨테이너 간에 리소스 할당을 미세 조정합니다.
- 자체 매치메이킹 규칙을 정의합니다.
- 용량 조정을 위한 사용자 지정 전략을 구현합니다.
- 게임 서버 코드를 추가하여 데이터베이스 또는 콘텐츠 스토리지 도구와 같은 기존 AWS 서비스와 통합합니다.
- 사용자 지정 기능으로 백엔드 서비스를 생성하여 플레이어 조인 요청(예: 플레이어 당사자)을 관리하거나, 인증을 처리 또는 사용자 지정 로직 및 지속성을 추가합니다.

AWS 에코시스템과의 통합

- 게임 상태 지속성을 위해 Amazon DynamoDB, Amazon Simple Storage Service, Amazon Aurora DSQL과 같은 서비스를 사용합니다.
- 플레이어 인증을 위해 Amazon Cognito를 구현합니다.
- Amazon Kinesis and Amazon S3를 사용하여 게임 분석을 처리합니다.
- Amazon Chime SDK로 음성 채팅을 추가합니다.
- AWS Lambda 및 Amazon API Gateway를 사용하여 사용자 지정 게임 기능을 생성합니다.

Amazon GameLift Servers 작업 방법

Amazon GameLift Servers와 작업하기 위해 이러한 도구를 사용합니다.

AWS CLI

AWS Command Line Interface (AWS CLI)를 사용하여 옹 서비스 API를 포함하여 AWS SDK를 호출합니다. Amazon GameLift Servers. AWS Command Line Interface 사용 설명서의 [AWS CLI 시작하기](#) 섹션을 참조하세요.

Amazon GameLift Servers 콘솔

[Amazon GameLift Servers용 AWS Management Console](#)을 사용하여 리소스를 구성하고, 게임 서버 배포를 관리하며, 성능과 사용량 지표를 추적합니다. Amazon GameLift Servers 콘솔은 프로그래밍 방식이나 AWS CLI를 통해 리소스 관리를 대신할 GUI입니다.

Amazon GameLift Servers SDK

Amazon GameLift Servers SDK에는 게임 클라이언트, 게임 서버, 게임 서비스 및 Amazon GameLift Servers 서비스와 통신하는 데 필요한 라이브러리가 포함되어 있습니다. 자세한 내용은 [Amazon GameLift Servers 개발 도구 가져오기](#) 단원을 참조하십시오.

AWS CloudFormation

AWS CloudFormation 를 사용하여 인프라는 코드로 처리하여 게임 호스팅 솔루션의 AWS 리소스를 모델링, 프로비저닝 및 관리합니다. 리소스를 설명하는 템플릿을 생성하고 지정한 위치에 리소스를 구성하고 배포하는 작업을 CloudFormation 자동화합니다.

Amazon GameLift Servers 게임 호스팅 옵션

Amazon GameLift Servers는 세션 기반 멀티플레이어 게임을 위한 게임 서버를 호스팅하기 위해 다양한 옵션을 제공합니다. Amazon GameLift Servers에서 관리하는 클라우드 기반 호스팅을 설정할지 아니면 이미 보유하고 있는 호스팅 리소스를 통합할지에 상관없이 Amazon GameLift Servers와 협력하여 플레이어에게 필요한 호스팅 솔루션을 구축할 수 있습니다. 게임 호스팅 솔루션의 모양에 대한 자세한 내용은 [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#) 섹션을 참조하세요.

주제

- [관리형 EC2](#)
- [관리형 컨테이너](#)
- [Anywhere 호스팅](#)

- [하이브리드 호스팅](#)

관리형 EC2

관리형 EC2 호스팅을 사용하여 프로덕션 게임 서버를 관리하는 작업을 Amazon GameLift Servers로 오프로드합니다. Amazon Elastic Compute Cloud(Amazon EC2) 및 AWS 글로벌 컴퓨팅 인프라의 고성능과 안정성에 여전히 의존하면서 멀티플레이어 게임 서버에 대한 최적화를 활용합니다.

[Amazon GameLift Servers 관리형 EC2 호스팅 솔루션 구축 시작](#)

특성

- Amazon Linux 또는 Windows Server 운영 체제에서 실행되는 멀티플레이어 게임을 호스팅합니다.
- Amazon GameLift Servers를 사용하여 사용자 지정 게임 서버를 호스팅 리소스 AWS 클라우드에 배포하고 관리합니다. 하드웨어 유형과 배포 위치를 선택하고 필요한 경우에만 추가 세부 정보를 구성합니다.
- 필요에 따라 게임 서버 프로세스를 자동으로 유지하여 게임 세션을 호스팅하도록 런타임 프로세스 관리자를 구성합니다.
- Amazon GameLift Servers가 성능 저하 게임 서버를 감지하고 해결하는 데 도움이 되도록 사용자 지정 상태 추적을 설정합니다.
- Amazon GameLift Servers 성능 모니터링을 활용합니다. 하드웨어 성능 및 서버 프로세스 수명 주기와 같은 지표를 추적합니다. 게임 세션 로그를 다운로드하고 저장할 수도 있습니다.
- Amazon GameLift Servers를 사용하여 게임 서버 프로세스와 기타 AWS 리소스 간의 액세스를 관리합니다.
- 프로덕션 호스팅의 경우 AWS CloudFormation 템플릿 및를 사용하여 게임 호스팅 배포 및 관리를 자동화합니다 AWS Cloud Development Kit (AWS CDK). AWS CodePipeline와 같은 지속적 통합 및 지속적 전송(CI/CD) 도구 및 서비스를 활용합니다.

관리형 컨테이너

Amazon GameLift Servers는 컨테이너화된 게임 서버를 위한 완벽한 클라우드 호스팅 솔루션을 제공하므로 이식성, 민첩성, 내결함성 등 컨테이너 사용의 핵심 이점을 활용할 수 있습니다. 관리형 EC2 호스팅과 마찬가지로 관리형 컨테이너 호스팅은 Amazon EC2 인스턴스에서 컨테이너를 배포하고 실행합니다.

[컨테이너화된 게임 서버를 위한 Amazon GameLift Servers 관리형 호스팅 솔루션 개발을 시작합니다.](#)

특성

- 경량 컨테이너로 사용자 지정 아키텍처를 개발하여 게임 서버 빌드, 종속성, 보조 소프트웨어를 실행합니다.
- Docker 도구를 사용하여 Linux 기반 컨테이너 이미지를 생성합니다. 배포를 위해 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 이미지를 저장합니다.
- Amazon GameLift Servers를 사용하여 사용자 지정 게임 서버를 호스팅 리소스 AWS 클라우드에 배포하고 관리합니다. 하드웨어 유형과 배포 위치를 선택하고 필요한 경우에만 추가 세부 정보를 구성합니다.
- 게임 서버 버전을 모델링하는 도구를 사용하여 호스팅 플릿 수명 주기를 관리합니다. 필요에 따라 게임 서버 업데이트 및 기타 구성 변경 사항을 배포합니다.
- 컨테이너별 성능 지표를 사용하여 게임 호스팅 성능을 추적합니다. 하드웨어 지표를 사용하여 플릿 리소스의 상태를 모니터링합니다.
- 용 AWS CloudFormation 템플릿을 사용하여 컨테이너 플릿 리소스를 관리합니다 Amazon GameLift Servers.

Anywhere 호스팅

Anywhere 호스팅을 사용하면 매치메이킹 및 게임 세션 지표를 비롯한 Amazon GameLift Servers 게임 세션 관리 기능을 활용하여 원하는 곳에서 게임 서버를 호스팅할 수 있습니다. 게임 서버 배포, 게임 서버 상태 모니터링, 용량 조정을 자체 관리합니다.

Note

Anywhere 플릿은 빠르고 반복적인 게임 개발을 위한 테스트 환경으로 특히 유용합니다. 자체 로컬 워크스테이션 또는 클라우드 기반 호스팅 리소스에 대한 Anywhere 플릿을 설정합니다. Anywhere와 함께 테스트하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#) 섹션을 참조하세요.

게임용 Amazon GameLift Servers Anywhere 호스팅 솔루션 개발을 시작하세요.

특성

- 자체 하드웨어, 온프레미스 인프라 또는 기타 클라우드 공급자에 게임 서버를 배포합니다. 플레이어와 가까운 위치에 있는 사용 가능한 하드웨어를 어디에서든 사용하세요.

- Amazon GameLift Servers 세션 관리 기능을 사용하여 게임 서버 가용성을 모니터링하고, 새 게임 세션을 시작하며, 플레이어를 게임 세션 슬롯에 참여시킵니다.

하이브리드 호스팅

프로덕션 호스팅의 경우, 제공하고 관리하는 호스팅 리소스용 Anywhere 플릿과 필요한 경우 채워지는 Amazon GameLift Servers 관리형 플릿이 포함된 하이브리드 접근 방식을 선택할 수 있습니다. 하이브리드 솔루션은 동일한 프로세스를 사용하여 게임 세션을 시작하고 플레이어를 연결하여 원활한 플레이어 경험을 제공합니다. 리소스 유형에 관계없이 모든 호스팅에 대한 사용량 지표를 모니터링하고 수집할 수 있습니다.

하이브리드 솔루션이 적합한 일반적인 시나리오는 다음과 같습니다.

- 호스팅 솔루션을 AWS 클라우드로 확장합니다. Amazon GameLift Servers 관리형 호스팅을 추가하여 기존 호스팅 솔루션(온프레미스 하드웨어 또는 기타 클라우드 기반 호스팅)의 기능을 보완합니다. 관리형 호스팅을 사용하면 호스팅 용량을 늘리거나 '버스트' 용량을 추가하여 빠르게 확장하고 필요할 때만 리소스 비용을 지불할 수 있습니다. 또한 Amazon GameLift Servers 서비스의 글로벌 풋프린트를 활용하여 전 세계 더 많은 플레이어를 연결하고 플레이어에게 지연 시간이 짧은 멀티플레이어 경험을 제공할 수 있습니다.
- 클라우드 기반 호스팅으로 마이그레이션할 준비를 합니다. AWS 클라우드로 마이그레이션하는 것을 고려하거나 계획하고 있다면 하이브리드 호스팅 솔루션은 필요에 따라 점진적으로 전환할 수 있는 방법이 될 수 있습니다. 이를 통해 현재 필요한 솔루션을 구축하는 동시에 향후 원하는 방향을 준비할 수 있습니다.
- Amazon GameLift Servers 서비스 범위를 넘어선 위치의 플레이어에 대해 짧은 지연 시간을 제공합니다. Amazon GameLift Servers 관리형 호스팅을 이미 사용하고 있는 경우 비정상적인 상황에서 플레이어를 지원해야 할 수 있습니다. 예를 들어, 비정상적으로 멀리 떨어진 위치에 있는 플레이어와 연결하거나, 특별한 액세스 지원이 필요한 플레이어를 지원하고자 할 수 있습니다. 사용자 지정 호스팅 위치를 추가하고 Amazon GameLift Servers Anywhere를 사용하여 관리형 호스팅 리소스와 함께 해당 위치를 관리합니다.
-

[게임용 Amazon GameLift Servers 하이브리드 호스팅 솔루션 개발을 시작하세요.](#)

Amazon GameLift Servers를 사용한 호스팅 작동 방식

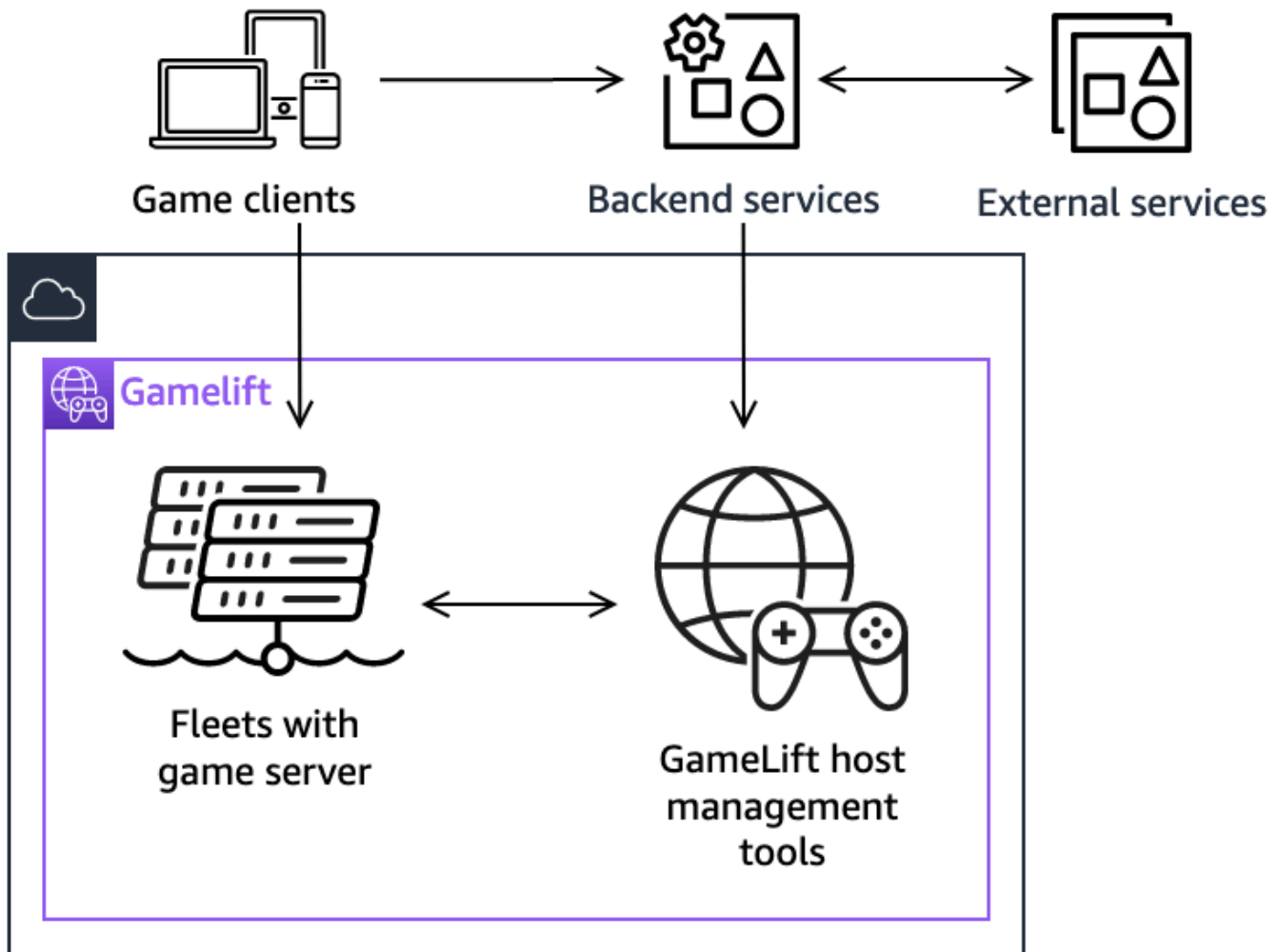
Amazon GameLift Servers는 세션 기반 멀티플레이어 게임을 위한 저렴한 전용 서버, 인프라 프로비저닝, 규모 조정, 세션 관리를 제공하는 서비스입니다. 이 서비스는 게임에 맞게 사용자 지정하거나 자체 시스템과 협력하여 사용할 수 있는 유연한 도구와 기능을 제공합니다. Amazon GameLift Servers 관리형 호스팅을 활용하면 서버 작업을 관리하는 대신 게임 플레이 경험을 만드는 데 집중할 수 있습니다.

이 주제는 완전한 Amazon GameLift Servers 호스팅 솔루션의 핵심 구성 요소와 이러한 구성 요소가 함께 작동하여 플레이어에게 멀티플레이어 게임 세션을 공급하는 방법에 대해 알아봅니다.

게임 호스팅 솔루션의 핵심 구성 요소

완전한 Amazon GameLift Servers 호스팅 솔루션은 멀티플레이어 게임 경험을 제공하기 위해 함께 작동하는 몇 가지 필수 구성 요소로 구성됩니다. 이러한 구성 요소 중 일부는 사용자가 빌드하고 관리하는 반면, 구성 선택에 따라 Amazon GameLift Servers에서 제공하고 관리하는 구성 요소도 있습니다. 효과적인 게임 호스팅 솔루션을 개발하려면 이러한 구성 요소가 상호 작용하는 방식과 해당 역할을 이해하는 것이 중요합니다. 구성 요소에는 다음이 포함됩니다.

- 게임 클라이언트는 플레이어 디바이스에서 실행됩니다.
- 백엔드 서비스를 사용하면 게임 클라이언트가 Amazon GameLift Servers와 통신하여 게임 세션 정보를 얻고 게임에 참여할 수 있습니다.
- 게임 서버 소프트웨어는 게임 호스팅 리소스에서 실행되며 플레이어 그룹을 위한 게임 세션을 호스팅합니다.
- 배치 시스템은 게임 세션을 시작하고 플레이어를 게임과 일치시킵니다.
- 게임 호스팅 플릿은 하나 이상의 지리적 위치에 게임 서버를 제공합니다.
- 게임 호스팅 관리 시스템은 게임 호스팅 상태를 모니터링하고 용량을 관리합니다.



게임 클라이언트

게임 클라이언트는 플레이어의 디바이스에서 실행되는 게임 소프트웨어입니다. 백엔드 서비스와 통신하여 게임 세션에 플레이어 배치를 시작하고 게임 서버에 직접 연결하여 게임 플레이에 참여합니다.

게임 호스팅을 위한 주요 기능

- 백엔드 서비스에 조인 요청을 전송합니다. 조인 요청의 일부로 관련 플레이어 데이터 및 게임 세션 데이터를 포함합니다.
- 게임 클라이언트의 지연 시간 데이터를 수집하고 해당 정보를 조인 요청에 포함합니다.
- 백엔드 서비스에서 게임 세션 연결 정보를 수신하고 이를 사용하여 게임 서버에 연결합니다.
- 플레이어 확인, Amazon GameLift Servers FlexMatch의 매치 수락, 세션 중단 또는 연결 해제와 같은 세션 연결 시나리오를 처리합니다.

빌드 대상

게임 클라이언트 소프트웨어에 게임 호스팅 기능을 추가하고 백엔드 서비스와의 통신을 설정합니다.

백엔드 서비스

백엔드 서비스는 게임 클라이언트와 Amazon GameLift Servers 서비스 간의 조정 계층입니다. 서비스와의 모든 통신을 제어하여 게임 세션 배치를 요청하고 게임 세션 및 플레이어 세션 정보를 검색합니다. 백엔드 서비스 사용은 게임 클라이언트를 대신하여 서비스와의 보안 통신을 유지하고 게임 클라이언트와 민감한 AWS 자격 증명 및 권한을 공유할 필요가 없는 모범 사례입니다.

백엔드 서비스는 게임 세션에 대한 플레이어 그룹화 전략을 구현합니다. 플레이어가 게임 세션에 참가하는 방식에는 상당한 유연성이 있습니다. 플레이어를 위한 게임 세션 브라우저를 설정하거나 플레이어 정의 당사자를 지원할 수 있습니다. 아니면 게임 세션 요청이 들어올 때 풀링하고, 플레이어 그룹을 구성하고(사전 정렬 가능), 각 그룹에 대해 게임 세션을 시작할 수 있습니다. FlexMatch 매치메이킹을 사용하면 플레이어 매치를 구성하고 기존 매치를 채울 수 있습니다.

게임 호스팅을 위한 주요 기능

- 게임 클라이언트의 통신을 인증합니다.
- Amazon GameLift Servers에 새 게임 세션을 시작하거나, 게임 세션 정보를 가져오거나, 기존 게임 세션에 참여하도록 요청합니다. 필요에 따라 해당 요청에 관련 플레이어 세션 및 플레이어 데이터를 포함합니다.
- 선택적으로 플레이어 세션에 대한 요청을 생성하여 게임 세션 가용성을 더 면밀히 모니터링할 수 있습니다. 플레이어 세션을 사용하여 게임 세션 슬롯을 예약하고, 플레이어가 연결될 때 플레이어를 검증하며, 플레이어 연결 해제를 추적할 수 있습니다.
- 요청에 따라 게임 세션 연결 세부 정보 및 기타 정보를 사용하여 게임 클라이언트에 응답합니다.

빌드 대상

게임 요구 사항을 지원하고 플레이어가 게임에 참여하는 방식을 구현하기 위해 사용자가 관리하는 환경에서 백엔드 서비스를 빌드합니다. AWS SDK를 통합하여 Amazon GameLift Servers 서비스 API를 호출합니다. 선택적으로 AWS Lambda Amazon Simple Storage Service(Amazon S3) 및 Amazon Cognito와 같은 서비스를 AWS 사용하여 백엔드 서비스를 빌드합니다.

게임 서버

게임 서버는 게임 상태를 관리하고, 플레이어 작업을 처리하며, 연결된 여러 플레이어 간의 게임 플레이를 동기화해주는 사용자 지정 서버 소프트웨어입니다. 게임 서버는 Amazon GameLift Servers 서비스와의 통신을 유지하여 게임 세션 호스팅을 관리합니다.

게임 호스팅을 위한 주요 기능

- Amazon GameLift Servers와 통신할 대상:
 - 보고서 상태(세션을 호스팅할 준비가 됨, 플레이어를 수락할 준비가 됨, 상태)입니다.
 - 서비스 호출(게임 세션 시작 또는 종료)합니다.
- 게임 세션 수명 주기를 관리하여 프로세스당 한 번에 하나의 게임 세션을 호스팅합니다.
- 추가 기능을 위해 다른 AWS 서비스와 조정합니다.
- 선택적으로 새 플레이어 연결을 검증합니다.

빌드 대상

게임 서버 소프트웨어를 빌드합니다. Amazon GameLift Servers용 서버 SDK를 통합하고 기능을 추가하여 서비스와의 연결을 설정하고 게임 세션 관리를 지원합니다.

게임 호스팅 플릿

호스팅 플릿은 게임 서버를 실행하는 컴퓨팅 리소스 컬렉션입니다. 플릿 리소스를 여러 지리적 위치에 분산하여 플레이어가 어디에 있든 지연 시간이 짧은 게임 플레이를 제공할 수 있습니다. 각 플릿 리소스는 Amazon GameLift Servers와 직접 통신하는 하나 이상의 게임 서버 프로세스를 실행합니다. 각 게임 서버 프로세스는 한 번에 하나의 게임 세션을 호스팅할 수 있습니다.

플릿의 특성과 기능은 플릿의 호스팅 유형에 따라 다릅니다. 관리형 플릿은 리소스를 배포 AWS 클라우드 하고에서 관리합니다 Amazon GameLift Servers. Anywhere 플릿은 Amazon GameLift Servers 외부에서 관리되는 고객 제공 컴퓨팅 리소스입니다.

게임 호스팅을 위한 주요 기능

- 게임 서버 호스팅 리소스를 프로비저닝합니다.
 - 관리형 플릿은 다양한 구성 설정으로 클라우드 기반 Amazon EC2 인스턴스를 배포합니다. 플릿 구성은 플릿에 있는 각 인스턴스의 컴퓨팅 성능, 인스턴스의 물리적 위치, 기타 세부 정보를 결정합니다. 관리형 플릿은 용량 조정에 대한 응답으로 EC2 인스턴스를 추가하거나 제거합니다.

- 모든 플릿 배포는 자체 관리형입니다. 플릿은 물리적 하드웨어 또는 기타 클라우드 리소스로 구성되며 필요에 따라 구성할 수 있습니다.
- 런타임 환경 및 게임 서버 소프트웨어를 설치합니다.
 - 관리형 Amazon EC2 Fleet 인스턴스는 게임 서버 빌드 및 호환되는 런타임 환경을 제공하는 Amazon Machine Image(AMI)와 함께 배포됩니다.
 - 관리형 컨테이너 플릿 인스턴스는 게임 서버 빌드 및 Docker 도구와 Amazon ECS 작업을 위한 다른 구성 요소들을 갖춘 컨테이너 최적화 AMI와 함께 배포됩니다.
 - Anywhere 플릿은 선택한 게임 서버 소프트웨어 및 운영 체제와 함께 배포됩니다.
- 게임 서버 프로세스 수명 주기를 관리합니다. 사전 구성된 런타임 지침을 적용하여 각 플릿 리소스에서 프로세스를 시작하고 중지합니다.

빌드 대상

이는 플릿의 호스팅 유형에 따라 다릅니다.

- 관리형 Amazon EC2 플릿: 게임 서버 빌드를 업로드하여 플릿 인스턴스에 배포하기 위해 Amazon GameLift Servers에 저장합니다.는 플릿의 AWS 클라우드 인프라를 Amazon GameLift Servers가 제공합니다. 플릿을 구성하고 각 플릿 인스턴스에서 게임 서버를 실행하는 방법을 지시합니다.
- 관리형 컨테이너 플릿: 배포를 위해 Amazon ECR에 저장할 컨테이너 이미지에 게임 서버 빌드 및 런타임 지침을 패키징합니다.는 플릿의 AWS 클라우드 인프라를 Amazon GameLift Servers가 제공합니다. 컨테이너 아키텍처를 제공하고 컨테이너를 호스팅하도록 플릿을 구성합니다.
- Anywhere 플릿: 모든 인프라를 프로비저닝하고 서버 소프트웨어 배포를 관리합니다. Anywhere 플릿을 생성하여 활성 게임 호스팅 리소스를 Amazon GameLift Servers와 연결합니다.

게임 세션 배치 시스템

게임 세션 배치 시스템은 새 게임 세션을 호스팅할 수 있는 사용 가능한 게임 서버를 찾습니다. 시스템은 게임 서버 가용성에 대한 실시간 정보를 사용하여 최적의 배치 결정을 내립니다.

Amazon GameLift Servers에서 기본 게임 세션 배치 메커니즘은 대기열입니다. 게임 세션 대기열은 최상의 결과를 위해 게임 세션을 배치하도록 구성할 수 있는 알고리즘을 사용합니다. 호스팅 비용 및 플레이어 지연 시간 최소화와 같은 요인에 따라 배치의 우선순위를 정하고 여러 지리적 위치에서 검색하도록 대기열을 구성할 수 있습니다. 대기열 대신 게임 세션을 호스팅할 특정 플릿을 지정할 수 있습니다.

게임 호스팅을 위한 주요 기능

- 백엔드 서비스에서 받은 게임 세션 배치 요청을 처리합니다.
- 호스팅 리소스 가용성에 대한 실시간 정보를 기반으로 배치합니다.
- 플레이어 지연 시간 데이터 및 기타 데이터를 사용하여 배치 옵션(대기열만 해당)의 우선 순위를 지정합니다.
- 새 게임 세션을 시작하도록 게임 서버 프로세스를 유도합니다.
- 게임 세션이 플레이어를 수락할 준비가 된 후 게임 세션 연결 정보를 업데이트합니다.
- 선택적으로 FlexMatch 매치메이킹을 설정하여 플레이어 매치를 생성하고 매치에 대한 게임 세션 배치를 요청합니다.

빌드 대상

Amazon GameLift Servers는 배치 시스템을 제공합니다. 백엔드 서비스에서 배치 요청을 수행하는 방법과 선택적으로 게임 세션 대기열 및 FlexMatch 매치메이커를 설정하여 배치 동작을 구성합니다. Amazon GameLift Servers 콘솔, AWS SDK 또는를 사용하여 게임에 대한 대기열 및 매치메이커를 AWS CLI 생성하고 구성합니다.

게임 호스팅 관리 시스템

게임 호스팅 관리 시스템은 게임 호스팅 솔루션의 모든 측면을 조정하고 모니터링하는 운영 백본입니다. 이 시스템은 Amazon GameLift Servers의 배치 및 조정 기능을 가능하게 하는 인텔리전스와 자동화를 제공합니다.

게임 호스팅을 위한 주요 기능

- 모든 플릿에서 게임 서버 프로세스, 게임 세션, 플레이어 세션의 실시간 상태 및 가용성을 추적하여 게임 세션 배치 및 자동 용량 조정을 지원합니다.
- 플릿 상태 및 성능을 모니터링합니다.
- 게임 호스팅 활동 지표를 수집하고 분석합니다.
- 플레이어 수요 및 플릿 사용률에 따른 자동 조정을 포함한 용량 조정 도구를 제공합니다.
- 게임 서버 소프트웨어 및 런타임 지침에 대한 업데이트를 관리합니다.

빌드 대상

- 관리형 플릿: Amazon GameLift Servers는 가용성 추적, 게임 호스팅 활동과 하드웨어 성능에 대한 지표, 용량 조정 도구를 제공합니다. 사용 가능한 AWS 도구를 사용하여 게임 서버 소프트웨어 업데이트

이트를 관리하고, 런타임 지침을 수정하고, 지표 사용(예: 모니터링을 위한 CloudWatch 설정)을 사용자 지정합니다. 사용자 지정 조정 정책을 구성하고 필요에 따라 수정합니다.

- Anywhere 플릿: Amazon GameLift Servers는 가용성 추적 및 게임 호스팅 활동 지표를 제공합니다. 게임 서버 소프트웨어 업데이트 및 런타임 지침을 포함하여 플릿 구성 변경을 관리합니다. 플릿 성능을 모니터링하고 플릿 용량 조정을 관리하는 시스템을 생성합니다.

구성 요소가 함께 작동하는 방식

게임 호스팅 컴퓨팅이 배포된 경우

- 게임 서버 소프트웨어 설치됨: 컴퓨팅이 런타임 환경 및 게임 서버 빌드와 함께 설치되었습니다.
- 게임 서버 시작: 컴퓨팅에서 선택적 시작 파라미터와 함께 게임 서버 실행 파일의 인스턴스가 하나 이상 시작됩니다.
- 게임 서버를 Amazon GameLift Servers에 연결: 게임 서버 프로세스는 시작 작업의 일부로 서버 SDK를 호출하여 서비스에 대한 연결을 초기화합니다.
- 게임 서버가 게임 세션을 호스팅할 준비가 되었음을 보고: 게임 서버 프로세스는 시작 작업을 완료한 다음 서버 SDK를 호출하여 준비 상태를 보고합니다. 구성에 따라 상태 보고를 시작합니다.
- Amazon GameLift Servers가 가용성 추적: 해당 서비스는 게임 세션 배치를 위한 게임 서버 프로세스의 가용성을 기록합니다. 또한 플릿, 인스턴스, 게임 세션에 대한 지표를 업데이트하여 사용량과 용량을 추적합니다.
- 게임 서버가 게임 세션 할당 대기: 게임 서버 프로세스는 게임 세션을 시작하라는 프롬프트를 기다리는 동안 유휴 상태를 유지합니다.

플레이어가 게임에 참여하려는 경우

- 플레이어가 게임 시작: 플레이어가 게임 클라이언트를 시작하고, 백엔드 서비스로 인증하며, 게임에 참여하도록 요청합니다.
- 백엔드 서비스가 게임 세션 요청: 백엔드 서비스는 플레이어 그룹화 전략에 따라 적절한 게임 세션을 찾거나 생성하기 위해 Amazon GameLift Servers를 호출합니다. 요청에는 게임 세션에 사용할 플레이어 또는 게임 데이터가 포함될 수 있습니다.
- Amazon GameLift Servers가 게임 세션 배치: 새 게임 세션을 시작할 때 배치 시스템은 세션을 호스팅할 최적의 위치를 식별하고 사용 가능한 게임 서버 프로세스를 선택합니다. 서비스는 선택한 프로세스에 새 게임 세션을 시작하라는 프롬프트를 표시하고 모든 플레이어 또는 게임 데이터를 전달합니다. FlexMatch 매치메이킹을 사용하는 경우 매치메이커는 먼저 매치를 생성한 다음 매치에 대한 배치를 요청합니다.

- 게임 서버가 게임 세션 시작: 게임 서버가 게임 세션을 시작하는 단계를 시작합니다. 완료되면 플레이어 연결을 수락할 준비가 되었음을 Amazon GameLift Servers에 보고합니다.
- 게임 클라이언트에 전달된 연결 정보: 게임 서버가 상태를 업데이트한 후 Amazon GameLift Servers는 게임 세션 연결 정보를 제공합니다. 백엔드 서비스는 이 정보를 수신하여 게임 클라이언트에 전달합니다.
- 게임 클라이언트에 플레이어 연결: 게임 클라이언트는 연결 정보를 사용하여 게임 서버에 직접 연결하고 게임 플레이를 시작합니다.
- Amazon GameLift Servers가 게임 세션 상태 모니터링: 게임 서버 프로세스는 상태, 선택적 플레이어 연결 상태, 게임 세션 상태를 보고하여 지속적인 게임 세션 가용성을 추적합니다.
- 게임 서버 프로세스 종료: 게임 서버 프로세스가 게임 세션을 종료하고, 상태를 보고하며, 자체적으로 종료합니다.

Amazon GameLift Servers 및 플레이어 경험

플레이어는 빠르고 공정하며 안정적인 멀티플레이어 게임 경험을 기대합니다. Amazon GameLift Servers는 플레이어가 멀티플레이어 게임에 연결하고 참여하는 방법을 직접 향상시키는 도구를 게임 개발자에게 제공하여 이러한 경험을 제공합니다. 이 주제에서는 Amazon GameLift Servers 기능이 제공하는 주요 플레이어 이점을 간략하게 설명합니다.

전 세계 플레이어에게 빠른 게임 플레이 제공

플레이어는 지연을 최소화하면서 가까운 서버에 연결하려고 합니다. Amazon GameLift Servers는 지능형 배치 및 글로벌 인프라를 통해 최적의 연결 환경을 제공합니다.

- 플레이어는 전 세계적으로 일관된 성능을 경험합니다 - 플레이어와 게임 서버 간의 거리를 최소화하기 위해 전 세계적으로 사용할 수 있는 AWS 리전 및 로컬 영역에서 게임을 호스팅합니다(참조 [Amazon GameLift Servers 서비스 위치](#)). 이러한 위치에서 다양한 Amazon EC2 인스턴스 유형 중에서 선택하여 게임 워크로드를 호스팅하는 데 가장 적합한 컴퓨팅 리소스를 찾을 수 있습니다.
- 사용 가능한 최고의 서버에 플레이어 연결 - Amazon GameLift Servers는 게임 클라이언트 네트워크 데이터를 수집하고 여러 위치에서 사용 가능한 서버를 검색하여 플레이어를 가능한 한 짧은 지연 시간으로 게임 세션에 배치합니다. UDP 핑 비콘을 사용하여 정확한 지연 시간 데이터를 수집합니다. 연결 품질과 플레이어의 허용 가능한 대기 시간의 균형을 맞추도록 지연 시간 정책을 구성합니다.
- 다른 리전의 플레이어와 플레이 가능 - 게임 세션 배치는 허용 가능한 지연 시간을 유지하면서 리전 플레이어 풀의 플레이어와 게임 세션의 균형을 맞출 수 있습니다. 덜 채워진 리전의 플레이어는 연결 품질을 저하시키지 않고 매치를 찾을 수 있습니다.

플레이어가 신속하게 게임에 참여할 수 있도록 지원

플레이어는 수요에 관계없이 게임에 빠르게 액세스하려고 합니다. Amazon GameLift Servers는 플레이어가 피크 및 정적 기간에 게임을 찾을 수 있도록 지원하는 다양한 용량 규모 조정 도구를 제공합니다.

- 플레이어가 사용 가능한 서버를 쉽게 찾을 수 있음 - 자동 조정 도구를 사용하면 호스팅 리소스가 필요할 때 준비되도록 할 수 있습니다. 대상 추적은 유연한 용량 버퍼를 관리하여 게임의 플레이어 수요 패턴을 예측하고 사전에 규모를 조정합니다.
- 특별 이벤트 중 플레이어가 원활한 성능 경험 - 자동 크기 조정과 수동 크기 조정 기술을 결합하여 게임 업데이트, 마케팅 캠페인 또는 토너먼트와 같은 계획된 이벤트 후 갑작스러운 유입을 처리합니다.
- 플레이어는 서버가 시작될 때까지 대기할 필요 없음 - 모든 게임 호스팅 리소스에서 게임 서버는 플레이어가 참여하는 즉시 항상 새 세션을 호스팅할 준비가 되도록 사전 워밍업됩니다.

게임 세션을 위해 플레이어를 그룹화하는 창의적인 방법 구축

플레이어는 공정하고 균형 잡힌 방식으로 다른 플레이어와 경쟁하거나 협업하고자 합니다. Amazon GameLift Servers는 플레이어를 게임 세션으로 그룹화하기 위한 다양한 옵션을 제공합니다.

- 플레이어를 그룹화하여 플레이어 경험 최적화 - 다양한 기능을 사용하여 게임에 가장 적합하도록 플레이어 그룹화를 관리할 수 있습니다. 플레이어가 스킬 레벨, 지연 시간, 기타 속성을 기반으로 게임 세션 또는 그룹 플레이어 요청을 검색하거나 탐색할 수 있도록 해줍니다. 플레이어 그룹화를 직접 관리하거나 사용자 지정 가능한 매치 규칙을 사용하여 FlexMatch 매치메이킹을 구현할 수 있습니다.
- 플레이어가 활성 매치에 참여할 수 있음 - 플레이어 그룹을 구축하여 새로운 게임 세션을 시작하거나, 플레이어 세션을 사용하여 기존 세션의 사용 가능한 슬롯을 추적하고 새로운 플레이어를 추가할 수 있습니다. FlexMatch를 사용하면 잘 매칭된 플레이어로 세션을 자동으로 채웁니다.
- 친구와 팀으로 함께 플레이할 수 있음 플레이어 파티 지원을 통해 친구 그룹이 함께 플레이할 수 있습니다. 게임 세션은 나머지 슬롯을 다른 플레이어로 채우거나 액세스를 제한할 수 있습니다. FlexMatch를 사용하면 친구가 팀을 구성할 수 있습니다.
- 플레이어는 게임 세션에 사용자 지정 데이터를 가져올 수 있음 - 게임 세션 및 플레이어 세션 요청에 게임 및 플레이어 정보를 포함합니다. 이 정보는 게임 세션에서 사용할 수 있도록 게임 서버로 전달됩니다.
- 플레이어는 완벽한 매치를 무기한 기다리지 않음 - FlexMatch를 사용하면 시간이 지남에 따라 매치 요구 사항을 완화하여 플레이어의 과도한 대기열 이탈을 방지할 수 있습니다. 매치 품질의 가장 중요한 측면을 우선시하는 점진적 완화 규칙을 생성합니다.

- 플레이어는 매치 수락 또는 거부를 선택할 수 있음 - FlexMatch를 사용하면 플레이어가 가능한 매치를 수락하거나 거부하도록 워크플로를 사용자 지정할 수 있습니다.

게임 플레이 전반에 걸쳐 안정적인 연결 제공

플레이어는 중단이나 연결 해제 없이 일관된 게임 환경을 기대합니다. Amazon GameLift Servers는 포괄적인 세션 관리 및 보호를 제공합니다.

- 플레이어는 클라우드에서 AWS 고품질 성능을 누릴 수 있습니다.에서 관리하는 호스팅을 통해 Amazon GameLift Servers플레이어는 AWS 컴퓨팅 인프라에서 실행되는 빠르고 안정적인 게임 서버를 얻을 수 있습니다.
- 연결 시 플레이어 검증 - 게임에 플레이어 세션을 활성화하면 Amazon GameLift Servers는 게임 세션에 플레이어 슬롯을 예약하고 연결 시 플레이어를 검증할 수 있습니다.
- 플레이어는 게임 진행 상황을 재연결하고 유지할 수 있음 - Amazon GameLift Servers는 서버 측 중단과 클라이언트 측 중단 모두에 대한 재연결을 지원합니다.
- 플레이어의 게임은 종료로부터 보호됨 - 선택적 게임 세션 보호 기능은 조정 이벤트 또는 기타 중단 중에 활성 세션이 종료되는 것을 방지합니다.

실제 데이터를 기반으로 플레이어 경험 개선

플레이어는 게임이 지속적으로 작동하고 시간이 지남에 따라 개선될 것으로 예상합니다. Amazon GameLift Servers는 플레이어 경험을 최적화하는 데 도움이 되는 포괄적인 모니터링 및 분석을 제공합니다.

- 플레이어 중단 횟수 감소 경험 - 플릿 성능, 게임 세션, 플레이어 활동을 실시간으로 모니터링하면 게임 플레이에 상당한 영향을 미치기 전에 문제를 식별할 수 있습니다. 정상적인 게임 세션 종료 및 마이그레이션을 사용자 지정할 수 있습니다.
- 플레이어는 사전 예방적 서버 상태 모니터링에서 혜택 - Amazon GameLift Servers는 지속적인 서버 상태 모니터링을 제공하고 비정상 게임 서버를 자동으로 교체하여 하드웨어 또는 소프트웨어 장애 영향을 최소화합니다. 상태 확인 파라미터를 구성하여 다양한 게임 요구 사항을 지원합니다.
- 플레이어는 게임 플레이에서 데이터 기반 개선 획득 - 게임 세션 로그와 세부 분석을 활용하여 플레이어 동작 및 서버 성능의 패턴을 공개합니다. 게임별 이벤트에 대한 사용자 지정 로깅 지원을 추가합니다.

향상된 게임 기능을 다른 AWS 서비스와 통합

플레이어는 음성 채팅, 보안 인증 및 지속적인 진행과 같은 통합 기능을 원합니다. 이를 다른 AWS 서비스와 통합하여 포괄적인 게임 경험을 제공할 Amazon GameLift Servers 수 있습니다.

- 플레이어의 인증 간소화 - Amazon Cognito 통합은 플레이어를 반환하기 위한 간소화된 로그인 프로세스를 통해 게임 세션 전반에서 보안 플레이어 자격 증명을 유지하면서 다양한 자격 증명 공급자 및 인증 방법을 지원합니다.
- 플레이어는 세션에 걸쳐 게임 진행 상황을 유지할 수 있음 - Amazon DynamoDB 통합을 사용하여 크로스 세션 연속성을 지원하면서 게임 플레이에 미치는 영향을 최소화하는 고성능 액세스를 통해 플레이어 진행 상황, 인벤토리, 영구 데이터를 저장할 수 있습니다.
- 플레이어는 분석 인사이트를 통해 혜택 수혜 - Amazon Kinesis 및 Amazon Simple Storage Service(Amazon S3) 통합은 게임 분석을 처리하여 플레이어 동작 및 선호도에 대한 인사이트를 수집하여 개발자가 변화하는 플레이어 패턴에 적응하고 게임을 최신 상태로 유지하고 흥미를 끌 수 있도록 실시간 분석을 지원합니다.

Amazon GameLift Servers로 프로덕션 환경에서 게임 호스팅 관리

Amazon GameLift Servers로 게임을 시작한 후에는 최적의 성능, 안정성, 플레이어 경험을 보장하기 위해 게임 호스팅 인프라를 관리해야 합니다. 이 주제에서는 프로덕션 중에 게임 호스팅 수명 주기를 관리하기 위한 주요 기능과 도구를 다룹니다.

호스팅 상태 및 성능 모니터링

Amazon GameLift Servers는 게임 호스팅 인프라의 상태와 성능을 추적하는 데 도움이 되는 포괄적인 모니터링 도구를 제공합니다.

- 성능 지표 - 인스턴스 사용률, 플레이어 세션, 게임 세션 배치와 같은 관리형 플릿의 주요 성능 지표를 모니터링합니다. Amazon GameLift Servers 또는 Amazon CloudWatch에서 이러한 지표를 추적하면 지표가 임계값을 초과할 때 알림을 받도록 경보를 설정할 수도 있습니다.
- 플릿 지표 및 이벤트 - 활성 서버 프로세스, 사용 가능한 게임 세션, 활성 플레이어 세션을 포함한 플릿별 지표를 추적하여 최적의 용량을 보장합니다.
- 게임 세션 배치 지표 및 이벤트 - 지표 및 이벤트를 추적하여 게임 세션 배치 시스템의 상태와 성능을 모니터링합니다.
- 게임 세션 로그 - 게임 세션 로그에 액세스하고 분석하여 문제를 식별하고 플레이어 동작 패턴을 이해합니다.

- Realtime 스크립트 로그 - 실시간 서버를 사용하는 게임의 경우 세부 로그를 통해 스크립트 실행 및 성능을 모니터링합니다.
- AWS 상태 대시보드 - Amazon GameLift Servers 배포에 영향을 미칠 수 있는 AWS 서비스 상태에 대한 최신 정보를 제공합니다.

Amazon GameLift Servers 콘솔, AWS CLI를 통해 이러한 모니터링 도구 Amazon GameLift Servers에 액세스하거나 옹 AWS SDK를 사용하여 사용자 지정 대시보드 및 모니터링 솔루션을 생성할 수 있습니다.

게임 서버 업데이트 및 패치 관리

게임 서버를 최신 상태로 유지하는 것은 보안, 성능, 새 기능 추가에 매우 중요합니다. Amazon GameLift Servers는 업데이트를 관리하기 위한 몇 가지 접근 방식을 제공합니다.

- 빌드 관리 - 게임 서버 빌드의 여러 버전을 업로드하고 관리합니다. 각 빌드는 버전이 지정되며 서로 다른 플릿에 배포할 수 있습니다.
- 플릿 교체 - AMI 버전 및 게임 서버 빌드 업데이트를 사용하여 새 플릿을 생성하고 게임 세션 배치 대기열 및 별칭을 사용하여 이전 플릿에서 새 플릿으로 트래픽을 점진적으로 이동합니다.
- 스크립트 업데이트 - 실시간 서버의 경우 새 스크립트 버전을 업로드하여 전체 플릿을 교체하지 않고 서버 스크립트를 업데이트합니다.
- 컨테이너 업데이트 - 컨테이너 기반 배포의 경우 컨테이너 이미지 및 정의를 업데이트합니다. 게임 서버의 새 버전을 기존 플릿에 배포합니다. 플릿을 교체하여 AMI 버전을 업데이트합니다.
- 자동 배포 - AWS CodePipeline 및 AWS CodeDeploy 를 사용하여 자동 게임 서버 업데이트를 위한 CI/CD 파이프라인을 생성합니다.

업데이트를 계획할 때는 블루/그린 배포 전략을 사용하여 활성 플레이어의 중단을 최소화하고 문제가 감지되면 빠른 롤백을 허용하는 것이 좋습니다.

성능 최적화 및 크기 조정

플레이어 기반이 발전함에 따라 최적의 성능과 비용 효율성을 유지하려면 호스팅 구성을 조정해야 합니다.

- Auto Scaling - 플레이어 수 또는 게임 세션 사용률과 같은 지표를 기반으로 플릿 조정 정책을 구성하여 용량을 자동으로 조정합니다.

- 리전별 배포 조정 - 플레이어 인구 통계 및 지연 시간 요구 사항에 따라 다중 리전 배포에서 리전을 추가하거나 제거합니다.
- 대기열 관리 - 게임 세션 배치 대기열 구성을 최적화하여 플레이어 경험과 호스팅 비용의 균형을 맞춥니다.
- 인스턴스 유형 선택 - 성능 지표를 분석하여 게임 서버 요구 사항에 가장 비용 효율적인 인스턴스 유형을 결정합니다.
- 스팟 인스턴스 사용 - 온디맨드 인스턴스에 대한 적절한 대체 전략을 사용하여 중요하지 않은 워크로드에 스팟 인스턴스를 활용하여 비용을 절감합니다.
- FlexMatch 튜닝 - 실제 플레이어 데이터 및 피드백을 기반으로 매치메이킹 규칙 및 알고리즘을 세부 조정합니다.

CloudWatch 지표 및 비용 보고서를 정기적으로 검토하여 최적화 기회를 식별하고 Amazon GameLift Servers 콘솔 또는 API를 통해 변경 사항을 구현합니다.

문제 해결 및 실시간 운영

긍정적인 플레이어 경험을 유지하려면 효과적인 문제 해결 및 실시간 운영 관리가 필수입니다.

- 플릿 이벤트 알림 - 조정 활동, 인스턴스 종료 또는 게임 세션 배치 실패와 같은 플릿 이벤트에 대한 Amazon Simple Notification Service 알림을 설정합니다.
- 게임 세션 배치 디버깅 - 자세한 배치 로그를 사용하여 게임 세션 배치 문제를 식별하고 해결합니다.
- 서버 프로세스 상태 확인 - 서버 프로세스 상태를 모니터링하고 상태 확인에 실패한 프로세스를 자동으로 교체합니다.
- 원격 액세스 - AWS Systems Manager Session Manager를 사용하여 직접 문제 해결을 위해 플릿 인스턴스에 연결합니다.
- 별칭 관리 - 별칭을 사용하여 클라이언트 구성을 변경하지 않고 문제가 있는 플릿에서 플레이어 트래픽을 빠르게 리디렉션합니다.
- 백업 및 복구 - 중요한 게임 데이터 및 구성의 정기적인 백업을 구현하여 장애로부터 신속하게 복구할 수 있습니다.

일반적인 문제에 대한 명확한 인시던트 대응 절차와 런북을 설정하여 가동 중지 시간과 플레이어에 미치는 영향을 최소화합니다.

도구 및 통합

Amazon GameLift Servers는 다양한 AWS 서비스 및 타사 도구와 통합되어 프로덕션 관리 기능을 제공합니다.

- AWS CloudFormation - GameLift 리소스를 일관된 배포를 위한 코드형 인프라로 정의하고 관리합니다.
- AWS Lambda - 서버리스 함수를 생성하여 일상적인 관리 작업을 자동화하고 이벤트에 응답합니다.
- Amazon EventBridge - Amazon GameLift Servers 환경의 변경 사항에 자동으로 응답하는 이벤트 기반 아키텍처를 구축합니다.
- AWS SDK 통합 - AWS SDKs 사용하여 특정 요구 사항에 맞는 사용자 지정 관리 도구 및 대시보드를 구축합니다.
- 타사의 모니터링 - CloudWatch 지표 내보내기를 사용하여 타사 모니터링 및 분석 플랫폼과 통합합니다.

이러한 통합을 활용하여 팀의 워크플로 및 요구 사항에 맞는 포괄적인 관리 솔루션을 생성합니다.

관련 리소스

- [Amazon CloudWatch로 Amazon GameLift 모니터링](#)
- [Amazon GameLift 플릿 업데이트](#)
- [Amazon GameLift를 사용하여 게임 호스팅 용량 확장](#)
- [Amazon GameLift Server SDK 참조](#)

Amazon GameLift Servers 주요 용어

이 주제에서는 Amazon GameLift Servers 설명서 및 서비스에 사용되는 주요 기술 용어에 대한 정의를 제공합니다.

활성 게임 세션

현재 실행 중이며 플레이어를 수락하거나 호스팅하는 게임 세션입니다. 플릿 사용률 및 용량 계획을 모니터링하는 지표로 사용됩니다. [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#)(를) 참조하세요.

Alias

게임 클라이언트와 플릿 리소스 간에 추상화 계층을 제공하는 플릿의 기억하기 쉬운 식별자입니다. 별칭을 사용하면 게임 클라이언트 코드를 변경하지 않고도 플릿 구성을 업데이트할 수 있습니다. [별칭을 사용하여 Amazon GameLift Servers 플릿 지정 추상화](#)을(를) 참조하세요.

Amazon Machine Image(AMI)

관리형 플릿용 Amazon EC2 인스턴스를 생성하는 데 사용되는 사전 구성된 가상 머신 이미지입니다. 게임 서버를 실행하는 데 필요한 운영 체제, 런타임 라이브러리, Amazon GameLift Servers 에 이전트 소프트웨어가 포함되어 있습니다. [Amazon GameLift Servers 관리형 EC2 플릿](#)을(를) 참조하세요.

Anywhere 플릿

Amazon GameLift Servers 외부에서 관리되는 고객 제공 컴퓨팅 리소스를 사용하는 플릿 유형입니다. Anywhere 플릿을 사용하면 기존 인프라 또는 하이브리드 클라우드 솔루션을 Amazon GameLift Servers 게임 세션 관리와 통합할 수 있습니다. [Amazon GameLift Servers Anywhere 플릿](#)을(를) 참조하세요.

인증 토큰

Anywhere 컴퓨팅 리소스에서 실행되는 게임 서버 프로세스를 인증하는 Amazon GameLift Servers 에서 생성된 보안 자격 증명입니다. 서버 프로세스가 Amazon GameLift Servers 서비스와 통신하는 데 필요합니다. [Amazon GameLift Servers Anywhere 플릿](#)을(를) 참조하세요.

백엔드 서비스

게임 세션 배치 요청에 대한 통신을 제어하고 게임 세션 정보를 검색하는 게임 클라이언트와 Amazon GameLift Servers 서비스 간의 조정 계층입니다. 악의적인 공격으로부터 호스팅 리소스를 보호하는 안전한 중개자 역할을 합니다. [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#)을(를) 참조하세요.

게임 서버 빌드

Amazon GameLift Servers 호스팅 리소스에서 게임 서버를 실행하는 데 필요한 모든 파일을 포함하는 게임 서버 소프트웨어의 패키지 버전입니다. 게임 서버 실행 파일, 종속성, 자산을 포함합니다. [배포를 위해 게임 서버 빌드 패키징](#)을(를) 참조하세요.

컴퓨팅

컨테이너 플릿에서 플릿 인스턴스에 있는 게임 서버 컨테이너 그룹의 사본을 나타냅니다. 각 컴퓨팅은 한 번에 하나의 게임 세션을 호스팅할 수 있습니다. [Amazon GameLift Servers의 컨테이너 작동 방식](#)을(를) 참조하세요.

컨테이너 플릿

기존 빌드 대신 컨테이너 이미지를 사용하여 컨테이너화된 게임 서버를 실행하는 관리형 플릿의 한 유형입니다. 특정 런타임 환경 및 종속성을 사용하여 게임 서버를 배포할 수 있는 유연성을 제공합니다. [Amazon GameLift Servers 관리형 컨테이너 플릿](#)을(를) 참조하세요.

컨테이너 그룹

게임 서버 기능을 제공하기 위해 함께 작동하는 컨테이너 모음입니다. 컨테이너 작업 또는 포드와 마찬가지로 컨테이너 플릿 내에서 컨테이너의 동작 방식, 종속성, 리소스 공유를 정의합니다. [Amazon GameLift Servers의 컨테이너 작동 방식](#)을(를) 참조하세요.

Destination

게임 세션 대기열에서 게임 세션 배치 요청을 수신할 수 있는 플릿 또는 별칭입니다. 대기열에는 우선순위 및 지연 시간 정책이 서로 다른 여러 대상이 있을 수 있습니다. [게임 세션 대기열 사용자 지정](#)을(를) 참조하세요.

플릿

게임 서버를 실행하는 컴퓨팅 리소스 컬렉션입니다. 플릿 리소스는 여러 지리적 위치에 분산될 수 있으며 각 리소스는 하나 이상의 게임 서버 프로세스를 실행합니다. [Amazon GameLift Servers용 호스팅 플릿 배포](#)을(를) 참조하세요.

플릿 위치

플릿 인스턴스가 배포되는 특정 AWS 리전 또는 로컬 영역입니다. 다중 위치 플릿은 여러 위치에 걸쳐 지연 시간을 줄이고 가용성을 개선할 수 있습니다. [Amazon GameLift Servers용 호스팅 플릿 배포](#)을(를) 참조하세요.

FlexMatch

구성 가능한 규칙에 따라 플레이어 매치를 자동으로 구성하고 완료된 매치에 대한 게임 세션 배치를 요청하는 Amazon GameLift Servers 매치메이킹 서비스입니다. 스킬 기반 매칭, 팀 밸런싱, 매치 채우기를 지원합니다. [GameLift FlexMatch는 무엇인가요?](#)를 참조하세요.

게임 클라이언트

플레이어의 디바이스에서 실행되는 게임 소프트웨어입니다. 백엔드 서비스와 통신하여 게임 세션에 플레이어 배치를 시작하고 게임 플레이를 위해 게임 서버에 직접 연결합니다. [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#)을(를) 참조하세요.

게임 속성

게임 세션과 연결된 사용자 지정 데이터의 키-값 페어입니다. 구성 정보, 게임 규칙 또는 기타 메타 데이터를 게임 서버 및 클라이언트에 전달하는 데 사용됩니다. API 참조의 [GameProperty](#)를 참조하세요.

게임 서버

게임 상태를 관리하고, 플레이어 작업을 처리하며, 연결된 여러 플레이어에서 게임 플레이를 동기화하는 사용자 지정 서버 소프트웨어입니다. Amazon GameLift Servers Server SDK와 통합하여 해당 서비스와 통신합니다. [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)을(를) 참조하세요.

게임 서버 활성화

호스팅 리소스에서 게임 서버 프로세스를 시작하고 초기화하는 프로세스입니다. 게임 서버 실행 파일 로드, Amazon GameLift Servers와의 통신 설정, 게임 세션 호스팅 준비를 포함합니다. [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)을(를) 참조하세요.

게임 서버 컨테이너

게임 서버 애플리케이션을 실행하는 게임 서버 컨테이너 그룹 내의 컨테이너입니다. 컨테이너 그룹 수명 주기에 필수적인 것으로 자동 지정됩니다. [Amazon GameLift Servers의 컨테이너 작동 방식을](#) (를) 참조하세요.

게임 서버 컨테이너 그룹

게임 서버 애플리케이션을 실행하고 소프트웨어를 지원하는 컨테이너를 관리하는 컨테이너 그룹 유형입니다. 컨테이너 플릿이 게임 세션을 호스팅하는 데 필요하며 플릿 인스턴스 간에 복제할 수 있습니다. [Amazon GameLift Servers의 컨테이너 작동 방식을](#) (를) 참조하세요.

게임 서버 프로세스

호스팅 리소스에서 실행하는 게임 서버 실행 파일의 인스턴스입니다. 각 프로세스는 한 번에 하나의 게임 세션을 호스팅하고 Amazon GameLift Servers와 직접 통신하여 상태를 보고하고 지침을 받을 수 있습니다. [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)을(를) 참조하세요.

게임 세션

게임 서버 프로세스에서 실행되는 멀티플레이어 게임의 인스턴스입니다. 여러 플레이어가 함께 조인하여 참여할 수 있는 단일 게임 플레이 경험을 의미합니다. API 참조의 [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#) 및 [GameSession](#)을 참조하세요.

게임 세션 데이터

플레이어 데이터, 게임 상태, 세션 속성, 게임 서버 및 클라이언트에서 사용하는 기타 런타임 정보를 포함한 게임 세션과 관련된 정보입니다. API 참조의 [GameSession](#)을 참조하세요.

게임 세션 배치

새 게임 세션을 호스팅하기 위해 사용 가능한 게임 서버를 찾는 프로세스입니다. 게임 서버 가용성에 대한 실시간 정보를 사용하여 비용 및 지연 시간과 같은 요인을 기반으로 최적의 배치를 결정합니다. API 참조의 [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 및 [StartGameSessionPlacement](#)를 참조하세요.

게임 세션 보호

이벤트 또는 플릿 업데이트 규모 조정 중에 Amazon GameLift Servers가 활성 게임 세션을 호스팅하는 게임 서버 프로세스를 종료하지 못하도록 하는 설정입니다. 진행 중인 게임이 중단되지 않도록 합니다. [Amazon GameLift Servers 플릿의 용량을 수동으로 설정](#)(를) 참조하세요.

게임 세션 대기열

구성 가능한 알고리즘을 사용하여 최적의 결과를 위해 게임 세션을 배치하는 배치 메커니즘입니다. 여러 지리적 위치에 걸쳐 호스팅 비용 및 플레이어 지연 시간 최소화와 같은 요인에 따라 배치의 우선순위를 정할 수 있습니다. [게임 세션 배치 구성](#)(를) 참조하세요.

유휴 인스턴스

실행 중이지만 현재 활성 게임 세션을 호스팅하지 않는 플릿 인스턴스입니다. 플릿 효율성 모니터링 및 규모 조정 결정을 위한 지표로 사용됩니다. [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#)(를) 참조하세요.

인바운드 권한

플릿 인스턴스에 연결하는 데 사용할 수 있는 IP 주소 및 포트 범위를 정의하는 네트워크 액세스 규칙입니다. 게임 서버에 대한 액세스를 제어하고 보안 연결을 보장합니다. 관리형 EC2 및 관리형 컨테이너 플릿 모두 인바운드 권한 구성이 필요합니다.

[IpPermission](#) 및 [네트워크 연결 구성](#) 섹션을 참조하세요.

지연 시간 정책

플레이어 연결에 허용되는 지연 시간 임계값을 정의하는 게임 세션 대기열의 규칙입니다. 플레이어가 허용 가능한 네트워크 성능으로 게임 세션에 배치하는 데 사용됩니다. [게임 세션 대기열 사용자 지정](#)(를) 참조하세요.

관리형 플릿

Amazon GameLift Servers가 AWS 클라우드에서 컴퓨팅 리소스를 배포하고 관리하는 플릿 유형입니다. 자동 조정, 상태 모니터링, 인프라 관리를 제공합니다. [Amazon GameLift Servers 관리형 EC2 플릿](#)을(를) 참조하세요.

매치메이킹

스킬 수준, 지연 시간 또는 기타 게임별 요인과 같은 기준에 따라 멀티플레이어 게임 플레이를 위해 플레이어들을 그룹화하는 프로세스입니다. Amazon GameLift Servers에서 주로 FlexMatch에서 처리합니다. [GameLift FlexMatch는 무엇인가요?](#)를 참조하세요.

지표 그룹

모니터링 및 자동 규모 조정 결정에 사용되는 관련 성능 지표의 모음입니다. 활성 게임 세션, 사용 가능한 게임 서버, 플레이어 사용률과 같은 지표를 그룹화합니다. [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#)을(를) 참조하세요.

인스턴스별 컨테이너 그룹

각 플릿 인스턴스에서 추가 소프트웨어를 실행하는 선택적 컨테이너 그룹 유형입니다. 인스턴스당 하나의 사본만 배포하여 백그라운드 서비스 또는 모니터링 도구에 유용합니다. [Amazon GameLift Servers의 컨테이너 작동 방식](#)을(를) 참조하세요.

플레이어 세션

게임 세션에서 플레이어의 연결 슬롯입니다. 게임 세션 슬롯을 예약하고, 플레이어가 연결될 때 플레이어를 검증하며, 플레이어 연결 상태 및 연결 해제를 추적하는 데 사용됩니다. API 참조의 [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 및 [PlayerSession](#)을 참조하세요.

우선순위 구성

게임 세션 배치 대상을 선택할 때 게임 세션 대기열의 우선순위 지정 순서를 사용자 지정하는 설정입니다. 개별 배치 요청에 대한 우선순위를 수정하는 우선순위 재정의 목록과 달리 전체 대기열 동작을 정의합니다. [게임 세션 배치 우선순위](#)을(를) 참조하세요.

우선순위 재정의 목록

게임 세션 배치 요청에 포함할 수 있는 대체 우선순위 위치 목록입니다. 다른 요청에 영향을 주지 않고 한 요청의 위치에 대해서만 대기열의 구성된 우선순위를 효과적으로 대체합니다. [게임 세션 배치 우선순위](#)을(를) 참조하세요.

프로세스 관리자

컨테이너 또는 인스턴스 내에서 여러 게임 서버 프로세스의 수명 주기를 관리하는 구성 요소입니다. 개별 게임 서버 프로세스의 시작, 중지, 모니터링을 처리합니다. [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)을(를) 참조하세요.

실시간 스크립트

Amazon GameLift Servers Realtime 서버에 대한 게임 로직을 정의하는 JavaScript 코드입니다. 사용자 지정 게임 서버 없이 플레이어 연결, 메시지 라우팅, 게임 상태 관리를 처리합니다. [Amazon GameLift Realtime 서버란 무엇인가요?](#)를 참조하세요.

라우팅 전략

별칭이 플릿으로 트래픽을 보내는 방법을 결정하는 구성입니다. 정의된 규칙에 따라 특정 플릿으로 라우팅하거나 여러 플릿에 트래픽을 분산할 수 있습니다. [별칭을 사용하여 Amazon GameLift Servers 플릿 지정 추상화](#)을(를) 참조하세요.

런타임 구성

동시 프로세스 수, 시작 파라미터, 리소스 할당을 포함하여 게임 서버 프로세스가 플릿 인스턴스에서 실행되는 방법을 정의하는 설정입니다. [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)을(를) 참조하세요.

스케일링

플레이어 수요에 따라 호스팅 리소스를 추가하거나 제거하여 플릿 용량을 조정하는 프로세스입니다. 오토(자동, 사용률 지표 기반) 또는 매뉴얼(수동, 관리자 제어)일 수 있습니다. [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#)을(를) 참조하세요.

조정 정책

플릿 용량을 자동으로 조정하는 시기와 방법을 정의하는 규칙입니다. 대상 기반(특정 사용률 유지) 또는 규칙 기반(지표 임계값을 기반으로 한 규모 조정)일 수 있습니다. [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#)을(를) 참조하세요.

Server SDK

게임 서버를 Amazon GameLift Servers와 통합하는 데 필요한 도구와 라이브러리를 제공하는 소프트웨어 개발 키트입니다. 세션 관리, 상태 보고, 상태 업데이트를 위해 게임 서버와 서비스 간의 통신을 활성화합니다. [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)을(를) 참조하세요.

세션 기반 멀티플레이어

게임 플레이가 시간 제한이 있는 개별 세션 내에서 이루어지는 멀티플레이어 게임 아키텍처입니다. 각 세션에는 정의된 시작 및 종료가 있으며, 플레이어는 필요에 따라 참여하고 종료할 수 있습니다. [Amazon GameLift Servers를 사용한 호스팅 작동 방식을\(를\)](#) 참조하세요.

스팟 인스턴스

저렴한 비용으로 예비 AWS 용량을 사용하는 Amazon EC2 인스턴스의 한 유형입니다. 에 용량이 다시 AWS 필요할 때 스팟 인스턴스를 중단할 수 있으므로 내결함성 게임 워크로드에 적합합니다. [스팟 인스턴스를 위한 대기열 빌드을\(를\)](#) 참조하세요.

TLS 인증서

게임 클라이언트와 서버 간에 암호화된 통신을 활성화하는 데 사용되는 디지털 인증서입니다. TLS 지원 플릿이 안전한 데이터 전송을 보장하는 데 필요합니다. [Amazon GameLift Servers 관리형 EC2 플릿 생성을\(를\)](#) 참조하세요.

Amazon GameLift Servers 시작

시작하기 리소스를 활용하여 Amazon GameLift Servers 서비스와 세션 기반 멀티플레이어 게임을 위한 사용자 지정 호스팅 솔루션 개발을 시작하는 방법에 대해 자세히 알아봅니다.

시작하기 전에

- 와 함께 사용할 생성 AWS 계정 (또는 기존을 지정)합니다 Amazon GameLift Servers.
- Amazon GameLift Servers 및 관련 AWS 서비스에 대한 권한이 있는 사용자를 설정합니다.
- 작업 AWS 리전 할을 선택합니다. 개발에서 위치와 가까운 리전을 선택합니다. 리전은 언제든지 변경할 수 있습니다.

[AWS 사용자 계정 설정](#)

경로 선택

Amazon GameLift Servers 빠른 탐색

가장 적합한 용도: 학습, 개념 증명 생성, 신속한 프로토타이핑

빠른 시작 옵션:

- [Amazon GameLift Servers용 게임 서버 래퍼](#) - [자습서: Amazon GameLift Servers 래퍼를 사용한 빠른 온보딩](#) 섹션을 참조하세요. 이 도구는 게임 코드를 변경할 필요없이 게임 서버를 호스팅하고 Amazon GameLift Servers로 게임 세션을 실행하는 가장 빠르고 쉬운 방법입니다. 게임 서버 래퍼는 기본 게임 세션 관리 기능과 간소화된 게임 서버 배포를 제공합니다. 자체 게임 프로젝트 또는 샘플 프로젝트 사용에 대한 Amazon GameLift Servers의 실습 평가를 수행하는 데 적합합니다. 사용자 지정 게임 호스팅 솔루션을 구축할 준비가 되면 Amazon GameLift Servers용 서버 SDK와 완전히 통합된 사용자 지정 개발 옵션 중 하나로 전환합니다. 게임에 사용자 지정 호스팅 솔루션이 필요하지 않은 경우 게임 서버 래퍼를 계속해서 사용하여 프로덕션에서 게임 서버를 배포하고 호스팅할 수 있습니다.
- [Unreal Engine 또는 Unity용 Amazon GameLift Servers 플러그인](#) - 플러그인은 GUI 워크플로와 샘플 자산을 제공하여 초기 단계를 안내하고 기본 호스팅 솔루션으로 게임 서버를 배포합니다. 플러그인을 사용하여 자체 관리형 Anywhere 플릿으로 호스팅을 설정하거나 클라우드 기반의 관리형 EC2 플릿 또는 컨테이너 플릿을 배포합니다. 사용자 지정 호스팅 솔루션을 개발할 준비가 되면 플러그인 구축 솔루션을 기반으로 빌드할 수 있습니다.

- [Amazon GameLift Servers 관리형 컨테이너용 스타터 키트](#) - 이 키트는 게임 서버를 통합하고, 게임 서버 컨테이너 이미지를 준비하며, 호스팅용 컨테이너 플릿을 배포하는 작업을 간소화합니다. 통합을 위해 이 키트는 게임 서버에 필수 게임 세션 관리 기능을 추가합니다. 이 키트는 사전 구성된 템플릿을 사용하여 컨테이너 플릿과 게임 서버용 자동 배포 파이프라인을 빌드합니다. 전체 게임 세션 관리 기능을 추가할 준비가 되면 사용자 지정 개발 로드맵 중 하나에 따라 Amazon GameLift Servers 용 서버 SDK를 통합합니다.

사용자 지정 호스팅 솔루션 구축

가장 적합한 용도: 사용자 지정 게임 서버를 위한 프로덕션 수준 솔루션 생성

[Amazon GameLift Servers 게임 호스팅 옵션](#) 페이지를 검토하고 솔루션 옵션 중 하나를 선택합니다. 다음 개발 로드맵 중 하나를 따라 게임을 위해 전체 기능을 갖춘 사용자 지정 호스팅 솔루션을 구축하기 시작합니다. 로드맵은 호스팅 솔루션에서 각 구성 요소를 생성, 테스트, 사용자 지정하는 방법에 대한 자세한 지침을 제공합니다.

- [Amazon GameLift Servers 관리형 EC2를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers Anywhere 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers를 사용한 하이브리드 호스팅을 위한 개발 로드맵](#)

권장 학습 경로

1. 가볍게 시작하세요. 게임 서버 래퍼 또는 Unreal용 플러그인을 사용합니다.
2. 개념을 이해하세요. [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#) 페이지를 검토합니다.
3. 아키텍처를 선택하세요. 게임 요구 사항에 따라 호스팅 모델을 선택합니다.
4. 빌드하고 테스트하세요. 적절한 개발 로드맵을 따릅니다. 각 구성 요소의 기본 버전을 빌드한 다음 반복하고 사용자 지정합니다.
5. 확장하고 최적화하세요. 프로덕션 수준 사용에 맞게 솔루션을 최적화합니다. 매치메이킹과 같은 고급 기능을 추가합니다.

AWS 사용자 계정 설정

Tip

다음 주제를 사용하여 이러한 작업에 대한 도움을 받으세요.

- AWS 계정에서 사용할 새를 가져옵니다 Amazon GameLift Servers.
- Amazon GameLift Servers 리소스 작업 권한이 있는 사용자 또는 그룹을 생성합니다.
- 보안 자격 증명 설정(Unreal 및 Unity용 AWS CLI 도구 및 Amazon GameLift Servers 플러그인을 사용하려면 필요)
-

모든 AWS 서비스와 마찬가지로 Amazon GameLift Servers 서비스 및 도구를 AWS 계정 사용하려면 필요합니다. AWS 계정은 두 가지 기본 함수를 제공합니다. 즉, (1) 계정으로 생성한 모든 AWS 리소스에 대한 컨테이너를 제공하고, (2) 사용자 인증 설정 및 사용자 액세스 권한 제어를 포함하여 AWS 리소스에 대한 보안을 관리할 수 있습니다. 를 생성하는 데는 비용이 들지 않습니다 AWS 계정.

AWS 계정을 Amazon GameLift Servers 사용하거나 사용하지 않고 탐색

다음과 같은 경우에는 AWS 계정이 필요하지 않습니다.

- [AWS for Games](#)에서 게임 경험을 구축, 실행 및 확장하기 위한 AWS 도구를 알아봅니다. [블로그](#)를 읽고 [게임용 솔루션 라이브러리](#)를 찾아보세요.
- [제품 개요, FAQ 및 리소스](#)에서 Amazon GameLift Servers에 대해 자세히 알아보세요. AWS에게 문의하여 제품 질문에 대한 답변을 찾습니다. (다음 시도: “저비용 옵션을 찾아 내 멀티플레이어 게임을 호스팅하세요.”)
- 자세한 내용은 호스팅 및 매치메이킹을 위한 개발자 가이드와 서비스 API 참조 가이드를 포함한 [기술 설명서](#)에서 Amazon GameLift Servers가 어떻게 작동하는지 알아보세요.
- [Amazon GameLift Servers 요금](#) 및 비용 최적화 기법에 대한 정보를 확인하세요. [요금 계산기](#)를 사용하여 최대 동시 플레이어 사용량(CCU)을 기준으로 호스팅 비용을 계산하는 방법을 확인하세요.
- Amazon GameLift Servers SDK, 플러그인, 툴킷용 코드 리포지토리를 다운로드하고 확인합니다. [Amazon GameLift Servers 시작하기](#)를 참조하세요. (사용 AWS 계정 하려면이 필요합니다.)

다음을 수행하려면 AWS 계정이 필요합니다.

- Unreal 및 Unity용 Amazon GameLift Servers 플러그인으로 온보딩 워크플로를 따르거나 게임 서버 래퍼를 사용합니다.
- AWS Management Console를 사용하여 AWS 리소스를 생성하고 관리합니다.
- AWS Command Line Interface를 사용하여 AWS 리소스를 생성하고 관리합니다.
- Amazon GameLift Servers 기술 설명서의 Amazon Q를 사용하여 답변, 지침, 권장 사항을 찾을 수 있습니다.

주제

- [에 가입 AWS 계정](#)
- [관리자 액세스 권한이 있는 사용자 생성](#)
- [Amazon GameLift Servers의 사용자 권한 설정](#)
- [사용자의 프로그래밍 방식 액세스 설정](#)
- [게임의 프로그래밍 방식 액세스 설정](#)
- [Amazon GameLift Servers의 IAM 권한 예제](#)
- [Amazon GameLift Servers의 IAM 서비스 역할 설정](#)

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. <https://portal.aws.amazon.com/billing/signup>을 엽니다.
2. 온라인 지시 사항을 따르세요.

등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

에 가입하면 AWS 계정AWS 계정 루트 사용자의 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 [루트 사용자 액세스 권한이 필요한 작업](#)을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 <https://aws.amazon.com/>으로 이동하고 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자 활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자 [AWS Management Console](#)로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In 사용 설명서의 [루트 사용자 로 로그인](#)을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

지침은 IAM 사용 설명서의 [AWS 계정 루트 사용자\(콘솔\)에 대한 가상 MFA 디바이스 활성화를 참조하세요.](#)

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [AWS IAM Identity Center 설정](#)을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리자 액세스 권한을 부여합니다.

를 자격 증명 소스 IAM Identity Center 디렉터리로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 [기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.](#)

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 포털에 로그인](#)을 참조하세요. AWS Sign-In

추가 사용자에게 액세스 권한 할당

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

지침은 AWS IAM Identity Center 사용 설명서의 [Create a permission set](#)를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [그룹 추가](#)를 참조하세요.

Amazon GameLift Servers의 사용자 권한 설정

Amazon GameLift Servers 리소스에 필요한 경우 추가 사용자를 생성하거나 액세스 권한을 기존 사용자로 확장합니다. 모범 사례([IAM의 보안 모범 사례](#))로써 모든 사용자에게 최소 권한 권한을 적용합니다. 권한 구문에 대한 지침은 [Amazon GameLift Servers의 IAM 권한 예제](#)를 참조하세요.

다음 지침에 따라 AWS 계정의 사용자를 관리하는 방법에 따라 사용자 권한을 설정합니다.

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 [권한 세트 생성](#)의 지침을 따릅니다.

- ID 제공업체를 통해 IAM에서 관리되는 사용자:

ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 [Create a role for a third-party identity provider \(federation\)](#)의 지침을 따릅니다.

- IAM 사용자:

- 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 [Create a role for an IAM user](#)의 지침을 따릅니다.
- (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 [사용자\(콘솔\)에 권한 추가](#)의 지침을 따르세요.

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

사용자의 프로그래밍 방식 액세스 설정

사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 AWS Management Console. 프로그래밍 방식 액세스를 부여하는 방법에는 액세스하는 사용자 유형에 따라 다릅니다 AWS.

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

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장) 콘솔 자격 증명을 임시 자격 증명으로 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. - 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS 로컬 개발을 위한 로그인을 참조하세요. - AWS SDKs 경우 SDK 및 도구 참조 안내서의 AWS 로컬 개발을 위한 로그인을 참조하세요. AWS SDKs
작업 인력 ID (IAM Identity Center에서 관리되는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. - 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center 참조하세요. - AWS SDKs, 도구 및 AWS APIs 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWS SDKs
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용 의 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. - 자세한 AWS CLI 내용은 사용 AWS Command Line Interface 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요. - AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWS SDKs - AWS APIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

액세스 키를 사용하는 경우 [AWS 액세스 키 관리 모범 사례를 참조하세요](#).

게임의 프로그래밍 방식 액세스 설정

대부분의 게임은 백엔드 서비스를 사용하여 SDK를 Amazon GameLift Servers 사용하여와 통신합니다. AWS SDKs 게임 클라이언트로 작동하는 백엔드 서비스를 사용하여 게임 세션을 요청하고 플레이어를 게임에 참여시키는 등의 작업을 요청합니다. 이러한 서비스에는 Amazon GameLift Servers용 서비스 API에 대한 호출을 인증하기 위한 프로그래밍 액세스 및 보안 자격 증명이 필요합니다.

의 Amazon GameLift Servers 경우 AWS Identity and Access Management (IAM)에서 플레이어 사용자를 생성하여 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 Amazon GameLift Servers에 요청하기 전에 이 역할을 맡는 코드를 포함해야 합니다. 보안 모범 사례에 따라 역할은 제한된 임시 액세스를 제공합니다. AWS 리소스

([IAM 역할](#)) 또는 외부 AWS ([IAM Roles Anywhere](#))에서 실행되는 워크로드에 역할을 사용할 수 있습니다.

- 플레이어 사용자 권한을 사용하여 IAM 사용자 그룹을 생성하고 플레이어 사용자를 그룹에 추가합니다. 이 옵션은 플레이어 사용자에게 장기간 자격 증명을 제공하며, 이 자격 증명은 백엔드 서비스가 Amazon GameLift Servers와 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 인증](#)을 참조하세요.

Amazon GameLift Servers의 IAM 권한 예제

이 예제의 구문을 사용하여 Amazon GameLift Servers 리소스에 액세스해야 하는 사용자의 AWS Identity and Access Management (IAM) 권한을 설정합니다. 사용자 권한 관리에 대한 자세한 내용은 [Amazon GameLift Servers의 사용자 권한 설정](#) 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 Servers FleetIQ를 독립 실행형 솔루션으로 사용하는 경우 [AWS 계정 용 설정을 참조하세요Amazon GameLift Servers FleetIQ](#).

관리 권한 예제

이 예제에서는 호스팅 관리자 또는 개발자에게 Amazon GameLift Servers 게임 호스팅 리소스를 관리할 수 있는 대상 액세스 권한을 부여합니다.

Example Amazon GameLift Servers 전체 액세스 리소스 권한의 구문

다음과 같은 예시는 모든 Amazon GameLift Servers 리소스에 대한 모든 액세스를 확장합니다.

JSON

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

```

    "Resource": "*"
  }
}

```

Example 기본적으로 활성화되지 않은 리전을 지원하는 Amazon GameLift Servers 리소스 권한 구문

다음 예제에서는 기본적으로 활성화되지 않은 모든 Amazon GameLift Servers 리소스 및 AWS 리전에 대한 액세스를 확장합니다. 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 [AWS 리전관리](#)를 참조하세요.

JSON

```

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

```

Example Amazon ECR에서 컨테이너 이미지에 액세스하기 위한 Amazon GameLift Servers 리소스 구문

다음 예시에서는 Amazon GameLift Servers 사용자가 관리형 컨테이너 플릿으로 작업할 때 필요한 Amazon Elastic Container Registry(Amazon ECR) 작업에 대한 액세스를 확장합니다.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ecr:DescribeImages",
      "ecr:BatchGetImage",

```



```

    "ecr:GetDownloadUrlForLayer"
  ],
  "Resource": "*"
}
}

```

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

다음 예제는 모든 Amazon GameLift Servers 리소스에 대한 액세스를 확장하고 사용자가 IAM 서비스 역할을 Amazon GameLift Servers에 전달할 수 있도록 허용합니다. [Amazon GameLift Servers의 IAM 서비스 역할 설정](#)에 설명된 대로 Amazon GameLift Servers는 서비스 역할을 사용하여 사용자를 대신하여 다른 리소스 및 서비스에 제한적으로 액세스할 수 있습니다. 예를 들어, CreateBuild 요청에 응답할 때 Amazon GameLift Servers는 Amazon S3 버킷의 빌드 파일에 액세스할 수 있어야 합니다. PassRole 작업에 대한 자세한 내용은 [IAM 사용 설명서의 IAM: 특정 AWS 서비스에 IAM 역할 전달](#)을 참조하세요.

JSON

```

{
  "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 Servers 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.

Example게임 세션 배치 권한의 구문

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

JSON

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

Example매치메이킹 권한의 구문

다음 예제에서는 FlexMatch 매치메이킹 활동을 관리하는 Amazon GameLift Servers API에 대한 액세스를 확장합니다. FlexMatch는 신규 또는 기존 게임 세션에 대해 플레이어를 매칭하고 Amazon GameLift Servers에서 호스팅되는 게임에 대해 게임 세션 배치를 시작합니다. FlexMatch에 대한 자세한 정보는 [Amazon GameLift Servers FlexMatch란 무엇입니까?](#)를 참조하세요.

JSON

```
{
```

```

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

```

Example게임 세션 배치 권한의 구문

다음 예제는 특정 플릿에서 게임 세션과 플레이어 세션을 수동으로 생성하는 Amazon GameLift Servers 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.

JSON

```

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

```

Amazon GameLift Servers의 IAM 서비스 역할 설정

일부 Amazon GameLift Servers 기능을 사용하려면 소유한 다른 AWS 리소스에 대한 제한된 액세스를 확장해야 합니다. AWS Identity and Access Management (IAM) 역할을 생성하여이 작업을 수행할 수 있습니다. [IAM 역할](#)은 계정에 생성할 수 있는, 특정 권한을 지닌 IAM 자격 증명입니다. IAM 역할은 자격 AWS 증명이 할 수 있는 것과 없는 것을 결정하는 권한 정책이 있는 자격 증명이라는 점에서 IAM 사용자와 유사합니다 AWS. 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 Servers 관리형 플릿에서 사용할 수 있는 역할을 생성하는 방법에 대해 알아봅니다. Amazon GameLift Servers FleetIQ를 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 게임 호스팅을 최적화하는 경우 [AWS 계정 용 설정을 Amazon GameLift Servers FleetIQ](#)참조하세요.

다음 절차에서 Amazon GameLift Servers가 역할을 수임하도록 허용하는 트러스트 정책과 사용자 지정 권한이 있는 역할을 생성합니다.

Amazon GameLift Servers 관리형 EC2 플릿에 대한 IAM 서비스 역할 생성

1단계: 권한 정책 생성.

이 페이지의 지침과 예제를 사용하여 작업 중인 Amazon GameLift Servers 플릿 유형에 대한 사용자 지정 권한 정책을 생성합니다.

JSON 정책 편집기를 사용하여 정책을 생성하는 방법

1. 에 로그인 AWS Management Console 하고 <https://console.aws.amazon.com/iam/> IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.

정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작을 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 참조](#)를 참조하세요.
6. [정책 검증](#) 중에 생성되는 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 사용자 안내서의 [정책 재구성을 참조](#)하세요.

7. (선택 사항)에서 정책을 생성하거나 편집할 때 템플릿에 사용할 AWS Management Console 수 있는 JSON 또는 YAML 정책 CloudFormation 템플릿을 생성할 수 있습니다.

이렇게 하려면 정책 편집기에서 작업을 선택한 다음, CloudFormation 템플릿 생성을 선택합니다. 자세한 CloudFormation 내용은 AWS CloudFormation 사용 설명서의 [AWS Identity and Access Management 리소스 유형 참조](#)를 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 태그를 사용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [AWS Identity and Access Management 리소스에 대한 태그를 참조](#)하세요.
11. 정책 생성을 선택하고 새로운 정책을 저장합니다.

2단계: Amazon GameLift Servers가 맡을 수 있는 역할 생성.

IAM 역할을 생성하려면

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.

JSON

```
{
  "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.

Amazon GameLift Servers 관리형 컨테이너에 대한 IAM 역할 생성

Amazon GameLift Servers 관리형 컨테이너를 사용하는 경우 컨테이너 플릿과 함께 사용할 IAM 서비스 역할을 생성해야 합니다. 이 역할은 Amazon GameLift Servers가 컨테이너 플릿 리소스를 관리하고 사용자를 대신하여 작업을 수행하는 데 필요한 제한된 권한을 부여합니다.

컨테이너 플릿에 대한 IAM 역할 을 만들려면

1. 에 로그인 AWS Management Console 하고 <https://console.aws.amazon.com/iam/> IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 선택 페이지에서 AWS 서비스를 선택하고 사용 사례 'GameLift'를 선택합니다. 다음을 선택합니다.
4. 권한 추가에서 관리형 정책 GameLiftContainerFleetPolicy를 선택합니다. 다음을 선택합니다. 이 정책에 대한 자세한 내용은 [AWS 에 대한 관리형 정책 Amazon GameLift Servers](#) 섹션을 참조하세요.
5. 이름 지정, 검토 및 생성에서 역할 이름을 입력하고 역할 생성을 선택하여 새 역할을 저장합니다.

권한 정책 구문

- Amazon GameLift Servers가 서비스 역할을 맡을 수 있는 권한

JSON

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

- 기본적으로 활성화되지 않은 AWS 리전에 액세스할 수 있는 권한

JSON

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

Amazon GameLift Servers 개발 도구 가져오기

Amazon GameLift Servers는 게임용 게임 호스팅 솔루션을 구축하는 데 도움이 되는 SDK 및 기타 도구 세트를 제공합니다. SDK는 Amazon GameLift Servers 서비스와 상호 작용해야 하는 게임 서버, 게임 클라이언트 및 백엔드 서비스에 기능을 추가할 수 있습니다. Amazon GameLift Servers SDK 버전 및 호환성에 대한 최신 정보는 [Amazon GameLift Servers 릴리스 정보](#) 섹션을 참조하세요.

게임 서버

64비트 게임 서버를 Amazon GameLift Servers용 서버 SDK와 통합하고 빌드합니다. 게임 서버는 서버 SDK를 사용하여 게임 세션 시작, 업데이트, 중지를 포함한 게임 세션 관리를 위해 Amazon GameLift Servers 서비스와 통신합니다. 서버 SDK를 게임 프로젝트에 통합하는 방법에 대한 도움말은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#) 섹션을 참조하세요.

개발 지원

- 개발 OS
 - Windows
 - Linux
- 프로그래밍 언어

[Amazon GameLift Servers SDK를 가져옵니다](#). 버전별 정보와 설치 지침은 각 패키지에 포함된 Readme 파일을 참조하세요.

- [C++ 서버 SDK](#)
 - [Server SDK 참조](#)
 - [통합 방법](#)
- [C# 서버 SDK](#)(.NET 4, .NET 6, .NET 8용 지원은 버전에 따라 다름, [SDK 버전](#) 참조)
 - [Server SDK 참조](#)
 - [통합 방법](#)
- [Go 서버 SDK](#)
 - [Server SDK 참조](#)
 - [통합 방법](#)
- 게임 엔진 지원

용 전체 기능 플러그인에는 UI 워크플로 및 샘플 자산과 AWS SDK 및 서버 SDK의 내장 버전이 Amazon GameLift Servers 포함됩니다. 워크플로는 관리형 EC2 플릿, 관리형 컨테이너 플릿 또는 자

체 관리형 Anywhere 플릿으로 호스팅하기 위해 게임 서버를 구성하고 배포하는 방법을 안내합니다. 안내 워크플로가 필요하지 않은 경우 동일한 GitHub 리포지토리에서 게임 엔진용 서버 SDK를 가져올 수 있습니다.

플러그인이 지원하지 않는 다른 게임 엔진 또는 개발 환경을 사용하는 경우 프로그래밍 언어에 맞는 서버 SDK를 가져와 게임 프로젝트에 추가합니다.

버전별 정보와 설치 지침은 각 패키지에 포함된 Readme 파일을 참조하세요.

- [Unreal Engine용 플러그인](#) - Unreal Engine 버전 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6과 함께 사용하도록 설계되었습니다. 버전별 Readme 파일에서 Unreal 지원을 확인합니다.
 - [Unreal Engine용 플러그인 가이드](#)
 - [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
- [Unity용 플러그인](#) - Unity Editor 6.0, 2022.3 또는 2021.3의 LTS 버전과 함께 사용하도록 설계되었습니다. NET Standard 2.1 및 .NET 4.x가 있는 Unity의 .NET Framework 및 .NET Standard 프로 필을 지원합니다. Unity 지원을 위해 버전별 Readme 파일을 확인합니다.
 - [Unity용 플러그인 가이드](#)
 - [C# 서버 SDK 참조](#)
- [Unreal용 서버 SDK](#)
 - [Server SDK 참조](#)
 - [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#)
- [Unity용 서버 SDK](#)
 - [Server SDK 참조](#)
 - [Amazon GameLift Servers를 Unity 프로젝트에 통합](#)

런타임 지원

관리형 호스팅 솔루션의 경우 게임 서버를 구축하여 다음 Amazon Machine Image(AMI) 중 하나를 실행합니다. 자세한 내용은 Amazon GameLift Servers의 [Amazon GameLift Servers AMI 버전](#) 섹션을 참조하세요.

- [Windows Server 2016](#)
- [Amazon Linux 2023](#)
- [Amazon Linux 2](#)

Note

Amazon Linux 2(AL2)는 2026년 6월 30일에 지원이 종료됩니다. 자세한 내용은 [Amazon Linux 2 FAQ](#)를 참조하세요. AL2에서 호스팅되고 Amazon GameLift Servers 서버 SDK 4.x를 사용하는 게임 서버의 경우 먼저 게임 서버 빌드를 서버 SDK 5.x로 업데이트한 다음 AL2023 인스턴스에 배포합니다. [Amazon GameLift Servers용 서버 SDK 5.x로 마이그레이션](#)(을) 참조하세요.

추가 도구

[Amazon GameLift Servers용 게임 서버 래퍼](#)

이 도구를 사용하면 일련의 기본 게임 세션 관리 기능으로 호스팅할 게임 서버를 배포할 수 있습니다. 또한 이 도구를 사용할 때 게임 코드를 변경하거나 Amazon GameLift Servers용 서버 SDK를 통합할 필요가 없습니다. 게임 서버 래퍼를 사용하여 게임 서버를 패키징하고 세 가지 Amazon GameLift Servers 호스팅 솔루션(Anywhere, 관리형 EC2 또는 관리형 컨테이너) 중 하나를 사용하여 게임 호스팅을 위해 배포합니다. 이 도구는 게임 서버 사용자 지정을 지원하지 않으므로 자체 게임 또는 샘플 게임을 사용한 초기 평가 또는 프로토타입에 가장 적합합니다. 게임에 사용자 지정 기능이 필요하지 않은 경우 프로덕션 호스팅을 위해 게임 서버 래퍼를 사용하여 게임 서버를 배포할 수 있습니다.

[Amazon GameLift Servers 도구 키트](#)

Amazon GameLift Servers 도구 키트는 일반적인 시나리오와 문제가 있는 개발자를 지원하기 위해 개발한 스크립트 및 기타 도구 모음입니다. 도구 키트 자료에는 스크립트, 샘플 코드, Readme 파일이 포함됩니다.

- [컨테이너 스타터 키트](#) - 이 도구를 사용하여 Amazon GameLift Servers 관리형 컨테이너로 호스팅하기 위한 게임 서버 빌드를 설정하는 작업을 간소화합니다. 키트는 게임 서버에 필수 게임 세션 관리 기능을 통합하고, 사전 구성된 템플릿을 사용하여 컨테이너 플릿을 생성하고 게임 서버 빌드를 위한 자동 배포 파이프라인을 설정합니다. 배포 후 Amazon GameLift Servers 콘솔 및 API 도구를 사용하여 플릿 성능을 모니터링하고, 게임 세션을 관리하며, 지표를 분석합니다. 이 키트는 빌드 자동화, 스토리지 및 AWS CloudFormation 인프라 배포를 위해와 통합됩니다 AWS CodeBuild .
- [빠른 빌드 업데이트 도구](#) - 이 도구를 사용하여 관리형 EC2 플릿에 이미 배포된 게임 서버 빌드를 수정합니다. 이 도구는 변경할 때마다 새 EC2 플릿을 구성하고 생성할 필요 없이 게임 빌드 파일을 빠르게 교체할 수 있도록 설계되었습니다. 개별 인스턴스 또는 플릿의 모든 인스턴스를 업데이트할 수 있습니다. 옵션을 사용하면 특정 빌드 파일 또는 전체 빌드를 대체하고 업데이트 후 게임 서버를 다시 시작하는 방법을 관리할 수 있습니다.

게임 클라이언트의 경우

게임 클라이언트를 용 클라이언트 SDK와 통합합니다Amazon GameLift Servers. 클라이언트 SDK는 게임 클라이언트가 플레이어 게이트웨이 및 UDP ping 비컨과 같은 Amazon GameLift Servers 기능을 사용할 수 있는 유틸리티를 제공합니다.

클라이언트 SDK 가져오기

- [C++용 클라이언트 SDK](#) - 내장 엔드포인트 선택 알고리즘이 있는 플레이어 게이트웨이 및 UDP ping 비컨 유틸리티와 통합을 보여주는 샘플 애플리케이션을 포함합니다. C++17 및 스레드 라이브러리가 필요합니다.
- [Unreal Engine용 클라이언트 SDK](#) - 플레이어 게이트웨이(엔진 수정 유무 무관) 및 엔드포인트 선택 알고리즘이 내장된 UDP ping 비컨 유틸리티를 포함합니다. 소스 빌드 Unreal Engine 5.1 이상이 필요합니다. 액세스하려면 [GitHub의 Epic Games 조직에](#) 가입해야 합니다.

통합 가이드는 각 GitHub 리포지토리의 설명서를 참조하세요.

게임 백엔드 서비스의 경우

게임에 대한 64비트 백엔드 서비스를 생성하고 용 서비스 API가 포함된 AWS SDK의 기능과 통합합니다Amazon GameLift Servers. 백엔드 서비스를 사용하여 게임 세션 시작 또는 찾기, 플레이어의 게임 참가 등 Amazon GameLift Servers 서비스와의 상호 작용을 처리합니다.

[AWS SDK 가져오기](#)

에서 AWS SDK를 사용하는 방법에 대한 자세한 내용은 다음 리소스를 Amazon GameLift Servers참조하세요.

- [Amazon GameLift Servers API 레퍼런스](#)
- 게임 클라이언트 기능 통합
 - [사용자 지정 게임 서버 빌드와 게임을 위한 클라이언트 측 서비스 통합](#)
 - [Realtime 서버와 게임을 위한 클라이언트 측 서비스 통합](#)
- [백엔드 서비스 설계](#)

Amazon GameLift Servers 리소스 관리의 경우

다음 도구를 사용하여 Amazon GameLift Servers 관리형 호스팅 리소스를 생성, 업데이트, 모니터링합니다.

- [AWS Management Console](#) - AWS 콘솔은 모든 개별 AWS 서비스 콘솔에 대한 중앙 집중식 액세스를 제공하는 웹 기반 애플리케이션입니다. Amazon GameLift Servers 콘솔을 사용하여 AWS 계정을 생성하거나 로그인하고 Amazon GameLift Servers 콘솔을 열어 게임 호스팅 리소스로 작업합니다. 호스팅 플릿 및 기타 리소스를 구성 및 배포하고, 사용량 및 성능 지표를 확인하며, 대시보드에서 리소스를 추적하고, 기타 여러 작업을 수행할 수 있습니다. [Amazon GameLift Servers 콘솔로 이동합니다.](#)
- [Amazon GameLift Servers용 서비스 API](#) - 이 API를 사용하면 모든 Amazon GameLift Servers 리소스에 프로그래밍 방식으로 액세스할 수 있습니다. SDK의 일부이며 AWS, 대부분의 인기 프로그래밍 언어와 함께 사용할 수 있도록 다운로드할 수 있습니다. [AWS SDK를 가져옵니다.](#)
- [AWS 명령줄 인터페이스\(CLI\)](#) - AWS CLI를 사용하면 명령줄 셸을 사용하여 AWS 서비스와 상호 작용할 수 있습니다. 도구는 AWS 서비스에 대한 퍼블릭 APIs와 서비스에 사용할 수 있는 사용자 지정 명령에 대한 직접 액세스를 제공합니다. [AWS CLI를 가져옵니다.](#)
- [AWS CloudFormation](#) for Amazon GameLift Servers-이 CloudFormation 서비스는 AWS 리소스를 모델링하고 설정하여 인프라 배포 및 관리를 간소화하는 데 도움이 됩니다. CloudFormation 템플릿을 생성하여 호스팅 솔루션의 Amazon GameLift Servers 리소스를 설명한 다음 템플릿을 사용하여 추가 리소스를 빌드하거나 구성을 업데이트합니다. [Amazon GameLift Servers 리소스 유형 참조](#)를 봅니다.

Amazon GameLift Servers Realtime의 경우

Realtime 서버를 구성하고 배포하여 멀티플레이어 게임을 호스팅할 수 있습니다. 게임 클라이언트가 Realtime 서버에 연결할 수 있도록 하려면 Amazon GameLift Servers Realtime Client SDK를 사용합니다. 시작하려면 [Realtime Client SDK를 다운로드](#)합니다. 구성 정보는 [Amazon GameLift Servers Realtime 게임 클라이언트 통합](#)을 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

- Node.js 런타임 – 10.x, 24.x

 Note

Node.js 10은 2026년 9월 30일에 지원이 종료됩니다. 자세한 내용은 [Node.js 10 FAQs](#). 마이그레이션 지침은 [Node.js 10에서 24로 마이그레이션](#)을 참조하세요.

게임 서버 운영 체제

Realtime 서버는 다음 플랫폼을 실행하는 호스팅 리소스에 배포될 수 있습니다.

- [Amazon Linux 2023](#)
- [Amazon Linux 2](#)

자습서: Amazon GameLift Servers 래퍼를 사용한 빠른 온보딩

Amazon GameLift Servers에 대한 온보딩 자습서에 오신 것을 환영합니다. 이 자습서에서는 클라우드 기반 컴퓨팅 리소스 플릿에서 호스팅할 게임 서버를 빠르게 배포합니다. 이 자습서를 사용하여 Amazon GameLift Servers용 서버 SDK를 게임 코드에 통합하는 작업을 건너뛰고, 대신 Amazon GameLift Servers 서비스와 통신하고 게임 세션을 실행하는 최소한의 기능으로 게임을 배포합니다. 기본 호스팅 솔루션을 설정하고 이를 사용하여 자동 조정 및 매치메이킹과 같은 모든 기능을 경험할 수 있습니다. 또한 라이브 데모의 일부나 테스트 목적으로 게임 프로토타입을 호스팅하는 훌륭한 방법입니다.

이 온보딩 방법의 주요 이점:

- 신속하게 호스팅할 수 있도록 게임 서버를 빠르게 배포합니다.
- 게임 코드 변경이 전혀 필요 없으며, 수정 사항도 요구되지 않습니다.
- 게임 엔진에 관계없이 모든 게임 실행 파일에서 이 방법을 사용합니다.
- 게임 세션 활동 모니터링 및 호스팅 상태를 포함한 모든 Amazon GameLift Servers 관리 도구를 살펴봅니다.

 Note

래퍼는 평가 및 기본 프로덕션 용도로 사용됩니다. 세부 플레이어 세션 관리와 같은 고급 기능을 사용하려면 전체 서버 SDK 통합이 필요합니다.

사전 조건

시작하기 전에 다음을 갖추었는지 확인하세요.

- 적절한 권한이 AWS 계정 있는
- AWS CLI 설치됨
- Go 1.18+
- 멀티플레이어 게임 서버 실행 파일
- Make(Linux/Mac)
- 활성 계정과 함께 설치된 Git

개요

이 자습서에서 배울 내용은 다음과 같습니다.

1. 래퍼 가져오기 및 빌드
2. 게임 빌드 준비
3. 래퍼 구성
4. 게임 서버 빌드 업로드
5. 관리형 EC2 플릿 생성
6. 게임 세션 생성 및 연결
7. 게임 서버 모니터링 및 관리
8. 게임 서버 확장

1단계: 게임 서버 래퍼 가져오기 및 빌드

다음 명령을 사용하여 게임 서버 래퍼 소스를 가져오고 래퍼를 빌드합니다. 이러한 명령은 SSH를 사용하지만 GitHub 리포지토리로 직접 이동할 수도 있습니다.

Windows

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

Mac 및 Linux

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

빌드가 성공하면 'out' 디렉터리가 amazon-gamelift-servers-game-server-wrapper에 추가됩니다. 이 디렉터리에는 지원되는 각 호스팅 플릿 옵션에 대해 하나씩 3개의 폴더가 있으며, 여기에는 빌드 아티팩트 세트가 포함되어 있습니다. 이 자습서에서는 관리형 EC2 플릿에 배포하므로 gamelift-servers-managed-ec2 폴더를 사용합니다.

2단계: 게임 서버 빌드 준비

이 단계에서는 Amazon GameLift Servers에 업로드할 게임 서버 빌드 파일을 준비합니다.

게임 디렉터리 생성

이제 로컬 시스템에서 게임 디렉터리를 준비합니다. 이 디렉터리에는 Amazon GameLift Servers를 사용하여 게임 서버를 실행하는 데 필요한 모든 파일이 포함되어야 합니다. 여기에는 게임 서버 래퍼, 게임 서버 빌드, 래퍼가 게임 서버와 함께 작동하도록 하는 config.yaml이 포함됩니다.

다음 단계를 사용합니다.

1. 관리형 EC2 플릿 게임 서버 래퍼 출력 폴더에서 관리형 EC2 플릿에 배포하기 위한 빌드 아티팩트를 찾습니다. 빌드는 out\linux\amd64\gamelift-servers-managed-ec2에 표시된 대로 출력 디렉터리에 기록됩니다.
2. 게임 서버 실행 파일과 gamelift-servers-managed-ec2 폴더로 실행하는 데 필요한 모든 관련 파일을 복사합니다. 필요에 따라 중첩된 디렉터리를 보유할 수 있습니다.

디렉터리 구조의 예는 다음과 같습니다.

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

3단계: 플릿의 래퍼 구성

Amazon GameLift Servers는 플릿 컴퓨팅 인스턴스의 수명 주기를 관리하여 서버 빌드가 설치된 새 인스턴스를 실행하고 필요에 따라 인스턴스를 재활용합니다. 서비스는 각 인스턴스에서 실행되는 게임 서버 프로세스 수명 주기를 관리합니다. 관리형 EC2 플릿은 어디에 있든 플레이어를 지원하기 위해 여러 위치에 인스턴스를 보유할 수 있습니다.

config.yaml 파일을 편집하여 로깅, 포트 설정, 서버 초기화를 위한 래퍼를 구성합니다.

1. 로깅 설정을 구성합니다. 게임 서버 래퍼는 각 게임 서버 프로세스에 대한 로그 메시지를 생성합니다. 기본적으로 로그 수준은 최대 상세 수준을 위해 디버깅하도록 설정됩니다. 이는 설정 및 문제 해결 시 매우 유용하며 로그 메시지가 얼마나 세부적인지를 결정합니다. 이 경우는 가장 상세 수준입니다. 옵션에는 디버그, 정보, 경고, 오류(최소 상세 수준)가 포함됩니다.
2. 게임 서버 로그 디렉터리의 경로를 지정합니다. 게임 서버 로그의 기본 경로는 ./game-server-logs입니다. 이 디렉터리에는 게임 서버에서 생성된 모든 로그가 포함되며 각 인스턴스에는 로그가 있습니다. 로그는 이벤트 탭에서 액세스할 수 있는 Amazon GameLift Servers에 자동으로 업로드됩니다. 자세한 내용은 문제 해결 섹션을 살펴보세요.
3. 네트워크 포트 구성을 정의합니다. 게임 포트를 원하는 대로 설정합니다. 이 자습서에서는 인스턴스당 하나의 동시 게임 서버 프로세스만 실행하는 플릿을 생성하므로 포트 하나만 지정합니다. 한 번에 여러 프로세스를 실행하기로 결정한 경우 각 동시 프로세스에 대해 충분한 포트를 구성해야 합니다. 기본값은 구성 파일에 표시된 대로 37016이지만 일반적으로 Linux 빌드를 사용하는 플릿의 경우 포트 22 및 1026-60000을 사용합니다. Windows 빌드를 사용하는 플릿의 경우 포트 1026-60000을 사용합니다.
4. 게임 서버 실행 파일의 경로를 설정합니다. ./MyGame/my-server-executable.exe의 경우 게임 서버 실행 파일의 경로를 실제 이름과 위치로 사용자 지정합니다. 게임 서버를 시작하기 위한 진입점입니다.

5. 게임 서버 인수를 구성합니다. 최소한 `--port` 인수를 지정하고 이전에 정의한 것과 동일한 게임 포트 값을 사용합니다. 'pos' 값 0은 첫 번째 인수임을 나타냅니다. 필요에 따라 다른 인수를 추가합니다. 이러한 인수는 시작 시 게임 서버로 전달되므로 런타임 동작을 구성할 수 있습니다.

- a. 인수: `"--port"`
- b. 값: `"{{.port number here}}"`
- c. 위치: 0(목록의 첫 번째 인수)

구성 예제:

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

4단계: 게임 서버 빌드 업로드

이제 게임 서버 빌드의 모든 필수 요소(게임 서버 래퍼, config.yaml, 게임 서버 파일)를 완료했으며 호스팅을 위해 게임 빌드를 Amazon GameLift Servers에 업로드할 준비가 되었습니다. 게임 빌드를 업로드하는 가장 빠른 방법은 다음 예제와 `upload-build` 같이 AWS CLI 명령을 사용하는 것입니다.

Windows를 사용하여 게임 빌드 업로드:

```
aws gamelift upload-build \
  --name gamelift-test-2025-03-11-1 \
  --build-version gamelift-test-2025-03-11-1 \
  --build-root out/windows/amd64/gamelift-servers-managed-ec2 \
  --operating-system WINDOWS_2022 \
  --server-sdk-version 5.3.0 \
  --region us-west-2
```

Note

Mac 및 Linux 빌드의 경우 --operating-system AMAZON_LINUX_2023 사용

빌드를 생성할 때 API 응답으로부터 빌드 ID를 기록하여 플릿 생성에 사용합니다.

5단계: 관리형 EC2 플릿 생성

다음 단계에서는 최소 플릿 구성을 설명하므로 가능한 한 빨리 준비를 마치고 실행할 수 있습니다.

플릿을 생성하려면

1. 에 로그인 AWS Management Console 하고 로 이동합니다Amazon GameLift Servers.
2. 콘솔 창 상단의 메뉴 모음에서 빌드가 어느 리전에 있는지 확인합니다. 플릿이 동일한 리전에 있어야 하거나 빌드를 찾거나 선택할 수 없으므로 기록해 둡니다.
3. 탐색 창의 관리형 EC2 섹션에서 빌드를 선택합니다.
4. 이전에 업로드한 빌드를 선택하여 빌드 세부 정보 페이지를 표시합니다.
5. 플릿 섹션에서 플릿 생성을 선택합니다. 그러면 플릿의 상태를 추적하고 이벤트 탭에서 플릿 생성 이벤트를 볼 수 있는 관리형 EC2 플릿 세부 정보 정의 페이지가 표시됩니다.
6. 이름과 설명을 입력하고 다음을 선택합니다.
7. 인스턴스 세부 정보 정의 페이지에는 기본적으로 빌드의 리전이 표시됩니다. 추가할 리전을 선택합니다.
8. 플릿 유형에서 온디맨드를 선택합니다.
9. 인스턴스 유형에서 c5.large를 선택하고 다음을 선택합니다.
10. 런타임 구성에서 업로드된 게임 빌드는 래퍼를 사용하기 때문에 대신 래퍼 실행 파일을 가리켜야 합니다. Windows 게임 서버의 경우 이는 C:\game\amazon-gamelift-servers-game-server-wrapper.exe입니다. Linux 게임 서버의 경우 이는 /local/game/amazon-gamelift-servers-game-server-wrapper입니다.

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

또한 config.yaml 및 런타임 구성 시작 파라미터에 설정된 포트를 허용하는 범위로 게임 포트 값을 구성합니다. config.yaml의 포트는 플릿의 런타임 구성에 지정된 포트와 일치할 필요는 없지만, 런타임 시 구성이 다른 포트를 지정하는 경우 해당 값은 config.yaml에 있는 값을 재정의합니다. 런타임 구성에 입력된 시작 파라미터도 config.yaml에 있는 파라미터를 재정의합니다.

11.검토 및 생성 페이지에서 모든 구성을 다시 확인한 다음 제출을 선택하여 플릿을 생성합니다. 플릿 상태는 게임 서버를 호스팅하기 위한 용량을 가동함에 따라 변경되며 곧 활성 상태로 표시됩니다. 활성화가 완료되고 플릿이 배포되면 서비스는 게임 세션 요청을 수신할 준비가 된 래퍼를 시작합니다.

6단계: 게임 세션 생성 및 연결

플릿 상태가 활성으로 표시되면 게임 서버가 게임 세션을 호스팅하기 위해 준비되고 대기 중임을 의미합니다. 게임 세션을 시작하려면 Amazon GameLift Servers 서비스에 게임 세션을 요청합니다. 여기서는 AWS CLI 를 사용하여이 요청을 수행합니다.

Note

를 사용하여 게임 세션을 생성하는 AWS CLI 것은 프로세스를 테스트하고 익숙해지는 데 유용합니다. 어느 시점에서는 게임 매치메이킹 또는 게임 세션 배치 시스템의 일부로 프로그래밍 방식의 AWS SDK 호출을 게임 백엔드 서비스에 추가합니다.

다음을 사용하여 게임 세션을 생성합니다.

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

사용자 지정 게임 속성을 서버 실행 파일에 전달할 수도 있습니다. 자세한 내용은 README의 게임 서버 인수를 참조하세요. create-game-session 호출을 수신하면 Amazon GameLift Servers는 래퍼에 게임 서버 실행 파일을 시작하고 게임 세션을 시작하도록 알립니다. config.yaml의 내용은 게임 서버의 구성에 영향을 미치며 콘솔에 설정된 시작 파라미터는 게임 세션 자체의 구성을 결정합니다.

게임 속성을 추가하기 위한 예제 형식:

```
defaultArgs:
  - arg: "--port"
    val: "${.GamePort}"
```

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

7단계: 플릿 관리 및 모니터링

이제 게임 서버 플릿이 설정되고 게임 세션이 시작되었으므로 Amazon GameLift Servers 콘솔에서 이를 관리하고 모니터링할 수 있습니다. 이를 수행하는 가장 좋은 방법은 플릿 세부 정보 페이지에서 플릿 세부 정보를 편집하거나 조정 탭에서 플릿 조정 및 용량을 변경할 수 있습니다. 게임 서버 조정에 대한 다음 섹션을 참조하세요.

지표 탭을 선택하면 활동 및 하드웨어 지표에 대한 모니터링을 보여주는 그래프를 볼 수 있습니다. 지표 그래프에 대한 자세한 내용은 플릿 ID 옆의 정보 링크를 선택하세요. 지표 탭에서 게임 서버를 면밀히 모니터링할 수 있는 반면, CloudWatch 대시보드에서 이러한 지표에 경보를 추가할 수도 있습니다.

콘솔에서 CloudWatch 대시보드로 이동하려면:

1. 검색창에 'CloudWatch'를 입력하고 검색 결과 목록에서 선택하여 CloudWatch 개요를 표시합니다.
2. 아래로 스크롤하여 GameLift 대시보드 보기를 선택하면 플릿 및 게임 세션의 주요 지표를 기반으로 한 그래프를 볼 수 있습니다.

8단계: 게임 서버 확장

다음 단계에서는 Auto Scaling을 설정합니다. Auto Scaling을 통해 게임 서버상의 활동에 따라 동적으로 플릿 용량을 조정합니다. 플레이어가 도착해 게임 세션을 시작하면 Auto Scaling이 더 많은 인스턴스를 추가하고, 플레이어의 수요가 줄어들면 Auto Scaling이 사용되지 않는 인스턴스를 해제할 수 있습니다. 이는 호스팅 리소스와 비용을 최소화하면서, 원활하고 빠른 플레이어 경험을 제공할 수 있는 효과적인 방법입니다.

게임 시작을 준비할 때 플릿에 대한 Auto Scaling을 설정해야 합니다. Auto Scaling은 호스팅 리소스와 비용을 최소화하면서, 원활하고 빠른 플레이어 경험을 제공할 수 있는 효과적인 방법으로 권장됩니다.

수동으로 플릿 용량을 설정하려면

수동으로 플릿 용량을 설정하려면

1. 플릿의 세부 정보 페이지에서 조정 탭으로 이동합니다.
2. 위치를 선택하고 편집을 선택합니다.
3. 현재 값 이상으로 확장하기 위해 원하는 인스턴스 값을 변경하고 최소 및 최대 크기 설정을 변경한 다음 확인을 선택합니다.

Note

최대 인스턴스 설정을 임시 방편으로 사용하여 과도한 조정 및 비용을 방지합니다.

대상 기반 Auto Scaling을 사용하려면

대상 기반 Auto Scaling을 사용하려면

대상 기반 Auto Scaling(대상 추적)은 플릿 크기 조정을 사용 가능한 게임 세션의 백분율에 연결합니다. 플레이어가 게임을 하기 위해 급증하고 사용 가능한 게임 세션이 감소함에 따라, 이는 자동으로 플릿에 더 많은 인스턴스를 추가하여 대응합니다.

1. 대상 기반 Auto Scaling 정책에서 정책 추가를 선택하고 설정한 사용 가능한 게임 세션 비율의 임계 값에 도달하면 자동으로 변경되도록 플릿의 용량을 설정합니다. 더 큰 버퍼는 급증하는 수요에 더 잘 대처하여 새로운 플레이어들이 게임에 빠르게 참여하도록 할 수 있지만, 호스팅 비용을 증가시킬 수도 있습니다.
2. 확인을 선택하여 변경 사항을 수락합니다.

규칙 기반 Auto Scaling은 조정을 다른 플릿 지표에 연결하고 사용자 지정 임계값 및 조정 응답을 설정하는 기능과 같은 보다 세분화된 제어를 제공합니다. 강력한 옵션들을 제공하지만, 또한 CLI를 사용해야 하며 사용자 지정 규칙이 작동하는 방식을 이해하기 위해 상당한 테스트를 요구합니다. 이 자습서에서는 먼저 대상 기반 접근 방식을 설정하는 데 중점을 둡니다.

일반적인 문제 해결

다음은 게임 서버와 게임 세션을 실행하는 동안 발생할 수 있는 일반적인 문제입니다. 서버 또는 게임 세션이 제대로 실행되지 않는 경우 첫 번째 단계는 프로덕션 환경에서 새 배포 또는 게임에 대해 아래에 설명된 문제 중 하나를 나타낼 수 있는 로그를 확인하는 것입니다.

다음이 로그에 표시되는 경우가 많습니다.

- 게임 서버 프로세스를 시작할 수 없습니다. 이는 래퍼 구성의 오류일 수 있습니다. 파일에 올바른 시작 경로와 올바른 시작 파라미터 및 인수가 있는지 확인합니다.
- 게임 서버 빌드를 실행할 수 없습니다. 게임 코드에 오류가 있을 수 있습니다.
- 플레이어는 게임 세션에 연결할 수 없습니다. 이는 포트 구성 오류일 수 있습니다.
- 지연되거나 느린 연결. 조정 정책 및 임계값을 검토합니다.
- 연결이 없습니다. 플릿의 포트 규칙 및 구성을 확인합니다.

Amazon GameLift Servers 플릿의 이벤트 로그를 확인하려면

Amazon GameLift Servers 플릿의 이벤트 로그를 확인하려면

1. Amazon GameLift Servers 콘솔을 엽니다.
2. 플릿 세부 정보 페이지에서 이벤트 탭을 선택하고 로그를 다운로드합니다. 지표 탭에서 게임 서버 상태 및 게임 세션 활성화에 대한 활동 및 하드웨어 지표를 모니터링할 수도 있습니다.

게임 세션 대기열을 확인하려면

게임 세션 대기열을 확인하려면

1. 콘솔에서 플릿을 열고 게임 세션 탭을 엽니다.
2. 목록에서 게임 세션 ID를 선택하여 개요 페이지를 표시합니다.
3. 로그 다운로드를 선택하여 로그 파일을 로컬로 다운로드합니다.

CLI를 사용하여 게임 세션 로그를 보려면 `GetGameSessionLogURL` API를 사용합니다. Amazon GameLift Servers는 14일 동안 로그를 자동으로 저장합니다.

플릿에 대한 Amazon CloudWatch logs를 구성할 수도 있습니다. 이를 통해 추가 로깅 기능을 제공하고 다른 AWS 모니터링 서비스와 통합할 수 있습니다.

CloudWatch를 통한 실시간 로그 액세스 또는 장기 보존 기간의 경우:

1. Amazon GameLift Servers 콘솔 대시보드 상단의 검색창에 'CloudWatch'를 입력하고 결과 드롭다운에서 선택합니다.
2. CloudWatch 로그 그룹으로 이동하여 특정 세션을 검색합니다. 가장 쉬운 방법은 모두 검색을 클릭하고 `gameSessionId` 또는 `clientId`를 사용하여 필터링하는 것입니다.

다음 단계

- [다중 위치 플릿을 생성하여 추가 위치에 호스팅 추가](#)
- [게임 세션 대기열을 추가하여 다중 위치에서 게임 세션을 최대한 배치](#)
- [게임에 대한 매치메이커 및 규칙 세트를 생성하여 FlexMatch 매치메이킹 실험](#)
- [플레이어가 참여 요청을 하고 게임 세션에 직접 연결할 수 있도록 게임 클라이언트 및 백엔드 서비스 구성 요소의 기능 작업 시작](#)
- [준비가 되면 완전히 통합된 솔루션으로 이동](#)

Amazon GameLift Servers 예제

Amazon GameLift Servers를 사용하여 사용자 지정 게임 서버를 관리하거나 Amazon GameLift Servers Realtime을 활용하려면 이 서비스를 자체 게임에 사용하기 전에 다음 예제를 시도해 보는 것이 좋습니다. 사용자 지정 게임 서버 예제는 Amazon GameLift Servers 콘솔에서의 게임 호스팅 경험을 제공합니다. Amazon GameLift Servers Realtime 예제는 Realtime 서버를 사용하여 호스팅할 게임을 준비하는 방법을 보여줍니다.

사용자 지정 게임 서버 예제

이 예제에서는 호스팅을 위해 샘플 게임 서버를 Amazon GameLift Servers 관리형 EC2 플릿에 배포하는 프로세스를 보여줍니다. 샘플 게임 클라이언트를 사용하여 라이브 게임 세션에 연결합니다. 콘솔 및 AWS CLI를 포함한 Amazon GameLift Servers 도구를 사용하여 플릿의 호스팅 성능 및 사용량을 모니터링하는 기능을 사용해볼 수 있습니다.

이 예제에서는 다음 단계를 살펴봅니다.

- 샘플 게임 서버 빌드를 업로드합니다.
- 게임 서버 빌드를 실행할 플릿을 생성합니다.
- 샘플 게임 클라이언트를 가져와 게임 서버에 연결하고 게임 세션에 연결합니다.
- 플릿 및 게임 세션 지표를 검토합니다.

여러 게임 클라이언트를 시작하고 플레이하여 호스팅 데이터를 생성할 수 있습니다. Amazon GameLift Servers 콘솔을 사용하여 호스팅 리소스를 보고, 지표를 추적하고, 플릿의 호스팅 용량을 조정하기 위한 옵션을 탐색할 수 있습니다.

시작하려면 [Amazon GameLift Servers 콘솔](#)에 로그인합니다. 왼쪽 탐색 창에서 리소스, 샘플 게임 시도로 이동합니다.

Amazon GameLift Servers를 사용하여 게임 호스팅 솔루션 구축

이 섹션에는 설계 모범 사례를 포함하여 Amazon GameLift Servers 게임 호스팅 솔루션의 모든 구성 요소를 구축하는 지침이 포함되어 있습니다. 처음부터 시작하든 기존 게임을 마이그레이션하든 Amazon GameLift Servers를 사용하여 게임 호스팅 인프라를 설계, 구현 및 최적화하는 데 도움이 되는 포괄적인 정보를 찾을 수 있습니다.

호스팅 솔루션 구성 요소

호스팅 솔루션에는 다음 구성 요소의 일부 버전이 있어야 합니다. 이러한 구성 요소와 구성 요소가 함께 작동하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers를 사용한 호스팅 작동 방식](#) 섹션을 참조하세요.

- 게임 클라이언트는 플레이어 디바이스에서 실행됩니다.
- 백엔드 서비스는 게임 클라이언트와 Amazon GameLift Servers 간의 통신을 조정합니다.
- 게임 서버 소프트웨어는 멀티플레이어 게임 플레이를 관리합니다.
- 배치 시스템은 플레이어를 게임 서버에 매칭합니다.
- 게임 호스팅 플릿은 지리적 위치에 컴퓨팅 리소스를 제공합니다.
- 게임 호스팅 관리 시스템은 게임 호스팅 상태를 모니터링하고 용량을 관리합니다.

주제

- [개발 로드맵](#)
- [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)
- [Amazon GameLift Servers에 대한 백엔드 서비스 구축](#)
- [Amazon GameLift Servers용 호스팅 플릿 배포](#)
- [게임 세션 배치 구성](#)
- [게임 호스팅 솔루션에 맞게 사용자 지정](#)

개발 로드맵

개발 로드맵은 Amazon GameLift Servers를 사용하여 다양한 유형의 게임 호스팅 솔루션을 구축하기 위한 구조 단계별 지침을 제공합니다. 각 로드맵은 특정 호스팅 접근 방식에 맞게 조정되며 초기 설정부터 배포 및 테스트에 이르기까지 필수 작업을 안내합니다.

호스팅 요구 사항 및 기술적 제약 조건에 가장 적합한 로드맵을 선택합니다. 또한 여러 로드맵에 따라 접근 방식을 결합하여 특정 요구 사항을 충족하는 하이브리드 솔루션을 만들 수 있습니다.

주제

- [Amazon GameLift Servers 관리형 EC2를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers Anywhere 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers를 사용한 하이브리드 호스팅을 위한 개발 로드맵](#)

Amazon GameLift Servers 관리형 EC2를 사용한 호스팅을 위한 개발 로드맵

이 로드맵은 멀티플레이어 게임을 위해 Amazon GameLift Servers 관리형 EC2 호스팅 솔루션을 개발하는 방법을 안내합니다. Amazon GameLift Servers는 여러 게임 호스팅 옵션을 제공합니다. 이러한 옵션에 대한 자세한 내용은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

Amazon GameLift Servers 관리형 호스팅을 사용하면 구성에 따라 Amazon GameLift Servers 소유하고 작동하는 AWS 클라우드 기반 가상 컴퓨팅 리소스에서 게임 서버가 호스팅됩니다. 멀티플레이어 게임 호스팅에 사용하도록 추가로 최적화된 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스의 보안, 안정성, 글로벌 가용성을 얻을 수 있습니다. Amazon GameLift Servers는 자동 서버 배포, 수명 주기 처리, 용량 Auto Scaling과 같은 도구를 사용하여 호스팅 관리를 간소화합니다.

Amazon GameLift Servers 관리형 솔루션은 다음 구성 요소로 구성됩니다.

- Amazon GameLift Servers용 서버 SDK와 통합되어 모든 플릿에 배포되는 게임 서버 빌드.
- AWS SDK와 통합된 게임 클라이언트 및 백엔드 서비스로, Amazon GameLift Servers 서비스와 상호 작용하고 게임 세션을 요청합니다.
- 모든 플릿에서 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하는 Amazon GameLift Servers 대기열 또는 다른 배치 메커니즘.
- (선택 사항) 다중 플레이어 매치를 생성하고 해당 매치에 대한 게임 세션을 설정하는 FlexMatch 매치 메이커.
- 멀티플레이어 게임 호스팅에 최적화된 Amazon Elastic Compute Cloud(Amazon EC2) 컴퓨팅 인스턴스를 사용하는 하나 이상의 Amazon GameLift Servers 관리형 플릿.

이 로드맵은 Amazon GameLift Servers 관리형 EC2 호스팅을 통해 멀티플레이어 게임을 성공적으로 시작하고 실행할 수 있는 간소화된 경로를 제공합니다. 필요한 구성 요소가 준비되면 게임 개발을 계속 반복하고 호스팅 솔루션을 사용자 지정할 수 있습니다. 출시일이 가까워지면 프로덕션 수준 사용을 위

한 호스팅 솔루션을 준비하는 데 필요한 [Amazon GameLift Servers 호스팅으로 출시 준비](#) 섹션을 참조하세요.

Unreal Engine 및 Unity용 Amazon GameLift Servers 플러그인으로 바로 시작

더 빠른 배포를 위해 Unreal Engine 및 Unity용 [Amazon GameLift Servers 플러그인](#)을 사용해 보세요. 최소한의 설정으로 게임 서버를 빠르게 배포할 수 있는 안내 UI 워크플로를 제공하므로 게임 구성 요소를 실제로 사용해 볼 수 있습니다. 그런 다음 이 기반을 바탕으로 게임에 대한 사용자 지정 호스팅 솔루션을 생성할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 플러그인으로 Unreal 또는 Unity 게임 준비](#) 섹션을 참조하세요.

1단계: Amazon GameLift Servers에서 작동하도록 게임 서버 준비

게임 서버가 호스팅을 위해 배포될 때 Amazon GameLift Servers 서비스와 통신할 수 있도록 게임 서버에 기능을 추가합니다.

- 게임 프로젝트에 대한 Amazon GameLift Servers용 서버 SDK(버전 5.x)를 가져옵니다. 서버 SDK는 C++, C# 및 Go로 지원됩니다. [Amazon GameLift Servers 서버 SDK를 다운로드](#)합니다.
- 게임 서버 코드를 수정하여 서버 SDK 기능을 추가합니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)를 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 코드를 추가하여 Amazon GameLift Servers SDK를 초기화하고 Amazon GameLift Servers 서비스와 WebSocket 연결을 설정합니다. 서버 SDK 작업 `InitSdk()`을 사용합니다.
 - 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessReady()`을 사용합니다.
 - 필요한 `OnProcessTerminate()`, `OnStartGameSession()` 콜백 함수를 구현합니다. 이러한 함수를 사용하면 게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 연결을 유지하고, Amazon GameLift Servers에서 메시지가 표시되면 게임 세션을 시작하고, 프롬프트에 응답하여 게임 서버 프로세스를 종료할 수 있습니다.
 - 서버 프로세스가 게임 세션을 종료하면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessEnding()`을 사용합니다.
- 게임 서버 빌드를 패키징합니다. 빌드 파일, 종속 항목 및 지원 소프트웨어를 사용하여 설치 스크립트를 생성합니다. [게임 빌드 파일 패키징하기](#)(를) 참조하세요. Amazon Simple Storage Service(Amazon S3) 버킷을 사용하여 게임 빌드 버전을 저장하는 것이 좋습니다.
- 게임 서버 통합을 테스트합니다. 이 작업의 경우 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)에 설명된 대로 로컬 워크스테이션에 Amazon GameLift Servers Anywhere 플릿

을 설정하는 것이 좋습니다. 이 단계에서는 게임 서버 빌드를 테스트 디바이스에 수동으로 설치하고 서버 프로세스를 시작합니다. AWS CLI를 사용하여 새 게임 세션을 요청하고 Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하는지 확인합니다.

2단계: 호스팅한 게임 세션에 연결할 게임 클라이언트 준비

게임 클라이언트가 게임 세션에 연결하도록 요청하고 연결 정보를 얻은 다음 호스팅된 게임 세션에 직접 연결할 수 있는 방법을 생성합니다. 가장 일반적인 접근 방식은 게임 클라이언트와 Amazon GameLift Servers 서비스 간에 중간자 역할을 하는 백엔드 서비스 기능을 설정하는 것입니다. 이 접근 방식은 호스팅 리소스를 보호하고 플레이어가 게임 세션에 배치되는 방식을 더 잘 제어할 수 있도록 합니다.

- 호스팅을 위한 백엔드 서비스 기능을 구축합니다. 백엔드 서비스가 Amazon GameLift Servers 서비스와 통신하고 게임 클라이언트에 연결 정보를 전송합니다. 이 기능에는 게임 세션 시작, 플레이어를 게임에 배치, 게임 세션 정보 검색이 포함됩니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
- 용 AWS SDK를 가져Amazon GameLift Servers와 백엔드 서비스 프로젝트에 추가합니다. [클라이언트 서비스에 대한 Amazon GameLift Servers SDK 리소스](#)를 참조하세요.
- Amazon GameLift Servers 클라이언트를 초기화하고 키 설정을 저장하는 코드를 추가합니다. [Amazon GameLift Servers API를 설정합니다.](#)을(를) 참조하세요.
- AWS SDK 작업을 호출CreateGameSession()하고 게임 세션 연결 정보를 게임 클라이언트에 제공하는 기능을 추가합니다. [특정 플릿에서 게임 세션 생성](#) 페이지를 참조하세요.

CreateGameSession() 호출로 간편하게 새 게임 세션 요청을 시작할 수 있습니다. 게임 세션 배치 시스템을 설치한 후(3단계 참조) 이 코드를 StartGameSessionPlacement()에 대한 호출(또는 FlexMatch를 사용하는 경우 StartMatchmaking() 코드)로 바꿉니다.

백엔드 서비스 설계에 대한 지침은 [Amazon GameLift Servers에 대한 백엔드 서비스 구축](#) 페이지를 참조하세요.

- 플레이어가 호스팅된 게임 세션에 연결할 수 있도록 게임 클라이언트에 기능을 추가합니다. 게임 클라이언트가 Amazon GameLift Servers가 아닌 백엔드 서비스에 직접 요청합니다. 백엔드 서비스가 게임 세션 연결 정보를 제공한 후 게임 클라이언트는 게임 세션과 직접 연결하여 게임을 플레이합니다.
- 게임 클라이언트 통합을 테스트합니다. 로컬 워크스테이션과 동일한 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다.

빠른 반복 개발을 위해 또는 다중 사용자 팀과 작업할 때 [클라우드 기반 테스트 환경](#)을 설정하는 것이 좋습니다. 이 Amazon GameLift Servers Toolkit 솔루션은 Amazon GameLift Servers 관리형 플릿의 동작을 모방하지만 최소 처리 시간으로 게임 서버 빌드를 업데이트할 수 있습니다.

3단계: 게임 세션 배치 설정

Amazon GameLift Servers가 새 게임 세션에 대한 요청을 처리하고 호스팅할 사용 가능한 게임 서버를 찾는 방법을 사용자 지정합니다. Amazon GameLift Servers는 모든 플릿에서 모든 게임 서버의 가용성을 자동으로 추적합니다. 게임 클라이언트가 게임 세션에 연결하라는 요청을 보내면 Amazon GameLift Servers는 최소 지연 시간, 비용 및 가용성과 같이 정의된 우선 순위 집합을 기반으로 '최상의' 배치를 찾습니다.

- 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하기 위한 게임 세션 대기열을 생성합니다. 대기열은 게임 세션 배치의 기본 메커니즘입니다. 자세한 지침은 [게임 세션 대기열 생성](#)을 참조하세요.
- 최소한 Anywhere 플릿을 대기열의 대상으로 추가합니다. 다른 모든 설정은 선택 사항입니다.
- 백엔드 서비스 코드에서 **CreateGameSession()** 호출을 **StartGameSessionPlacement()**로 변환합니다. [다중 위치 대기열에서 게임 세션 생성](#) 페이지를 참조하세요.
- 게임 세션에 연결할 준비가 되면 게임 클라이언트에 알리는 메커니즘을 생성합니다. 개발 중에 DescribeGameSessionPlacement에 대한 호출을 사용하여 게임 세션 상태를 폴링할 수 있습니다. 그러나 대기열을 사용하여 대용량을 처리하기 전에 이벤트 알림을 활성화해야 합니다. [게임 세션 배치의 이벤트 알림 설정](#)(를) 참조하세요.
- (선택 사항) FlexMatch 매치메이킹 구성 요소를 추가합니다. 지침은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.

4단계: 관리형 클라우드 기반 플릿 생성

지금까지 자체 관리형 Anywhere 플릿을 사용하여 게임 구성 요소를 테스트하고 반복했으며 게임 세션 배치를 미세 조정했습니다. 솔루션의 마지막 부분은 프로덕션 시스템에 필요한 호스팅 리소스 유형을 설정하는 것입니다. 프로덕션을 위한 계획 및 구성을 시작하려면 Amazon GameLift Servers 관리형 플릿 작업으로 전환하려고 합니다.

- 게임 서버 빌드를 패키징하고 Amazon GameLift Servers에 업로드합니다. 빌드 파일, 종속 항목 및 지원 소프트웨어를 사용하여 설치 스크립트를 생성합니다. [Amazon GameLift Servers의 게임 서버 빌드 생성](#)(를) 참조하세요. 콘솔 또는 AWS CLI를 Amazon GameLift Servers 사용하여 빌드를 업로드할 수 있습니다.

빌드를 업로드하기 전에 플릿을 AWS 리전 생성할 항목을 결정합니다. 빌드를 동일한 리전에 업로드해야 합니다. 플릿 위치 선택에 대한 자세한 내용은 [지리적 위치](#) 페이지를 참조하세요.

- 관리형 EC2 플릿을 생성합니다. 플릿을 생성하면 Amazon GameLift Servers가 호스팅을 위해 게임 서버 빌드 배포를 즉시 시작합니다. 관리형 플릿의 여러 부분을 구성할 수 있습니다. 자세한 지침은 [Amazon GameLift Servers 관리형 EC2 플릿 생성](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 플릿에 이름을 지정하고 업로드된 게임 빌드 중 배포할 빌드를 지정합니다.
 - 플릿의 온디맨드 인스턴스를 선택하고 플릿 위치에서 사용할 수 있는 인스턴스 유형을 선택합니다. 스팟 플릿은 중요한 옵션이지만 추가 설계 및 구성이 필요합니다.
 - 플릿에 대한 런타임 구성을 생성합니다. 최소한 게임 서버 실행 파일의 시작 경로를 지정합니다.
 - 인바운드 트래픽이 게임 서버에 액세스할 수 있도록 포트 설정을 지정합니다.
- 관리형 플릿을 대기열에 추가합니다. 게임 세션 대기열에서 관리형 플릿을하여 Anywhere 플릿으로 바꿉니다.
- 관리형 플릿으로 게임 호스팅을 테스트합니다. 이 시점에서 게임 클라이언트가 게임 세션을 요청하고, 연결 정보를 얻고, 게임 세션에 성공적으로 연결하여 전체 호스팅 주기를 테스트할 수 있어야 합니다.

5단계: 관리형 플릿 사용자 지정

게임 출시를 준비할 때 관리형 호스팅 리소스를 미세 조정해야 합니다. 고려해야 할 몇 가지 결정 사항은 다음과 같습니다.

- 비용 절감을 위해 스팟 플릿을 추가하는 것이 좋습니다. [스팟 플릿으로 게임 호스팅 비용 절감](#)을(를) 참조하세요.
- 게임 서버가 다른 AWS 리소스를 통신해야 하는 경우 액세스를 관리할 IAM 역할을 설정합니다. [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#)을(를) 참조하세요.
- 게임 서버를 위치시킬 지리적 위치를 결정합니다. 관리형 플릿에 원격 위치를 추가합니다. [리소스 사용자 지정 호스팅](#)을(를) 참조하세요.
- 인스턴스 유형과 크기를 선택하고 여러 서버 프로세스를 실행하도록 런타임을 구성하여 플릿 성능을 최적화합니다. [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)을(를) 참조하세요.
- 우선 순위 설정 사용자 지정을 포함하여 관리형 플릿에 대한 게임 세션 배치 옵션을 실험합니다. [게임 세션 대기열 사용자 지정](#)을(를) 참조하세요.

- 예상 플레이어 수요에 맞게 자동 용량 조절을 설정합니다. [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)을(를) 참조하세요.
- 다른에서 대기 풀릿을 설정하고 필요한 경우 장애 조치를 처리하도록 대기열 및 Auto Scaling을 AWS 리전 수정합니다.
- 분석 및 로깅을 포함한 호스팅 관찰성 도구를 설정합니다. [Amazon GameLift Servers 모니터링](#)을(를) 참조하세요.
- [코드형 인프라\(IaC\)](#)를 사용하여 배포를 자동화합니다. [를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation](#)을(를) 참조하세요.

Amazon GameLift Servers는 배포별 구성에 대한 AWS CloudFormation 템플릿 사용을 지원합니다. AWS Cloud Development Kit (AWS CDK) 를 사용하여 Amazon GameLift Servers 리소스를 정의할 수도 있습니다. 에 대한 자세한 내용은 [AWS Cloud Development Kit \(AWS CDK\) 개발자 안내서](#)를 AWS CDK참조하세요.

CloudFormation 스택 배포를 관리하려면와 같은 지속적 통합 및 지속적 전달(CI/CD) 도구 및 서비스를 사용하는 것이 좋습니다 AWS CodePipeline. 이러한 도구를 사용해 게임 서버 바이너리를 빌드할 때마다 자동으로 배포하거나 승인을 받아 배포할 수 있습니다. CI/CD 도구 또는 서비스를 사용하면 다음과 같은 새 게임 서버 버전에 대한 리소스 배포가 가능합니다.

- 게임 서버 바이너리를 빌드하고 테스트합니다.
- 바이너리를 Amazon GameLift Servers에 업로드합니다.
- 새 빌드로 새 풀릿을 배포합니다.
- 게임 세션 대기열에 새 풀릿을 추가하고 이전 빌드 버전의 풀릿을 제거합니다.
- 이전 빌드의 풀릿이 더 이상 활성 게임 세션을 호스팅하지 않는 경우 해당 풀릿의 CloudFormation 스택을 삭제합니다.

Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵

이 로드맵은 컨테이너화된 게임 서버를 위한 Amazon GameLift Servers 관리형 호스팅 솔루션을 개발하는 방법을 안내합니다. 관리형 컨테이너는 Amazon GameLift Servers에서 제공하는 하나의 호스팅 솔루션일 뿐입니다. 호스팅 옵션에 대한 자세한 내용은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

Amazon GameLift Servers를 사용하는 관리형 컨테이너 솔루션의 구성 요소는 다음과 같습니다.

- Amazon Elastic Container Registry(Amazon ECR) 프라이빗 리포지토리에 업로드된 게임 서버 빌드가 포함된 컨테이너 이미지입니다. 게임 서버 빌드는 Amazon GameLift Servers용 서버 SDK와 통합되며 Linux에서 실행되도록 구축됩니다.
- 게임 클라이언트를 대신하여 Amazon GameLift Servers 서비스와 상호 작용하는 백엔드 서비스입니다. 백엔드 서비스는 AWS SDK의 일부인 Amazon GameLift Servers용 서비스 API의 기능을 사용합니다.
- 새 게임 세션에 대한 요청을 처리하고, 모든 플릿에서 사용 가능한 게임 서버를 검색하며, 게임 서버에 게임 세션을 시작하라는 메시지를 표시하는 Amazon GameLift Servers 게임 세션 대기열 또는 다른 배치 메커니즘입니다.
- (선택 사항) 다중 플레이어 매치를 생성하고 해당 매치에 대한 게임 세션을 설정하는 FlexMatch 매치 메이커
- 멀티플레이어 게임 호스팅에 최적화된 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 사용하는 하나 이상의 컨테이너 플릿

이 로드맵은 Amazon GameLift Servers 관리형 컨테이너를 사용하여 컨테이너화된 게임 서버를 성공적으로 시작하고 실행할 수 있는 간소화된 경로를 제공합니다. 필요한 구성 요소가 준비되면 게임 개발을 계속 반복하고 호스팅 솔루션을 사용자 지정할 수 있습니다. 출시일이 가까워지면 프로덕션 수준 사용을 위한 호스팅 솔루션을 준비하는 데 필요한 [Amazon GameLift Servers 호스팅으로 출시 준비](#) 섹션을 참조하세요.

 관리형 컨테이너용 다음 도구로 온보딩 속도를 높입니다.

- [컨테이너 스타터 키트](#)는 통합 및 플릿 설정을 간소화합니다. 게임 서버에 필수 게임 세션 관리 기능을 추가하고 사전 구성된 템플릿을 사용하여 게임 서버를 위한 컨테이너 플릿과 자동 배포 파이프라인을 빌드합니다. 배포 후 Amazon GameLift Servers 콘솔 및 API 도구를 사용하여 플릿 성능을 모니터링하고, 게임 세션을 관리하며, 지표를 분석합니다.
- Unreal Engine 및 Unity 개발자의 경우 [Amazon GameLift Servers 플러그인](#)을 사용하여 게임 서버를 통합하고 게임 엔진의 개발 환경 내에서 컨테이너 플릿을 빌드합니다. 플러그인의 안내 워크플로는 관리형 컨테이너를 사용한 클라우드 기반 호스팅으로 빠르고 간단한 솔루션을 만드는 데 도움이 됩니다. 이 기반을 바탕으로 게임에 대한 사용자 지정 호스팅 솔루션을 생성할 수 있습니다.

1단계: Amazon GameLift Servers에서 작동하도록 게임 서버 준비

게임 서버가 호스팅을 위해 배포될 때 Amazon GameLift Servers 서비스와 통신할 수 있도록 게임 서버에 기능을 추가합니다.

- 게임 프로젝트에 대한 Amazon GameLift Servers용 서버 SDK(버전 5.x)를 가져옵니다. 서버 SDK는 C++, C# 및 Go로 지원됩니다. [Amazon GameLift Servers용 서버 SDK를 다운로드합니다](#). 서버 SDK는 C++, C# 및 Go로 지원됩니다.
- 게임 서버 코드를 수정하여 서버 SDK 기능을 추가합니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 코드를 추가하여 Amazon GameLift Servers SDK를 초기화하고 Amazon GameLift Servers 서비스와 WebSocket 연결을 설정합니다. 서버 SDK 작업 `InitSdk()`을 사용합니다.
 - 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessReady()`을 사용합니다.
 - 필요한 `OnStartGameSession()`, `OnProcessTerminate()` 콜백 함수를 구현합니다. 이러한 함수를 사용하면 게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 연결을 유지하고, Amazon GameLift Servers에서 메시지가 표시되면 게임 세션을 시작하고, 요청에 응답하여 게임 서버 프로세스를 종료할 수 있습니다.
 - 서버 프로세스가 게임 세션을 종료하면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessEnding()`을 사용합니다.
- 게임 서버 빌드를 패키징합니다. Linux에서 실행할 게임 서버를 빌드합니다. 게임 서버를 실행하는데 필요한 빌드 및 기타 파일을 준비합니다. Windows에서 개발하는 경우 이 단계에는 별도의 Linux 워크스페이스를 설정하거나 Linux용 Windows 하위 시스템(WSL)과 같은 도구를 사용하는 작업이 포함될 수 있습니다. 게임 서버 빌드를 테스트하고 컨테이너 이미지를 빌드 및 테스트하려면 Linux 환경이 필요합니다.
- 게임 서버 통합을 테스트합니다. 통합 게임 서버가 Amazon GameLift Servers 서비스에 연결하고 프로토타입에 응답할 수 있는지 확인합니다. [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)에 설명된 대로 로컬 워크스테이션을 테스트 호스트로 사용하여 간단한 Amazon GameLift Servers Anywhere 플릿 설정을 권장합니다. 게임 서버 빌드를 테스트 호스트에 설치하고 서버 프로세스를 시작합니다. AWS CLI를 사용하여 새 게임 세션을 요청하고 Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하는지 확인합니다.

2단계: 게임 서버 컨테이너 이미지 생성

게임 서버를 성공적으로 통합한 후 게임 서버 실행 파일과 함께 컨테이너 이미지를 생성합니다.

Amazon GameLift Servers에서 사용하기 위해 Amazon Elastic Container Registry(Amazon ECR) 리포

지토리에 저장합니다. 자세한 지침은 [Amazon GameLift Servers용 컨테이너 이미지 빌드](#) 섹션을 참조하세요.

- 게임 서버 컨테이너(Amazon GameLift Servers에서 제공)에 대한 Dockerfile 템플릿을 가져옵니다. 게임 서버 빌드 파일을 위해 해당 파일을 수정합니다.
- 게임 서버 컨테이너 이미지를 생성합니다. Linux 환경에서 작업할 때는 Docker 도구를 사용하여 이미지를 생성합니다.
- Amazon ECR에 컨테이너 이미지를 푸시합니다. 컨테이너 플릿을 배포할 동일한 AWS 계정 및 AWS 리전 를 사용하여 Amazon ECR에서 퍼블릭 또는 프라이빗 리포지토리를 생성합니다. 여기에 컨테이너 이미지를 푸시합니다.
- Anywhere 플릿을 사용하여 컨테이너 이미지를 테스트합니다(선택 사항). 클라우드 호스팅 컨테이너 플릿에 배포하기 전에 로컬에서 컨테이너 이미지를 테스트할 수 있습니다. 로컬 워크스테이션으로 기존 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다. 게임 서버 컨테이너를 설치 및 실행하고 (1) Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하고 (2) 게임 클라이언트가 게임 세션에 연결할 수 있는지 확인합니다.

3단계: 클라우드 기반 컨테이너 플릿 생성

지금까지 자체 관리형 Anywhere 플릿을 사용하여 게임 구성 요소를 테스트하고 반복했습니다. Amazon GameLift Servers용 통합 게임 서버 빌드가 작동하는 경우 프로덕션 환경에 필요한 클라우드 기반 Amazon GameLift Servers 관리형 컨테이너 플릿 호스팅 리소스 설정을 시작할 수 있습니다.

- 컨테이너 그룹 정의를 생성합니다. 컨테이너 그룹 정의는 플릿의 컨테이너 아키텍처를 설명하고 배포할 컨테이너 이미지를 식별합니다. [Amazon GameLift Servers 컨테이너 플릿에 대한 컨테이너 그룹 정의 생성](#)(를) 참조하세요. 컨테이너 이미지가 저장 AWS 리전 되는 동일한에서 컨테이너 그룹 정의를 생성합니다. 플릿 위치 선택에 대한 자세한 내용은 [지리적 위치](#) 페이지를 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 게임 서버 컨테이너 그룹 정의를 생성합니다.
 - 게임 서버 빌드와 함께 컨테이너 이미지와 함께 컨테이너 정의를 추가합니다.
 - 컨테이너의 게임 서버 프로세스에 대한 포트 범위를 구성합니다.
- 관리형 컨테이너 플릿을 생성합니다. 플릿을 생성하면 Amazon GameLift Servers가 호스팅을 위해 게임 서버 빌드 배포를 즉시 시작합니다. 관리형 플릿의 여러 부분을 구성할 수 있습니다. 자세한 지침은 [Amazon GameLift Servers 관리형 컨테이너 플릿 생성](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.

- 컨테이너 플릿에 대한 AWS Identity and Access Management (IAM) 서비스 역할을 설정합니다. [Amazon GameLift Servers의 IAM 서비스 역할 설정](#)을(를) 참조하세요.
- 플릿 인스턴스에 배포할 게임 서버 컨테이너 그룹 정의를 지정합니다.
- 다른 모든 파라미터에 사용할 수 있는 경우 기본값을 사용합니다. Amazon GameLift Servers는 최적의 구성을 위해 일부 파라미터를 계산합니다.
- 컨테이너 플릿을 대기열에 추가합니다. 게임 세션 대기열에서 Anywhere 테스트 플릿을 관리형 컨테이너 플릿으로 바꿉니다.
- 컨테이너 플릿으로 게임 호스팅을 테스트합니다. 이 시점에서 전체 솔루션을 테스트할 수 있어야 합니다. 게임 클라이언트를 시작하고 백엔드 서비스를 통해 게임 세션을 요청합니다. 연결 정보를 가져오고 컨테이너 플릿의 게임 세션에 연결합니다.
- 플릿 배포를 반복합니다. 컨테이너 그룹 정의 및 플릿 구성을 업데이트한 다음 기존 플릿에 업데이트를 배포할 수 있습니다.

4단계: 호스팅된 게임 세션에 참여할 게임 클라이언트 준비

게임 클라이언트가 게임 세션에 연결하도록 요청하고 연결 정보를 얻은 다음 호스팅된 게임 세션에 직접 연결할 수 있는 방법을 생성합니다. 가장 일반적인 접근 방식은 게임 클라이언트와 Amazon GameLift Servers 서비스 간에 중간자 역할을 하는 백엔드 서비스 기능을 설정하는 것입니다. 이 접근 방식은 호스팅 리소스를 보호하고 플레이어가 게임 세션에 배치되는 방식을 더 잘 제어할 수 있도록 합니다.

- 호스팅을 위한 백엔드 서비스 기능을 구축합니다. 백엔드 서비스가 Amazon GameLift Servers 서비스와 통신하고 게임 클라이언트에 연결 정보를 전송합니다. 이 기능에는 게임 세션 시작, 플레이어들을 게임에 배치, 게임 세션 정보 검색이 포함됩니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
- 용 AWS SDK를 가져와 Amazon GameLift Servers와 백엔드 서비스 프로젝트에 추가합니다. [클라이언트 서비스에 대한 Amazon GameLift Servers SDK 리소스](#)를 참조하세요.
- Amazon GameLift Servers 클라이언트를 초기화하고 키 설정을 저장하는 코드를 추가합니다. [Amazon GameLift Servers API를 설정합니다](#)을(를) 참조하세요.
- AWS SDK 작업을 호출 `CreateGameSession()` 하고 게임 세션 연결 정보를 게임 클라이언트에 제공하는 기능을 추가합니다. [특정 플릿에서 게임 세션 생성](#) 페이지를 참조하세요.

`CreateGameSession()` 호출로 간편하게 새 게임 세션 요청을 시작할 수 있습니다. 게임 세션 배치 시스템을 설치한 후(3단계 참조) 이 코드를 `StartGameSessionPlacement()`에 대한 호출(또는 FlexMatch를 사용하는 경우 `StartMatchmaking()` 코드)로 바꿉니다.

백엔드 서비스 설계에 대한 지침은 [Amazon GameLift Servers에 대한 백엔드 서비스 구축](#) 페이지를 참조하세요.

- 플레이어가 호스팅된 게임 세션에 연결할 수 있도록 게임 클라이언트에 기능을 추가합니다. 게임 클라이언트가 Amazon GameLift Servers가 아닌 백엔드 서비스에 직접 요청합니다. 백엔드 서비스가 게임 세션 연결 정보를 제공한 후 게임 클라이언트는 게임 세션과 직접 연결하여 게임을 플레이합니다.
- 게임 클라이언트 통합을 테스트합니다. 로컬 워크스테이션으로 기존 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다. 새 백엔드 서비스를 사용하여 새 게임 세션을 요청하고 (1) Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하고 (2) 게임 클라이언트가 게임 세션에 연결할 수 있는지 확인합니다.

5단계: 게임 세션 배치 설정

Amazon GameLift Servers가 새 게임 세션에 대한 요청을 처리하고 호스팅할 사용 가능한 게임 서버를 찾는 방법을 사용자 지정합니다. Amazon GameLift Servers는 모든 플릿에서 모든 게임 서버의 가용성을 자동으로 추적합니다. 게임 클라이언트가 게임 세션에 연결하라는 요청을 보내면 Amazon GameLift Servers는 최소 지연 시간, 비용 및 가용성과 같이 정의된 우선 순위 집합을 기반으로 '최상의' 배치를 찾습니다.

- 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하기 위한 게임 세션 대기열을 생성합니다. 대기열은 게임 세션 배치의 기본 메커니즘입니다. 자세한 지침은 [게임 세션 대기열 생성](#)을 참조하세요.
- 최소한 Anywhere 플릿을 대기열의 대상으로 추가합니다. 다른 모든 설정은 선택 사항입니다.
- 백엔드 서비스 코드에서 **CreateGameSession()** 호출을 **StartGameSessionPlacement()**로 변환합니다. [다중 위치 대기열에서 게임 세션 생성](#) 페이지를 참조하세요.
- 게임 세션에 연결할 준비가 되면 게임 클라이언트에 알리는 메커니즘을 생성합니다. 개발 중에 DescribeGameSessionPlacement에 대한 호출을 사용하여 게임 세션 상태를 폴링할 수 있습니다. 그러나 대기열을 사용하여 대용량을 처리하기 전에 이벤트 알림을 활성화해야 합니다. [게임 세션 배치의 이벤트 알림 설정](#)(를) 참조하세요.
- FlexMatch 매치메이킹을 추가합니다(선택 사항). 매치메이킹 규칙 세트를 빌드하고 매치메이킹 구성을 생성하여 게임 세션 대기열에서 작업합니다. 매치메이킹 시스템 설정에 대한 지침은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.
- 배치 시스템을 테스트합니다. 로컬 워크스테이션으로 기존 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다. 백엔드 서비스를 사용하여 새 게임 세션을 요청하고 Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하는지 확인합니다.

6단계: 관리형 컨테이너 솔루션 사용자 지정

게임 출시를 준비할 때 관리형 호스팅 리소스를 미세 조정해야 합니다. 고려해야 할 몇 가지 결정 사항은 다음과 같습니다.

- 컨테이너 플릿 구성을 최적화합니다. [Amazon GameLift Servers 컨테이너 플릿 사용자 지정](#)(를) 참조하세요.
- 비용 절감을 위해 스팟 플릿을 추가하는 것이 좋습니다. [스팟 플릿으로 게임 호스팅 비용 절감](#)(를) 참조하세요.
- 게임 서버가 다른 AWS 리소스를 통신해야 하는 경우 액세스를 관리할 IAM 역할을 설정합니다. [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#)(를) 참조하세요.
- 게임 서버를 위치시킬 지리적 위치를 결정합니다. 관리형 플릿에 원격 위치를 추가합니다. [리소스 사용자 지정 호스팅](#)(를) 참조하세요.
- 우선 순위 설정 사용자 지정을 포함하여 관리형 플릿에 대한 게임 세션 배치 옵션을 실험합니다. [게임 세션 대기열 사용자 지정](#)(를) 참조하세요.
- 예상 플레이어 수요에 맞게 자동 용량 조정을 설정합니다. [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)(를) 참조하세요.
- 다른 플릿을 생성하고 필요에 따라 장애 조치를 처리하도록 대기열 및 Auto Scaling을 AWS 리전 수정합니다.
- 분석 및 로깅을 포함한 호스팅 관찰성 도구를 설정합니다. [Amazon GameLift Servers 모니터링](#)(를) 참조하세요.
- [코드형 인프라\(IaC\)](#)를 사용하여 플릿 배포를 자동화합니다. [를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation](#)(를) 참조하세요.

Amazon GameLift Servers는 배포별 구성에 대한 AWS CloudFormation 템플릿 사용을 지원합니다. AWS Cloud Development Kit (AWS CDK) 를 사용하여 Amazon GameLift Servers 리소스를 정의할 수도 있습니다. 에 대한 자세한 내용은 [AWS Cloud Development Kit \(AWS CDK\) 개발자 안내서](#)를 AWS CDK참조하세요.

CloudFormation 스택 배포를 관리하려면와 같은 지속적 통합 및 지속적 전달(CI/CD) 도구 및 서비스를 사용하는 것이 좋습니다 AWS CodePipeline. 이러한 도구를 사용해 게임 서버 바이너리를 빌드할 때마다 자동으로 배포하거나 승인을 받아 배포할 수 있습니다. CI/CD 도구 또는 서비스를 사용하면 다음과 같은 새 게임 서버 버전에 대한 리소스 배포가 가능합니다.

- 게임 서버 바이너리를 빌드하고 테스트합니다.
- 바이너리를 Amazon GameLift Servers에 업로드합니다.
- 새 빌드로 새 플릿을 배포합니다.

- 게임 세션 대기열에 새 플릿을 추가하고 이전 빌드 버전의 플릿을 제거합니다.
- 이전 빌드의 플릿이 더 이상 활성 게임 세션을 호스팅하지 않는 경우 해당 플릿의 CloudFormation 스택을 삭제합니다.

Amazon GameLift Servers Anywhere 호스팅을 위한 개발 로드맵

이 로드맵은 멀티플레이어 게임이 자체 리소스(온프레미스 하드웨어 또는 가상 머신)와 함께 사용할 수 있는 호스팅 솔루션을 개발하는 방법을 안내합니다. Amazon GameLift Servers는 여러 게임 호스팅 옵션을 제공합니다. 이러한 옵션에 대한 자세한 내용은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

Amazon GameLift Servers Anywhere 호스팅을 사용하면 사용자가 제공하고 관리하는 컴퓨팅 리소스에 게임 서버가 호스팅됩니다. 어디든 플레이어가 있는 지리적 위치에서 필요한 구성으로 Anywhere 플릿을 생성할 수 있습니다. Amazon GameLift Servers는 Anywhere 플릿에 대해 다음과 같은 기능을 제공합니다.

- 다음을 포함하여 구성에 따라 게임 세션 배치 프로세스를 처리합니다.
 - Anywhere 플릿 전반의 게임 서버 가용성 추적
 - 게임 클라이언트 서비스의 게임 요청을 처리하고 사용 가능한 서버와 게임 요청 매치
 - Anywhere 플릿의 게임 서버에 게임 세션을 시작하도록 프롬프트
 - 연결 세부 정보를 게임 클라이언트에 다시 전달
- 세션 배치 프로세스의 성능 지표와 게임 세션 및 플레이어의 사용량 지표를 수집합니다.
- 매치메이커를 빌드하고 게임 세션 배치 시스템과 통합할 수 있도록 전체 FlexMatch 매치메이킹 기능 세트를 지원합니다.
- Amazon GameLift Servers 에이전트가 Anywhere 플릿에서 주요 호스트 관리 작업을 처리할 수 있도록 합니다.
- 유연한 하이브리드 솔루션을 위해 Amazon GameLift Servers 관리형 플릿과의 결합을 지원합니다.

Amazon GameLift Servers Anywhere 솔루션은 다음 구성 요소로 구성됩니다.

- Amazon GameLift Servers용 서버 SDK와 통합되어 모든 플릿에 배포되는 게임 서버 빌드
- AWS SDK와 통합된 게임 클라이언트 및 백엔드 서비스로, Amazon GameLift Servers 서비스와 상호 작용하고 게임 세션을 요청합니다.
- 모든 플릿에서 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하는 Amazon GameLift Servers 대기열 또는 다른 배치 메커니즘.

- (선택 사항) 다중 플레이어 매치를 생성하고 해당 매치에 대한 게임 세션을 설정하는 FlexMatch 매치 메이커
- 기존 구성 관리 및 배포 도구로 관리되는 온프레미스 또는 기타 호스팅 리소스가 있는 하나 이상의 Amazon GameLift Servers Anywhere 플릿 (선택 사항으로 사용할 수 있습니다 AWS Systems Manager.)

이 로드맵은 Amazon GameLift Servers Anywhere 호스팅을 통해 멀티플레이어 게임을 성공적으로 시작하고 실행할 수 있는 간소화된 경로를 제공합니다. 필요한 구성 요소가 준비되면 게임 개발을 계속 반복하고 호스팅 솔루션을 사용자 지정할 수 있습니다. 출시일이 가까워지면 프로덕션 수준 사용을 위한 호스팅 솔루션을 준비하는 데 필요한 [Amazon GameLift Servers 호스팅으로 출시 준비](#) 섹션을 참조하세요.

Unreal Engine 및 Unity용 Amazon GameLift Servers 플러그인으로 바로 시작

더 빠른 배포를 위해 Unreal Engine 및 Unity용 [Amazon GameLift Servers 플러그인](#)을 사용해 보세요. 최소한의 설정으로 게임 서버를 빠르게 배포할 수 있는 안내 UI 워크플로를 제공하므로 게임 구성 요소를 실제로 사용해 볼 수 있습니다. 그런 다음 이 기반을 바탕으로 게임에 대한 사용자 지정 호스팅 솔루션을 생성할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 플러그인으로 Unreal 또는 Unity 게임 준비](#) 섹션을 참조하세요.

1단계: Amazon GameLift Servers에서 작동하도록 게임 서버 준비

게임 서버가 호스팅을 위해 배포될 때 Amazon GameLift Servers 서비스와 통신할 수 있도록 게임 서버에 기능을 추가합니다.

- 게임 프로젝트에 대한 Amazon GameLift Servers용 서버 SDK(버전 5.x)를 가져옵니다. 서버 SDK는 C++, C# 및 Go로 지원됩니다. [Amazon GameLift Servers 서버 SDK를 다운로드](#)합니다.
- 게임 서버 코드를 수정하여 서버 SDK 기능을 추가합니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)를 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 코드를 추가하여 Amazon GameLift Servers SDK를 초기화하고 Amazon GameLift Servers 서비스와 WebSocket 연결을 설정합니다. 서버 SDK 작업 InitSdk()를 사용하고 Anywhere 플릿에 필요한 서버 파라미터를 포함합니다.
 - 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 ProcessReady()를 사용합니다.

- 필요한 `OnProcessTerminate()`, `OnStartGameSession()` 콜백 함수를 구현합니다. 이러한 함수를 사용하면 게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 연결을 유지하고, Amazon GameLift Servers에서 메시지가 표시되면 게임 세션을 시작하고, 프롬프트에 응답하여 게임 서버 프로세스를 종료할 수 있습니다.
- 서버 프로세스가 게임 세션을 종료하면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessEnding()`을 사용합니다.
- 게임 서버 빌드를 패키징합니다. 빌드 파일, 종속 항목 및 지원 소프트웨어를 사용하여 설치 스크립트를 생성합니다. [게임 빌드 파일 패키징하기](#)을(를) 참조하세요. Amazon Simple Storage Service(Amazon S3) 버킷을 사용하여 게임 빌드 버전을 저장하는 것이 좋습니다.
- 게임 서버 통합을 테스트합니다. 이 작업의 경우 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)에 설명된 대로 로컬 워크스테이션에 Amazon GameLift Servers Anywhere 플릿을 설정하는 것이 좋습니다. 이 단계에서는 게임 서버 빌드를 테스트 디바이스에 수동으로 설치하고 서버 프로세스를 시작합니다. AWS CLI를 사용하여 새 게임 세션을 요청하고 Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하는지 확인합니다.

2단계: 호스팅한 게임 세션에 연결할 게임 클라이언트 준비

게임 클라이언트가 게임 세션에 연결하도록 요청하고 연결 정보를 얻은 다음 호스팅된 게임 세션에 직접 연결할 수 있는 방법을 생성합니다. 가장 일반적인 접근 방식은 게임 클라이언트와 Amazon GameLift Servers 서비스 간에 중간자 역할을 하는 백엔드 서비스 기능을 설정하는 것입니다. 이 접근 방식은 호스팅 리소스를 보호하고 플레이어가 게임 세션에 배치되는 방식을 더 잘 제어할 수 있도록 합니다.

- 호스팅을 위한 백엔드 서비스 기능을 구축합니다. 백엔드 서비스가 Amazon GameLift Servers 서비스와 통신하고 게임 클라이언트에 연결 정보를 전송합니다. 이 기능에는 게임 세션 시작, 플레이어를 게임에 배치, 게임 세션 정보 검색이 포함됩니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
- 용 AWS SDK를 가져Amazon GameLift Servers와 백엔드 서비스 프로젝트에 추가합니다. [클라이언트 서비스에 대한 Amazon GameLift Servers SDK 리소스](#)를 참조하세요.
- Amazon GameLift Servers 클라이언트를 초기화하고 키 설정을 저장하는 코드를 추가합니다. [Amazon GameLift Servers API를 설정합니다](#)을(를) 참조하세요.
- AWS SDK 작업을 호출`CreateGameSession()`하고 게임 세션 연결 정보를 게임 클라이언트에 제공하는 기능을 추가합니다. [특정 플릿에서 게임 세션 생성](#) 페이지를 참조하세요.

CreateGameSession() 호출로 간편하게 새 게임 세션 요청을 시작할 수 있습니다. 게임 세션 배치 시스템을 설치한 후(3단계 참조) 이 코드를 StartGameSessionPlacement()에 대한 호출 (또는 FlexMatch를 사용하는 경우 StartMatchmaking() 코드)로 바꿉니다.

백엔드 서비스 설계에 대한 지침은 [Amazon GameLift Servers에 대한 백엔드 서비스 구축](#) 페이지를 참조하세요.

- 플레이어가 호스팅된 게임 세션에 연결할 수 있도록 게임 클라이언트에 기능을 추가합니다. 게임 클라이언트가 Amazon GameLift Servers가 아닌 백엔드 서비스에 직접 요청합니다. 백엔드 서비스가 게임 세션 연결 정보를 제공한 후 게임 클라이언트는 게임 세션과 직접 연결하여 게임을 플레이합니다.
- 게임 클라이언트 통합을 테스트합니다. 로컬 워크스테이션과 동일한 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다.

3단계: 게임 세션 배치 설정

Amazon GameLift Servers가 새 게임 세션에 대한 요청을 처리하고 호스팅할 사용 가능한 게임 서버를 찾는 방법을 사용자 지정합니다. Amazon GameLift Servers는 모든 플릿에서 모든 게임 서버의 가용성을 자동으로 추적합니다. 게임 클라이언트가 게임 세션에 연결하라는 요청을 보내면 Amazon GameLift Servers는 최소 지연 시간, 비용 및 가용성과 같이 정의된 우선 순위 집합을 기반으로 '최상의' 배치를 찾습니다.

- 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하기 위한 게임 세션 대기열을 생성합니다. 대기열은 게임 세션 배치의 기본 메커니즘입니다. 자세한 지침은 [게임 세션 대기열 생성](#)을 참조하세요.
 - 최소한 Anywhere 플릿을 대기열의 대상으로 추가합니다. 다른 모든 설정은 선택 사항입니다.
- 백엔드 서비스 코드에서 **CreateGameSession()** 호출을 **StartGameSessionPlacement()**로 변환합니다. [다중 위치 대기열에서 게임 세션 생성](#) 페이지를 참조하세요.
- 게임 세션에 연결할 준비가 되면 게임 클라이언트에 알리는 메커니즘을 생성합니다. 개발 중에 DescribeGameSessionPlacement에 대한 호출을 사용하여 게임 세션 상태를 폴링할 수 있습니다. 그러나 대기열을 사용하여 대용량을 처리하기 전에 이벤트 알림을 활성화해야 합니다. [게임 세션 배치의 이벤트 알림 설정](#)(를) 참조하세요.
- (선택 사항) FlexMatch 매치메이킹 구성 요소를 추가합니다. 지침은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.

4단계: Amazon GameLift Servers 에이전트를 사용하여 Anywhere 플릿 설정

지금까지 로컬 디바이스(Anywhere 플릿 컴퓨팅으로 등록됨)로 작업하여 게임 구성 요소를 테스트하고 반복해 왔습니다. 다음 단계는 프로덕션 시스템에 필요한 플릿 유형을 설정하는 것입니다. 이러한 리소스의 경우 Amazon GameLift Servers 에이전트를 사용하여 일부 주요 온컴퓨터 호스트 관리 작업을 관리합니다. 자세한 내용은 [Amazon GameLift Servers 에이전트 사용법](#) 섹션을 참조하세요.

- Amazon GameLift Servers 에이전트를 가져와서 게임 서버 설치 패키지에 추가합니다. 에이전트 [Amazon GameLift Servers GitHub 리포지토리에서 사용할 수 있는 에이전트](#) 소스 코드를 가져와 빌드합니다. 결과 JAR 파일 실행 파일을 게임 빌드 실행 파일과 동일한 디렉터리에 넣습니다.
- 필요에 따라 에이전트에 대한 시작 스크립트를 수정합니다. 컴퓨팅 실행이 시작되는 즉시 에이전트 실행 파일이 시작되는지 확인합니다. 호스팅 컴퓨팅에서 에이전트를 설치하고 실행하는 방법에 대한 도움말은 에이전트 리포지토리의 readme 파일을 참조하세요. 시작 명령에는 최소한 Anywhere 플릿 ID와 AWS 리전, 사용자 지정 위치 및 컴퓨팅 이름을 지정하는 옵션이 포함되어야 합니다.

에이전트는 자동으로 다음 작업을 처리하므로 스크립트를 사용하여 이러한 작업을 처리한 경우 제거할 수 있습니다.

- RegisterCompute()를 호출하여 Anywhere 플릿에 컴퓨팅을 추가합니다.
- GetComputeAuthToken()을 호출하여 Amazon GameLift Servers 서비스에 연결할 때 게임 서버를 인증합니다. 에이전트는 인증 토큰 가져오기 및 새로 고침을 관리합니다. 인증 토큰은 컴퓨팅에서 실행 중인 모든 게임 서버 프로세스에서 사용할 수 있습니다.
- 런타임 지침 세트를 기반으로 컴퓨팅에서 새 서버 프로세스를 시작합니다.
- Anywhere 플릿의 컴퓨팅에 대한 런타임 구성을 생성합니다. Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 플릿에 대한 런타임 지침을 생성하거나 수정할 수 있습니다. 에이전트는 이러한 지침을 수행하고 Amazon GameLift Servers 서비스에서 정기적으로 업데이트를 요청합니다.
- 필요에 따라 게임 세션 대기열을 설정하거나 수정합니다. Amazon GameLift Servers 에이전트와 함께 배포된 Anywhere 플릿을 사용하려면 새 대기열을 생성합니다(또는 기존 대기열 업데이트).
- Anywhere 플릿과의 에이전트 통합을 테스트합니다. 에이전트가 런타임 구성을 기반으로 서버 프로세스를 올바르게 시작하는지 확인합니다.

5단계: Anywhere 플릿 사용자 지정

게임 출시를 준비할 때 관리형 호스팅 리소스를 미세 조정해야 합니다. 고려해야 할 몇 가지 결정 사항은 다음과 같습니다.

- 게임 서버 소프트웨어 설치 및 실행을 포함하여 필요에 따라 컴퓨팅을 시작하고 종료하는 프로세스를 자동화합니다. 컴퓨팅 재할용은 컴퓨팅이 정기적으로 업데이트되도록 하는 데 유용하며 컴퓨팅을 종료하면 필요하지 않을 때 비용을 절감할 수 있습니다.
- 게임 서버가 다른 AWS 리소스를 통신해야 하는 경우 IAM 역할을 설정하여 액세스를 관리합니다. [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결을\(를\)](#) 참조하세요.
- 게임 서버를 위치시킬 지리적 위치를 결정합니다. 관리형 플릿에 원격 위치를 추가합니다. [리소스 사용자 지정 호스팅을\(를\)](#) 참조하세요.
- 컴퓨팅 리소스 구성을 선택하여 플릿 성능을 최적화한 다음 컴퓨팅당 최적의 서버 프로세스 수를 실행하도록 런타임 지침을 구성합니다.
- 우선 순위 설정 사용자 지정을 포함하여 관리형 플릿에 대한 게임 세션 배치 옵션을 실험합니다. [게임 세션 대기열 사용자 지정을\(를\)](#) 참조하세요.
- 예상 플레이어 수요에 맞게 수동 또는 자동 용량 조정을 처리하는 메커니즘을 생성합니다. 시스템이 게임 세션을 호스팅하는 데 사용할 수 있는 컴퓨팅 수를 늘리거나 줄이도록 유도해야 하는 요인을 고려합니다.
- 필요한 경우 다른 리소스에 대한 장애 조치를 설계하고 구현합니다.
- 분석 및 로깅을 포함한 호스팅 관찰성 도구를 설정합니다. [Amazon GameLift Servers 모니터링](#) 섹션을 참조하세요.

Amazon GameLift Servers를 사용한 하이브리드 호스팅을 위한 개발 로드맵

이 로드맵은 멀티플레이어 게임을 위해 호스팅 솔루션을 개발하는 방법을 안내합니다. Amazon GameLift Servers는 여러 게임 호스팅 옵션을 제공합니다. 이러한 옵션에 대한 자세한 내용은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

하이브리드 솔루션은 Amazon GameLift Servers에서 관리하는 클라우드 기반 리소스와 자체 관리형 호스팅 리소스를 비롯한 호스팅 리소스의 조합을 사용합니다. 하이브리드 호스팅에 대한 자세한 내용은 [Amazon GameLift Servers Anywhere를 사용한 하이브리드 게임 서버](#) 문서를 참조하세요. Amazon GameLift Servers를 사용하면 공통 구성 요소와 프로세스를 사용하는 하이브리드 솔루션을 설정할 수 있으므로 글로벌 플릿을 중앙에서 관리하고 모든 유형의 리소스 간에 로드를 쉽게 이동할 수 있습니다.

하이브리드 아키텍처는 다음 구성 요소로 구성됩니다.

- Amazon GameLift Servers용 서버 SDK와 통합되어 모든 플릿에 배포되는 단일 게임 서버 빌드.
- Amazon GameLift Servers 서비스와 상호 작용하고 게임 세션을 요청할 수 있도록 AWS SDK와 통합된 단일 게임 클라이언트 및 백엔드 서비스.

- 모든 플릿에서 사용 가능한 게임 서버와 로드 밸런서가 있는 새 게임 세션을 배치하는 Amazon GameLift Servers 공유 대기열.
- (선택 사항) Anywhere 플릿과 함께 배포되어 모든 플릿의 컴퓨팅 전반에서 서버 프로세스 관리 작업을 간소화하는 프로세스 관리자 에이전트.
- (선택 사항) 다중 플레이어 매치를 생성하고 해당 매치에 대한 게임 세션을 설정하는 FlexMatch 매치 메이커.
- 멀티플레이어 게임 호스팅에 최적화된 Amazon Elastic Compute Cloud(Amazon EC2) 컴퓨팅 인스턴스를 사용하는 하나 이상의 Amazon GameLift Servers 관리형 플릿.
- 구성 관리 및 배포 도구를 포함하여 기존 온프레미스 또는 기타 호스팅 리소스를 사용하는 하나 이상의 Amazon GameLift Servers Anywhere 플릿. (선택 사항: AWS Systems Manager)

이 로드맵은 Amazon GameLift Servers 하이브리드 호스팅을 통해 멀티플레이어 게임을 성공적으로 시작하고 실행할 수 있는 간소화된 경로를 제공합니다. 필요한 구성 요소가 준비되면 게임 개발을 계속 반복하고 호스팅 솔루션을 사용자 지정할 수 있습니다. 출시일이 가까워지면 프로덕션 수준 사용을 위한 호스팅 솔루션을 준비하는 데 필요한 [Amazon GameLift Servers 호스팅으로 출시 준비](#) 섹션을 참조하세요.

Amazon GameLift Servers 플러그인으로 바로 시작

Unreal Engine 또는 Unity로 프로젝트를 개발 중이라면 Amazon GameLift Servers 플러그인으로 호스팅할 게임 설정을 바로 시작할 수 있습니다. 플러그인을 사용하면 Amazon GameLift Servers SDK를 게임 프로젝트에 추가하고 안내 워크플로를 사용하여 Anywhere 플릿과 Amazon GameLift Servers 관리형 플릿을 모두 사용하는 하이브리드 호스팅 솔루션의 간단한 작동 버전을 구축할 수 있습니다. 그런 다음 이러한 기본 사항을 기반으로 필요에 따라 빌드 및 사용자 지정이 가능합니다.

1단계: Amazon GameLift Servers에서 작동하도록 게임 서버 준비

게임 서버가 호스팅을 위해 배포될 때 Amazon GameLift Servers 서비스와 통신할 수 있도록 게임 서버에 기능을 추가합니다. 게임 서버가 Amazon GameLift Servers 관리형 플릿 또는 Anywhere 플릿에서 실행 중인 경우 동일한 기능이 필요합니다.

- 게임 프로젝트에 대한 Amazon GameLift Servers용 서버 SDK(버전 5.x)를 가져옵니다. 서버 SDK는 C++, C# 및 Go로 지원됩니다. [Amazon GameLift Servers용 서버 SDK를 다운로드합니다.](#)
- 게임 서버 코드를 수정하여 서버 SDK 기능을 추가합니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)를 참조하세요. 최소한 다음 작업을 수행해야 합니다.

- 코드를 추가하여 Amazon GameLift Servers SDK를 초기화하고 Amazon GameLift Servers 서비스와 WebSocket 연결을 설정합니다. 서버 SDK 작업 `InitSdk()`를 사용합니다. Anywhere 플릿 컴퓨팅에서 실행할 때 서버 파라미터를 지정하는 코드를 포함합니다.
- 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessReady()`을 사용합니다.
- 필요한 `OnProcessTerminate()`, `OnStartGameSession()` 콜백 함수를 구현합니다. 이러한 함수를 사용하면 게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 연결을 유지하고, Amazon GameLift Servers에서 메시지가 표시되면 게임 세션을 시작하고, 프롬프트에 응답하여 게임 서버 프로세스를 종료할 수 있습니다.
- 서버 프로세스가 게임 세션을 종료하면 Amazon GameLift Servers 서비스에 보고하는 코드를 추가합니다. 서버 SDK 작업 `ProcessEnding()`을 사용합니다.
- 게임 서버 빌드를 패키징합니다. 빌드 파일, 종속 항목 및 지원 소프트웨어를 사용하여 설치 스크립트를 생성합니다. [게임 빌드 파일 패키징하기](#)을(를) 참조하세요. Amazon Simple Storage Service(Amazon S3) 버킷을 사용하여 게임 빌드 버전을 저장하는 것이 좋습니다.
- 게임 서버 통합을 테스트합니다. 이 작업의 경우 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)에 설명된 대로 로컬 워크스테이션을 사용하는 Amazon GameLift Servers Anywhere 플릿을 설정하는 것이 좋습니다. 이 단계에서는 게임 서버 빌드를 테스트 디바이스에 수동으로 설치하고 서버 프로세스를 시작합니다. AWS CLI를 사용하여 새 게임 세션을 요청하고 Amazon GameLift Servers 서비스가 서버 프로세스에 게임 세션을 시작하라는 메시지를 성공적으로 표시하는지 확인합니다.

2단계: 호스팅한 게임 세션에 연결할 게임 클라이언트 준비

게임 클라이언트가 게임 세션에 연결하도록 요청하고 연결 정보를 얻은 다음 호스팅된 게임 세션에 직접 연결할 수 있는 방법을 생성합니다. 가장 일반적인 접근 방식은 게임 클라이언트와 Amazon GameLift Servers 서비스 간에 중간자 역할을 하는 백엔드 서비스 기능을 설정하는 것입니다. 이 접근 방식은 호스팅 리소스를 보호하고 플레이어가 게임 세션에 배치되는 방식을 더 잘 제어할 수 있도록 합니다.

- 호스팅을 위한 백엔드 서비스 기능을 구축합니다. 백엔드 서비스가 Amazon GameLift Servers 서비스와 통신하고 게임 클라이언트에 연결 정보를 전송합니다. 이 기능에는 게임 세션 시작, 플레이어에게 게임에 배치, 게임 세션 정보 검색이 포함됩니다. 자세한 지침은 [Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
- 용 AWS SDK를 가져와 Amazon GameLift Servers와 백엔드 서비스 프로젝트에 추가합니다. [클라이언트 서비스에 대한 Amazon GameLift Servers SDK 리소스](#)를 참조하세요.

- Amazon GameLift Servers 클라이언트를 초기화하고 키 설정을 저장하는 코드를 추가합니다. [Amazon GameLift Servers API를 설정합니다](#)을(를) 참조하세요.
- AWS SDK 작업을 호출 `CreateGameSession()` 하고 게임 세션 연결 정보를 게임 클라이언트에 제공하는 기능을 추가합니다. [특정 플릿에서 게임 세션 생성](#) 페이지를 참조하세요.

`CreateGameSession()` 호출로 간편하게 새 게임 세션 요청을 시작할 수 있습니다. 게임 세션 배치 시스템을 설치한 후(3단계 참조) 이 코드를 `StartGameSessionPlacement()`에 대한 호출(또는 FlexMatch를 사용하는 경우 `StartMatchmaking()` 코드)로 바꿉니다.

백엔드 서비스 설계에 대한 지침은 [Amazon GameLift Servers에 대한 백엔드 서비스 구축](#) 페이지를 참조하세요.

- 플레이어가 호스팅된 게임 세션에 연결할 수 있도록 게임 클라이언트에 기능을 추가합니다. 게임 클라이언트가 Amazon GameLift Servers가 아닌 백엔드 서비스에 직접 요청합니다. 백엔드 서비스가 게임 세션 연결 정보를 제공한 후 게임 클라이언트는 게임 세션과 직접 연결하여 게임을 플레이합니다.
- 게임 클라이언트 통합을 테스트합니다. 로컬 워크스테이션과 동일한 Amazon GameLift Servers Anywhere 플릿을 사용하여 테스트할 수 있습니다.

개발 단계에서 게임 빌드가 Amazon GameLift Servers 관리형 플릿에서 어떻게 작동하는지 테스트하려면 [클라우드 기반 테스트 환경](#)도 설정하는 것이 좋습니다. 이 Amazon GameLift Servers Toolkit 솔루션은 관리형 플릿의 동작을 모방하지만 최소 처리 시간으로 게임 서버 빌드를 업데이트할 수 있습니다.

3단계: 게임 세션 배치 설정

Amazon GameLift Servers가 새 게임 세션에 대한 요청을 처리하고 호스팅할 사용 가능한 게임 서버를 찾는 방법을 사용자 지정합니다. Amazon GameLift Servers는 모든 플릿에서 모든 게임 서버의 가용성을 자동으로 추적합니다. 게임 클라이언트가 게임 세션에 연결하라는 요청을 보내면 Amazon GameLift Servers는 최소 지연 시간, 비용 및 가용성과 같이 정의된 우선 순위 집합을 기반으로 '최상의' 배치를 찾습니다.

- 사용 가능한 게임 서버가 있는 새 게임 세션을 배치하기 위한 게임 세션 대기열을 생성합니다. 대기열은 게임 세션 배치의 기본 메커니즘입니다. 자세한 지침은 [게임 세션 대기열 생성](#)을 참조하세요.
- 최소한 Anywhere 플릿을 대기열의 대상으로 추가합니다. 다른 모든 설정은 선택 사항입니다.
- 백엔드 서비스 코드에서 `CreateGameSession()` 호출을 `StartGameSessionPlacement()`로 변환합니다. [다중 위치 대기열에서 게임 세션 생성](#) 페이지를 참조하세요.

- 게임 세션에 연결할 준비가 되면 게임 클라이언트에 알리는 메커니즘을 생성합니다. 개발 중에 DescribeGameSessionPlacement에 대한 호출을 사용하여 게임 세션 상태를 폴링할 수 있습니다. 그러나 대기열을 사용하여 대용량을 처리하기 전에 이벤트 알림을 활성화해야 합니다. [게임 세션 배치의 이벤트 알림 설정](#)(를) 참조하세요.
- (선택 사항) FlexMatch 매치메이킹 구성 요소를 추가합니다. 지침은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.

4단계: Amazon GameLift Servers 에이전트를 사용하여 Anywhere 플릿 설정

지금까지 로컬 디바이스(Anywhere 플릿 컴퓨팅으로 등록됨)로 작업하여 게임 구성 요소를 테스트하고 반복해 왔습니다. 다음 단계는 프로덕션 시스템에 필요한 플릿 유형을 설정하는 것입니다. Anywhere 플릿으로 시작하고 Amazon GameLift Servers 에이전트를 추가하여 일부 주요 온컴퓨트 호스트 관리 작업을 관리합니다. 자세한 내용은 [Amazon GameLift Servers 에이전트 사용법](#) 섹션을 참조하세요.

- Amazon GameLift Servers 에이전트를 가져와서 게임 서버 설치 패키지에 추가합니다. 에이전트 [Amazon GameLift Servers GitHub 리포지토리에서 사용할 수 있는 에이전트](#) 소스 코드를 가져와 빌드합니다. 결과 JAR 파일 실행 파일을 게임 빌드 실행 파일과 동일한 디렉터리에 넣습니다.
- 필요에 따라 에이전트에 대한 시작 스크립트를 수정합니다. 컴퓨팅 실행이 시작되는 즉시 에이전트 실행 파일이 시작되는지 확인합니다. 호스팅 컴퓨팅에서 에이전트를 설치하고 실행하는 방법에 대한 도움말은 에이전트 리포지토리의 readme 파일을 참조하세요. 시작 명령에는 최소한 Anywhere 플릿 ID와 사용자 지정 위치 AWS 리전 및 컴퓨팅 이름을 지정하는 옵션이 포함되어야 합니다.

에이전트는 자동으로 다음 작업을 처리하므로 스크립트를 사용하여 이러한 작업을 처리한 경우 제거할 수 있습니다.

- RegisterCompute()를 호출하여 Anywhere 플릿에 컴퓨팅을 추가합니다.
- GetComputeAuthToken()을 호출하여 Amazon GameLift Servers 서비스에 연결할 때 게임 서버를 인증합니다. 에이전트는 인증 토큰 가져오기 및 새로 고침을 관리합니다. 인증 토큰은 컴퓨팅에서 실행 중인 모든 게임 서버 프로세스에서 사용할 수 있습니다.
- 런타임 지침 세트를 기반으로 컴퓨팅에서 새 서버 프로세스를 시작합니다.
- Anywhere 플릿의 컴퓨팅에 대한 런타임 구성을 생성합니다. 최소한 게임 서버 실행 파일의 시작 경로를 지정합니다. Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 플릿에 대한 런타임 지침을 생성하거나 수정할 수 있습니다. 에이전트는 이러한 지침을 수행하고 Amazon GameLift Servers 서비스에서 정기적으로 업데이트를 요청합니다.
- 필요에 따라 게임 세션 대기열을 설정하거나 수정합니다. 새 대기열을 생성(또는 기존 대기열 업데이트)하고 Amazon GameLift Servers 에이전트와 함께 배포된 Anywhere 플릿의 대상을 지정합니다.

- Anywhere 플릿과의 에이전트 통합을 테스트합니다. 에이전트가 런타임 구성을 기반으로 서버 프로세스를 올바르게 시작하는지 확인합니다.

5단계: 관리형 클라우드 기반 플릿 생성

Amazon GameLift Servers 관리형 EC2 플릿을 생성하여 Anywhere 플릿을 보완합니다. 2단계에서 개발 속도를 높이기 위해 클라우드 기반 테스트 환경을 설정한 경우, 대부분의 게임 개발 및 테스트를 완료한 후 관리형 플릿을 생성하도록 계획합니다. 자동 용량 조정과 같은 추가 설정을 구성하고 테스트하려면 완전 관리형 플릿이 필요합니다.

- 게임 서버 빌드를 패키징하고 Amazon GameLift Servers에 업로드합니다. 빌드 파일, 종속 항목 및 지원 소프트웨어를 사용하여 설치 스크립트를 생성합니다. Anywhere 및 관리형 플릿 모두에서 동일한 빌드 소프트웨어를 사용할 수 있습니다. [Amazon GameLift Servers의 게임 서버 빌드 생성](#)(를) 참조하세요. 콘솔 또는 AWS CLI를 Amazon GameLift Servers 사용하여 빌드를 업로드할 수 있습니다.

빌드를 업로드하기 전에 관리형 플릿을 생성할 AWS 리전을 결정합니다. 빌드를 동일한 리전에 업로드해야 합니다. 플릿 위치 선택에 대한 자세한 내용은 [지리적 위치](#) 페이지를 참조하세요.

- 관리형 EC2 플릿을 생성합니다. Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 관리형 플릿을 생성할 수 있습니다. 플릿을 생성하면 Amazon GameLift Servers가 호스팅을 위해 게임 서버 빌드 배포를 즉시 시작합니다. 관리형 플릿의 여러 부분을 구성할 수 있습니다. 자세한 지침은 [Amazon GameLift Servers 관리형 EC2 플릿 생성](#)을 참조하세요. 최소한 다음 작업을 수행해야 합니다.
 - 플릿에 이름을 지정하고 업로드된 게임 빌드 중 배포할 빌드를 지정합니다.
 - 플릿의 온디맨드 인스턴스를 선택하고 플릿 위치에서 사용할 수 있는 인스턴스 유형을 선택합니다. 스팟 플릿은 중요한 옵션이지만 추가 설계 및 구성이 필요합니다.
 - Anywhere 플릿과 함께 사용한 것과 유사한 설정으로 런타임 구성을 생성합니다. 최소한 게임 서버 실행 파일의 시작 경로를 지정합니다.
 - 인바운드 트래픽이 게임 서버에 액세스할 수 있도록 포트 설정을 지정합니다.
- 공유 게임 세션 대기열에 관리형 플릿을 추가합니다. Amazon GameLift Servers 에이전트와 함께 배포된 관리형 플릿과 Anywhere 플릿의 대상을 모두 포함하도록 4단계에서 생성한 대기열을 업데이트합니다.
- 관리형 플릿으로 게임 호스팅을 테스트합니다. 이 시점에서 게임 클라이언트가 게임 세션을 요청하고, 연결 정보를 얻고, 게임 세션에 성공적으로 연결하여 전체 호스팅 주기를 테스트할 수 있어야 합니다.

6단계: 플릿 사용자 지정

게임 출시를 준비할 때 호스팅 솔루션을 미세 조정해야 합니다. 고려해야 할 몇 가지 결정 사항은 다음과 같습니다.

- Anywhere 플릿의 경우, 게임 서버 소프트웨어 설치 및 실행을 포함하여 필요에 따라 컴퓨팅을 시작하고 종료하는 프로세스를 자동화합니다. 컴퓨팅 재할용은 컴퓨팅이 정기적으로 업데이트되도록 하는 데 유용하며 컴퓨팅을 종료하면 필요하지 않을 때 비용을 절감할 수 있습니다.
- 게임 서버가 다른 AWS 리소스와 통신해야 하는 경우 IAM 역할을 설정하여 액세스를 관리합니다. [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#)(를) 참조하세요.
- 게임 서버를 위치시킬 지리적 위치를 결정합니다. 관리형 플릿에 원격 위치를 추가합니다. [리소스 사용자 지정 호스팅](#)(를) 참조하세요.
- 관리형 플릿의 경우 비용 절감을 위해 스팟 플릿을 사용하는 것이 좋습니다. [스팟 플릿으로 게임 호스팅 비용 절감](#)(를) 참조하세요.
- 컴퓨팅 리소스 구성을 선택하여 플릿 성능을 최적화한 다음 컴퓨팅당 최적의 서버 프로세스 수를 실행하도록 런타임 지침을 구성합니다. Anywhere 플릿과 관리형 플릿 모두에 대해 이 작업을 수행합니다. [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)(를) 참조하세요.
- 우선 순위 설정 사용자 지정을 포함하여 관리형 플릿에 대한 게임 세션 배치 옵션을 실험합니다. [게임 세션 대기열 사용자 지정](#)(를) 참조하세요.
- 관리형 플릿의 경우 예상 플레이어 수요에 맞게 자동 용량 조정을 설정합니다. [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)(를) 참조하세요.
- Anywhere 플릿의 경우 예상 플레이어 수요에 맞게 수동 또는 자동 용량 조정을 처리하는 메커니즘을 생성합니다.
- 필요한 경우 다른 리소스에 대한 장애 조치를 설계하고 구현합니다. 다른 AWS 리전 에서 대기 플릿을 설정하고 필요한 경우 장애 조치를 처리하도록 대기열 및 오토 스케일링을 수정합니다.
- 분석 및 로깅을 포함한 호스팅 관찰성 도구를 설정합니다. [Amazon GameLift Servers 모니터링](#)(를) 참조하세요. 지표 그룹을 생성하여 모든 호스팅 리소스에 대한 분석을 집계합니다.
- [코드형 인프라\(IaC\)](#)를 사용하여 배포를 자동화합니다. [를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation](#)(를) 참조하세요.

Amazon GameLift Servers는 배포별 구성에 대한 AWS CloudFormation 템플릿 사용을 지원합니다. AWS Cloud Development Kit (AWS CDK) 를 사용하여 Amazon GameLift Servers 리소스를 정의할 수도 있습니다. 에 대한 자세한 내용은 [AWS Cloud Development Kit \(AWS CDK\) 개발자 안내서](#)를 AWS CDK참조하세요.

CloudFormation 스택 배포를 관리하려면와 같은 지속적 통합 및 지속적 전달(CI/CD) 도구 및 서비스를 사용하는 것이 좋습니다 AWS CodePipeline. 이러한 도구를 사용해 게임 서버 바이너리를 빌드할 때마다 자동으로 배포하거나 승인을 받아 배포할 수 있습니다. CI/CD 도구 또는 서비스를 사용하면 다음과 같은 새 게임 서버 버전에 대한 리소스 배포가 가능합니다.

- 게임 서버 바이너리를 빌드하고 테스트합니다.
- 바이너리를 Amazon GameLift Servers에 업로드합니다.
- 새 빌드로 새 플릿을 배포합니다.
- 게임 세션 대기열에 새 플릿을 추가하고 이전 빌드 버전의 플릿을 제거합니다.
- 이전 빌드의 플릿이 더 이상 활성 게임 세션을 호스팅하지 않는 경우 해당 플릿의 CloudFormation 스택을 삭제합니다.

Amazon GameLift Servers를 사용하여 호스팅할 게임 준비

Amazon GameLift Servers에서 호스팅하기 위해 게임을 배포하려면 먼저 게임 서버와 게임 클라이언트 구성 요소를 모두 준비해야 합니다. 여기에는 게임 서버를 Amazon GameLift Servers 서버 SDK와 통합하고 플레이어 인증 및 게임 세션 관리를 처리하는 백엔드 서비스를 구축하는 작업이 포함됩니다.

서버 SDK 통합을 통해 게임 서버는 Amazon GameLift Servers 서비스와 통신하고, 상태를 보고하며, 게임 세션을 관리할 수 있습니다. 백엔드 서비스는 플레이어 요청을 처리하고 게임 세션 배치를 조정해주는 게임 클라이언트와 Amazon GameLift Servers 서비스 간의 중개자 역할을 합니다.

주제

- [Amazon GameLift Servers 플러그인으로 Unreal 또는 Unity 게임 준비](#)
- [Amazon GameLift Servers와 게임 서버 통합](#)
- [배포를 위해 게임 서버 빌드 패키징](#)
- [Amazon GameLift Servers와 게임 클라이언트/서버 상호 작용](#)

Amazon GameLift Servers 플러그인으로 Unreal 또는 Unity 게임 준비

Amazon GameLift Servers 플러그인은 Unreal 또는 Unity 게임 엔진에 대한 모든 기능을 갖춘 추가 기능입니다. Amazon GameLift Servers로 호스팅용 게임을 배포하기 위한 기본 단계를 안내합니다. 플러그인 도구 세트와 워크플로를 사용하면 게임 엔진 개발 환경에서 작업하여 게임 서버 호스팅을 준비하고, 테스트를 위해 로컬 머신에서 호스팅을 설정하며, 간단한 백엔드 서비스를 생성하고, 관리형 클라우드 기반 호스팅에 게임 서버를 배포할 수 있습니다.

플러그인을 사용하여 Amazon GameLift Servers 작업을 이해하고 게임 호스팅 솔루션을 빠르게 시작 및 실행할 수 있습니다. 샘플 게임 자산 또는 자체 게임 프로젝트로 작업할 수 있습니다. 플러그인은 간단한 작업 솔루션을 빠르게 구축할 수 있도록 여러 단계를 자동화합니다. 플러그인의 안내 워크플로를 완료하면 Amazon GameLift Servers를 통해 게임 클라이언트를 라이브 호스팅 게임 세션에 연결할 수 있습니다. 플러그인을 사용하여 간단한 호스팅 솔루션을 생성한 후 게임의 요구 사항에 맞게 솔루션을 사용자 지정할 수 있습니다.

플러그인은 다음 게임 엔진에 사용할 수 있습니다.

- Unreal Engine
- Unity

플러그인에는 각 게임 엔진에 대한 다음과 같은 구성 요소가 포함되어 있습니다.

- 게임 엔진 편집기용 플러그인 모듈. 플러그인이 설치되면 새 기본 메뉴 버튼을 통해 Amazon GameLift Servers 기능에 액세스할 수 있습니다.
- 클라이언트 측 기능을 갖춘 Amazon GameLift Servers 서비스 API용 라이브러리.
- Amazon GameLift Servers 서버 SDK(버전 5)용 라이브러리.
- 서버 통합 테스트에 사용할 샘플 자산
- 게임 서버 솔루션을 정의하는 CloudFormation 템플릿 형태의 편집 가능한 구성입니다.

주제

- [플러그인 워크플로](#)
- [Unreal Engine용 Amazon GameLift Servers 플러그인](#)
- [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#)
- [Unity용 Amazon GameLift Servers 플러그인\(서버 SDK 4.x\)](#)

플러그인 워크플로

다음 단계에서는 Amazon GameLift Servers에서 게임 프로젝트를 준비하고 배포하는 일반적인 경로를 설명합니다. 게임 엔진 편집기와 게임 코드에서 작업하여 이 단계를 완료합니다.

1. AWS 계정 사용자에게 연결하고 액세스 자격 증명에 사용 권한을 제공하는 사용자 프로필을 생성합니다. Amazon GameLift Servers.

2. 플러그인이 호스팅 솔루션에서 사용하는 관련 AWS 리소스를 설정합니다("부트스트래핑"이라고 함).
3. 프로젝트에 서버 코드를 추가하여 실행 중인 게임 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다.
4. 프로젝트에 클라이언트 코드를 추가하면 게임 클라이언트가 Amazon GameLift Servers에 요청을 보내 새 게임 세션을 시작한 다음 해당 세션에 연결할 수 있습니다.
5. Anywhere 워크플로를 사용하여 로컬 워크스테이션을 게임 서버의 Anywhere 컴퓨팅으로 설정하고 게임 서버를 호스트합니다. 플러그인을 통해 게임 서버와 클라이언트를 로컬로 시작하고, 게임 세션에 연결하며 통합을 테스트합니다.
6. 관리형 EC2 워크플로를 사용하여 게임 서버를 Amazon GameLift Servers에 업로드하고 단순하지만 완전한 클라우드 호스팅 솔루션을 배포합니다. 플러그인을 통해 게임 서버와 클라이언트를 로컬로 시작하고, 게임 세션을 요청해 연결한 후 게임을 실행합니다.

플러그인에서 작업할 때 AWS 리소스를 생성하고 사용합니다. 이러한 작업에는 사용 중인 AWS 계정에 요금이 발생할 수 있습니다. 를 처음 사용하는 경우 AWS이러한 작업은 [AWS 프리 티어](#)에 포함될 수 있습니다.

Unreal Engine용 Amazon GameLift Servers 플러그인

이 플러그인은 Amazon GameLift Servers C++ Server SDK 및 도구를 UE 편집기에 추가합니다. 안내 UI 워크플로를 사용하여 서버 SDK 기능을 게임 프로젝트에 통합하고 게임 서버에 Amazon GameLift Servers 호스팅 솔루션을 배포합니다.

플러그인 기능을 사용하면 기본 작업 호스팅 솔루션을 빌드한 다음 필요에 따라 최적화하고 사용자 지정할 수 있습니다. 로컬 워크스테이션을 호스트로 사용하여 Amazon GameLift Servers Anywhere 플릿을 설정합니다. 관리형 EC2 또는 관리형 컨테이너 플릿을 사용한 클라우드 호스팅의 경우 게임 세션 요청 및 클라이언트 연결을 관리할 수 있는 완전한 솔루션으로 게임 서버를 배포합니다.

주제

- [Unreal 게임 프로젝트에 대한 플러그인 설치](#)
- [다음 단계: 게임 호스팅 솔루션 사용자 지정](#)
- [Unreal용 플러그인: AWS 사용자 프로필 설정](#)
- [Unreal용 플러그인: 게임 코드 통합](#)
- [Unreal용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬에서 게임 호스팅](#)
- [Unreal용 플러그인: 관리형 EC2 플릿에 게임 배포](#)
- [Unreal용 플러그인: 관리형 컨테이너 플릿에 게임 배포](#)

Unreal 게임 프로젝트에 대한 플러그인 설치

[GitHub에서 Unreal Engine용 Amazon GameLift Servers 플러그인 가져오기](#)

게임 프로젝트의 Unreal Editor에 플러그인을 설치하는 방법에 대한 자세한 내용은 GitHub 리포지토리 Readme 파일을 참조하세요.

플러그인에는 다음과 같은 구성 요소가 포함됩니다.

- UE 편집기용 플러그인 모듈. 플러그인이 설치되면 새 기본 메뉴 버튼을 통해 Amazon GameLift Servers 기능에 액세스할 수 있습니다.
- Amazon GameLift Servers 서비스 API용 C++ 라이브러리입니다. 클라이언트 측 백엔드 서비스에서 API 기능을 사용하면 게임 클라이언트가 게임 세션을 요청하고 게임 세션 정보를 전송/검색하는 데 도움이 됩니다.
- Amazon GameLift Servers Server SDK(버전 5)용 Unreal 라이브러리. 게임 서버 코드에서 서버 SDK를 사용하여 호스팅된 게임 서버 프로세스와 Amazon GameLift Servers 서비스 간의 통신을 관리합니다.
- 서버 통합 테스트에 사용할 기본 블루프린트와 UI 요소가 포함된 스타트업 게임 맵과 테스트 맵 2개를 포함한 테스트용 콘텐츠.
- 플러그인이 호스팅을 위해 게임 서버를 배포할 때 사용하는 CloudFormation 템플릿 형태의 편집 가능한 구성입니다.

이 플러그인은 AWS CloudFormation 템플릿을 사용하여 일반적인 게임 시나리오를 위한 호스팅 솔루션을 배포합니다. 이러한 솔루션은 제공된 대로 사용하거나 게임에 필요한 대로 사용자 지정합니다.

다음 단계: 게임 호스팅 솔루션 사용자 지정

플러그인의 안내 워크플로를 사용하면 Amazon GameLift Servers 호스팅 솔루션을 빠르게 시작하고 실행할 수 있습니다. 플러그인을 사용하면 각 솔루션 구성 요소의 기본 버전을 설정할 수 있습니다.

준비가 되면 각 구성 요소를 사용자 지정하고 게임 시작을 준비할 때 솔루션을 미세 조정하여 이 기본 솔루션을 구축할 수 있습니다. 다음 옵션을 고려하세요.

- 플릿 및 플릿 구성을 수정합니다. [리소스 사용자 지정 호스팅](#)(를) 참조하세요.
- 게임 세션 대기열 구성을 사용자 지정합니다. [게임 세션 대기열 사용자 지정](#)(를) 참조하세요.
- 게임 서버 및 게임 클라이언트에 기능을 추가합니다. [Amazon GameLift Servers와 게임 서버 통합 및 Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 섹션을 참조하세요.

- 백엔드 서비스를 사용자 지정합니다. [Amazon GameLift Servers에 대한 백엔드 서비스 구축을\(를\)](#) 참조하세요.
- 예상 플레이어 수요에 맞게 자동 용량 조정을 설정합니다. [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)을(를) 참조하세요.
- 분석 및 로깅을 포함한 호스팅 관찰성 도구를 설정합니다. [Amazon GameLift Servers 모니터링](#)을(를) 참조하세요.
- [코드형 인프라\(IaC\)](#)를 사용하여 배포를 자동화합니다. 관리형 솔루션에 대한 플러그인의 안내 워크플로는 AWS CloudFormation 템플릿을 사용합니다. 필요에 따라 이를 사용자 지정할 수 있습니다. [를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation](#)을(를) 참조하세요.

주제

- [Unreal용 플러그인: AWS 사용자 프로필 설정](#)
- [Unreal용 플러그인: 게임 코드 통합](#)
- [Unreal용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬에서 게임 호스팅](#)
- [Unreal용 플러그인: 관리형 EC2 플릿에 게임 배포](#)
- [Unreal용 플러그인: 관리형 컨테이너 플릿에 게임 배포](#)

Unreal용 플러그인: AWS 사용자 프로필 설정

플러그인을 설치한 후 유효한 AWS 계정으로 사용자 프로필을 설정합니다. 프로필을 여러 개 유지할 수 있지만 한 번에 한 프로필만 선택할 수 있습니다. 플러그인에서 작업할 때마다 사용할 프로필을 선택합니다. 각 워크플로 페이지에는 현재 선택한 프로필이 표시됩니다.

프로필을 여러 개 유지하면 서로 다른 호스팅 배포 간에 전환할 수 있습니다. 예를 들어 동일한 AWS 계정을 사용하지만 다른 AWS 리전에 배포하는 프로필을 설정할 수 있습니다. 또는 다른 AWS 계정 또는 사용자 및 권한 세트로 프로필을 설정할 수 있습니다.

Note

워크스테이션에 AWS CLI를 설치하고 프로필이 이미 구성된 경우 Amazon GameLift Servers 플러그인이 CLI를 감지하여 기존 프로필로 나열합니다. 플러그인은 [default] 이름이 지정된 프로필을 자동으로 선택합니다. 기존의 데이터 세트를 사용하거나 새 프로필을 생성할 수 있습니다.

계정 사용자 아래에 일부 필수 AWS 리소스를 설정하려면 모든 프로필을 부트스트래핑해야 합니다.

AWS 프로필을 관리하려면

1. Unreal 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 AWS 자격 증명 액세스를 선택합니다. 이 작업을 수행하면 사용자 프로필 설정 페이지에 대한 Amazon GameLift Servers 플러그인이 열립니다.
2. 버튼을 사용하여 새 AWS 계정을 생성하거나 이미 보유한 AWS 계정에 대한 사용자 프로필을 설정합니다.
3. 사용자 프로필이 아직 없는 경우 프로필 세부 정보를 입력하고 새 프로필을 생성하라는 메시지가 표시됩니다. 다음 정보를 제공합니다.
 - AWS 계정. 새 AWS 계정을 생성하는 경우에 대한 링크를 사용하고 프롬프트를 AWS Management Console 따릅니다. 자세한 내용은 [AWS 계정 생성](#)을 참조하세요.
 - Amazon GameLift Servers 및 기타 필수 AWS 서비스를 사용할 권한이 있는 AWS 사용자입니다. 장기 자격 증명을 사용하여 Amazon GameLift Servers 권한 및 프로그래밍 방식으로 액세스할 수 있는 AWS Identity and Access Management (IAM) 사용자를 설정하는 방법에 [AWS 사용자 계정 설정](#) 대한 지침은 섹션을 참조하세요.
 - AWS 사용자에 대한 자격 증명입니다. 이러한 자격 증명은 AWS 액세스 키 ID와 AWS 보안 키로 구성됩니다. 자세한 내용은 [액세스 키 가져오기](#)를 참조하세요.
 - AWS region. 이는 호스팅용 AWS 리소스를 생성하려는 지리적 위치입니다. 개발 중에는 실제 위치와 가까운 리전을 사용하는 것이 좋습니다. [지원되는 리전 목록에서 AWS 리전](#)을 선택합니다.
4. 플러그인이 기존 프로필을 감지하면 사용 가능한 프로필 목록이 표시됩니다. 목록에서 기존 프로필을 선택하거나 다른 프로필 추가를 선택하여 새 프로필을 생성합니다.

사용자 프로필 부트스트랩

Amazon GameLift Servers 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.

프로필을 부트스트랩하려면

1. AWS 액세스 자격 증명 페이지에서 사용하려는 사용자 프로필의 부트스트랩 상태를 확인합니다. 프로필의 부트스트랩 상태가 '비활성'이고 나열된 S3 버킷이 없는 경우 프로필을 부트스트랩해야 합니다.

2. 사용할 프로필을 선택하고 프로필 부트스트랩을 선택합니다.
3. 부트스트랩 상태가 “활성”으로 변경될 때까지 기다립니다. 몇 분 정도 소요될 수 있습니다.

Unreal용 플러그인: 게임 코드 통합

게임 서버를 플릿에 배포하려면 먼저 Amazon GameLift Servers 서비스와 함께 사용할 게임 코드 및 패키지 게임 구성 요소에 대한 일련의 업데이트를 수행해야 합니다.

이 주제에서는 최소한의 통합을 수행하는 단계를 안내합니다. 서버 통합의 경우 제공된 코드 샘플을 사용하여 프로젝트의 게임 모드를 업데이트합니다.

- [빌드 대상 및 모듈 규칙 설정](#)
- [게임 서버 코드 업데이트](#)
- [클라이언트 게임 맵 통합](#)
- [게임 구성 요소 패키징](#)

빌드 대상 및 모듈 규칙 설정

게임 프로젝트 파일을 수정하여 Amazon GameLift Servers에 사용할 빌드 구성 요소를 올바르게 생성합니다.

클라이언트 및 서버 빌드 대상을 추가하려면:

1. 게임 프로젝트의 코드 파일을 열고 `.../Games/[your application name]Source/[your application name]Target.cs` 파일을 찾습니다. 예시: `.../Source/GameLiftUnrealAppTarget.cs`. (Visual Studio를 사용하는 경우 프로젝트의 `.sln` 파일을 엽니다.)
2. 이 파일을 복사하여 `Source/` 디렉터리에 두 개의 새 대상 파일을 생성합니다.
 - 클라이언트 대상 - 새 파일의 이름을 `[your application name]Client.Target.cs`로 변경합니다. 다음 샘플 코드에 설명한 대로 콘텐츠를 편집하여 클래스 이름과 대상 유형 값을 업데이트합니다.

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

public class GameLiftUnrealAppClientTarget : TargetRules
{
```



```

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

```

- 서버 대상 - 새 파일의 이름을 *[your application name]*Server.Target.cs로 변경합니다. 다음 샘플 코드에 설명한 대로 콘텐츠를 편집하여 클래스 이름과 대상 유형 값을 업데이트합니다.

```

using UnrealBuildTool;
using System.Collections.Generic;

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

```

3. 프로젝트 파일을 다시 생성합니다. Visual Studio를 사용하는 경우 게임 프로젝트의 .uproject 파일을 마우스 오른쪽 버튼으로 클릭하고 Visual Studio 프로젝트 파일 생성을 선택할 수 있습니다.

게임 프로젝트 모듈 규칙을 업데이트하려면:

플러그인에 종속되도록 게임 프로젝트의 모듈 규칙을 업데이트합니다.

1. 게임 프로젝트의 코드 파일을 열고 .../Games/*[your application name]*Source/*[your application name]*.Build.cs 파일을 찾습니다. 예시: .../Source/GameLiftUnrealApp.Build.cs. (Visual Studio를 사용하는 경우 프로젝트의 .sln 파일을 엽니다.)

2. ModuleRules 클래스를 찾고 다음 샘플 코드에 표시된 대로 업데이트합니다.

```
using UnrealBuildTool;

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

3. 새 대상 파일을 생성하고 모듈 규칙을 수정한 후 게임 프로젝트를 다시 빌드합니다.

게임 서버 코드 업데이트

게임 서버 코드를 업데이트하여 게임 서버 프로세스와 Amazon GameLift Servers 서비스 간에 통신할 수 있습니다. 게임 서버는 새 게임 세션을 시작하고 중지하는 등 Amazon GameLift Servers의 요청에 응답할 수 있어야 합니다.

Amazon GameLift Servers에 대한 서버 코드를 추가하려면

1. 코드 편집기에서 일반적으로 프로젝트 루트 폴더에 있는 게임 프로젝트의 솔루션(.sln) 파일을 엽니다. 예를 들어 GameLiftUnrealApp.sln입니다.
2. 솔루션이 열린 상태에서 프로젝트 게임 모드 헤더 파일인 [project-name]GameMode.h 파일을 찾습니다. 예를 들어 GameLiftUnrealAppGameMode.h입니다.

3. 헤더 파일을 다음 코드에 맞게 변경합니다. 반드시 'GameLiftServer'을 사용자 고유의 프로젝트 이름으로 변경해야 합니다. 이러한 업데이트는 게임 서버에만 적용되므로 클라이언트에서 사용할 수 있도록 원본 게임 모드 파일의 백업 복사본을 만드는 것이 좋습니다.

gameMode.h 코드 예제

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

#pragma once

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

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

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

- 관련 소스 파일인 [project-name]GameMode.cpp 파일(예: GameLiftUnrealAppGameMode.cpp)을 엽니다. 코드를 다음 예제 코드에 맞게 변경합니다. 반드시 "GameLiftUnrealApp"을 사용자 고유의 프로젝트 이름으로 변경해야 합니다. 이러한 업데이

트는 게임 서버에만 적용되므로 클라이언트에서 사용할 수 있도록 원본 파일의 백업 복사본을 만드는 것이 좋습니다.

다음 예제 코드는 Amazon GameLift Servers와 서버 통합에 필요한 최소 요소를 추가하는 방법을 보여 줍니다.

- Amazon GameLift Servers API 클라이언트를 초기화합니다. Amazon GameLift Servers 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.
- OnStartGameSession, OnProcessTerminate, onHealthCheck를 포함한 Amazon GameLift Servers 서비스의 요청에 응답하는 필수 콜백 함수를 구현합니다.
- 지정된 포트와 함께 ProcessReady()를 호출하여 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 알립니다.

게임 서버 코드 예제

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

#include "GameLiftUnrealAppGameMode.h"

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

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

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

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

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

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

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

#ifdef WITH_GAMELIFT
    InitGameLift();
#endif
}

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

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

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

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

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

        // If GameLift Anywhere is enabled, parse command line arguments and pass them
        in the ServerParameters object.
```

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

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

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

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

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

```

    }

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

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

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

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

    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
    UE_LOG(GameServerLog, Log, TEXT(">>>> WebSocket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);

```

```

        UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Aws Region: %s"),
*ServerParametersForAnywhere.m_awsRegion);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Access Key: %s"),
*ServerParametersForAnywhere.m_accessKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

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

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

    ProcessParameters = MakeShared<FProcessParameters>();

    //When a game session is created, Amazon GameLift Servers sends an activation
request to the game server and passes along the game session object containing game
properties and other settings.
    //Here is where a game server should take action based on the game session object.
    //Once the game server is ready to receive incoming player connections, it should
invoke GameLiftServerAPI.ActivateGameSession()

```



```

    ProcessParameters->OnStartGameSession.BindLambda( [=]
(Aws::GameLift::Server::Model::GameSession InGameSession)
    {
        FString GameSessionId = FString(InGameSession.GetGameSessionId());
        UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*GameSessionId);
        GameLiftSdkModule->ActivateGameSession();
    });

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

```

```

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)
{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

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

```

```
//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

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

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

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 게임에 맞는 사용자 지정 백엔드 서비스를 만들기 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 참여 등이 포함됩니다.

- 프로젝트의 Amazon Cognito 사용자 풀(배포된 Anywhere 솔루션의 일부)에서 고유한 플레이어 ID를 가져옵니다.

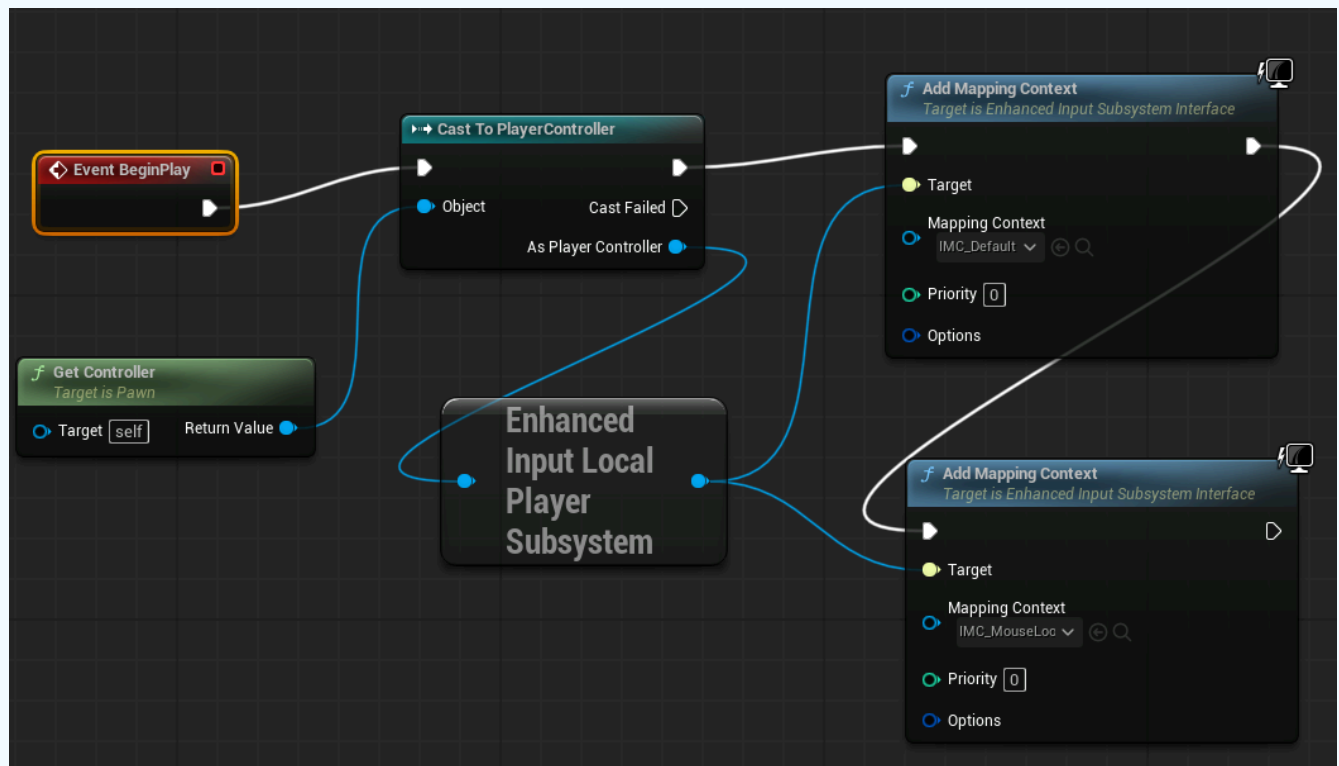
스타트업 게임 맵을 사용하려면

1. UE 편집기에서 프로젝트 설정, 맵 및 모드 페이지를 열고 기본 맵 섹션을 확장합니다.
2. 편집기 스타트업 맵의 경우 드롭다운 목록에서 'StartupMap'을 선택합니다. ... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Servers Plugin Content/Maps에 있는 파일을 검색해야 할 수도 있습니다.
3. 게임 기본 맵의 경우 드롭다운 목록에서 동일한 'StartupMap'을 선택합니다.
4. 서버 기본 맵에서 Unreal Engine 5.6 이상의 경우 "Lv1_ThirdPerson"을 선택하고 이전 버전의 경우 "ThirdPersonMap"을 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 재정의의 '없음'으로 설정합니다.
6. 기본 모드 섹션을 확장하고 전역 기본 서버 게임 모드를 서버 통합을 위해 업데이트한 게임 모드로 설정합니다.

프로젝트를 이렇게 변경했으면 게임 구성 요소를 빌드할 준비가 된 것입니다.

Note

Unreal Engine 5.6 이상의 경우 게임 서버에 연결한 후 캐릭터를 이동할 수 없는 경우 아래와 IMC_MouseLook 같이 BP_ThirdPersonCharacter 블루프린트를 업데이트하여 IMC_Default 및에 대한 입력 매핑 컨텍스트를 추가합니다.



게임 구성 요소 패키징

게임 서버 및 게임 클라이언트 빌드를 패키징하려면

1. Unreal Engine 편집기의 소스 빌드 버전에서 게임 프로젝트를 엽니다.
2. Unreal Engine 5.6 이상을 사용하는 경우 편집, 프로젝트 설정, 패키징으로 이동합니다. 프로젝트 콘텐츠 디렉터리에서 모든 Cook을 찾아 활성화합니다.
3. 편집기를 사용하여 게임 클라이언트 및 서버 빌드를 패키징합니다.
 - 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.

Unreal용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬에서 게임 호스팅

Anywhere 플러그인을 사용하여 로컬 워크스테이션을 게임 서버 호스트로 설정하기 위해 이 워크플로를 사용합니다. 클라우드 기반 관리형 플릿에 배포하기 전에 이를 사용하여 게임 서버 통합을 테스트할 수 있습니다. 반복 게임 개발 중 로컬 테스트에 유용할 수도 있습니다.

Amazon GameLift Servers Anywhere 워크플로를 시작하려면:

- Unreal 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 Anywhere와 호스트를 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 프로필의 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

사용자 프로필을 설정하려면

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

2단계: 게임 코드 설정

이 단계에서는 Amazon GameLift Servers와 함께 작동하도록 게임 서버 및 게임 클라이언트 빌드를 준비합니다. 게임 코드를 아직 통합하지 않은 경우 [Unreal용 플러그인: 게임 코드 통합](#) 섹션을 참조하세요. 로컬 워크스테이션에서 게임 실행 파일의 경로를 입력합니다.

- 게임 서버: 게임 서버를 Amazon GameLift Servers용 서버 SDK와 통합하고 게임 서버 빌드를 패키징합니다. 지침은 [Unreal용 플러그인: 게임 코드 통합](#) 섹션을 참조하세요. Amazon GameLift Servers 서비스와의 통신을 설정하고 프롬프트에 응답하여 새 게임 세션을 시작하고 게임 클라이언트 연결을 수락하려면 게임 서버를 서버 SDK와 통합해야 합니다.
- 게임 클라이언트: 최소한 IP 주소 및 포트 정보를 사용하여 게임 서버에 연결할 수 있는 게임 클라이언트가 필요합니다. 아직 게임 클라이언트 구성 요소들에 대해 설정하지 않은 경우 AWS CLI 도구를

사용하여 새 게임 세션을 수동으로 요청하고, 연결 정보를 가져오고, 해당 정보를 사용하여 게임 클라이언트를 연결할 Amazon GameLift Servers 수 있습니다.

어느 시점에 새 게임 세션 요청을 Amazon GameLift Servers 서비스에 보내고 연결 정보를 게임 클라이언트에 다시 전달하려면 백엔드 서비스가 있어야 합니다. 플러그인에 포함된 테스트 맵을 사용하여 클라이언트 Amazon GameLift Servers 기능을 게임 프로젝트에 추가할 수 있습니다. 사용자 지정 솔루션 구축에 대한 도움말은 [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 섹션을 참조하세요.

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 프로필에 대한 AWS 리전의 Anywhere 플릿만 표시됩니다.
- 기존 플릿이 없거나 새 플릿을 생성하려는 경우 새 Anywhere 플릿 생성을 선택하고 플릿 이름을 입력합니다.

프로젝트에 Anywhere 플릿을 선택하면 Amazon GameLift Servers는 플릿 상태가 활성 상태인지 확인하고 플릿 ID를 표시합니다. Unreal 편집기의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

4단계: 워크스테이션 등록

이 단계에서는 로컬 워크스테이션을 새로운 Anywhere 플릿의 컴퓨팅 리소스로 등록합니다.

워크스테이션을 Anywhere 컴퓨팅으로 등록하려면

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 시스템의 퍼블릭 IP 주소입니다. 게임 클라이언트와 서버를 같은 시스템에서 실행하는 한 localhost(127.0.0.1)를 사용할 수도 있습니다.
3. 컴퓨팅 등록을 선택합니다. Unreal 편집기의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

이 작업에 대한 응답으로 Amazon GameLift Servers는 컴퓨팅에 연결할 수 있는지 확인하고 새로 등록된 컴퓨팅에 대한 정보를 반환합니다. 또한 Amazon GameLift Servers 서비스와의 통신을 초기화할 때 게임 실행 파일에 필요한 콘솔 인수를 생성합니다.

5단계: 인증 토큰 생성

Anywhere 컴퓨팅에서 실행되는 게임 서버 프로세스에는 Amazon GameLift Servers 서비스를 호출하기 위한 인증 토큰이 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.

위에 제공된 코드 예제에서는 [AWS API 요청에 서명 버전 4\(SigV4\)](#)을 사용할 수도 있습니다. SigV4는 API 요청에 인증 정보를 추가하기 위한 AWS 서명 프로토콜입니다.

이 단계에서는 어떤 조치도 취하지 않아도 됩니다.

6단계: 게임 실행

이제 Amazon GameLift Servers를 사용하여 로컬 워크스테이션에서 멀티플레이어 게임을 시작하고 플레이하는 데 필요한 모든 작업을 완료했습니다.

호스팅 게임을 플레이하려면

1. 게임 서버를 시작합니다. 게임 서버는 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers에 알립니다.
2. 게임 클라이언트를 시작하고 새 기능을 사용하여 새 게임 세션을 시작합니다. 이 요청은 새 백엔드 서비스를 통해 Amazon GameLift Servers로 전송됩니다. 이에 대한 응답으로 Amazon GameLift Servers는 로컬 시스템에서 실행되는 게임 서버를 호출하여 새 게임 세션을 시작합니다. 게임 세션이 플레이어를 받아들일 준비가 되면 Amazon GameLift Servers는 게임 클라이언트가 게임 세션에 참여할 수 있도록 연결 정보를 제공합니다.

Unreal용 플러그인: 관리형 EC2 플릿에 게임 배포

이 워크플로에서는 플러그인을 사용하여 Amazon GameLift Servers에서 관리하는 클라우드 기반 컴퓨팅 리소스에서 호스팅하도록 게임을 수정합니다. 배포를 위해 통합 게임 서버 빌드를 Amazon GameLift Servers 서비스에 업로드합니다. 게임 코드를 아직 통합하지 않은 경우 [Unreal용 플러그인: 게임 코드 통합](#) 섹션을 참조하세요. 이 워크플로가 완료되면 클라우드의 게임 서버에 연결할 수 있도록 작동하는 게임 클라이언트가 생깁니다.

Amazon GameLift Servers 관리형 Amazon EC2 워크플로를 시작하려면

- Unreal 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 관리형 EC2와 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 프로필의 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

사용자 프로필을 설정하려면

- 사용 가능한 프로필 드롭다운 목록에서 프로필을 선택합니다. 아직 프로필이 없거나 새 프로필을 생성하려면 Amazon GameLift 메뉴로 이동하여 AWS 사용자 프로필 설정을 선택합니다.
- 부트스트랩 상태가 '활성'이 아닌 경우, 프로필 부트스트랩을 선택하고 상태가 '활성'으로 변경될 때까지 기다립니다.

2단계: 게임 코드 설정

이 단계에서는 Unreal용 Amazon GameLift Servers C++ Server SDK와 함께 작동하도록 게임 서버 및 게임 클라이언트 빌드를 준비합니다. 게임 코드를 아직 통합하지 않고 게임 클라이언트 및 서버 실행 파일을 빌드하지 않은 경우 [Unreal용 플러그인: 게임 코드 통합](#) 섹션을 참조하세요. 로컬 워크스테이션에서 게임 실행 파일의 경로를 입력합니다.

이 워크플로우의 단계에서 플러그인은 소스 기반으로 빌드된 Unreal 편집기 버전을 설정하기 위한 지침과 소스 코드로의 링크를 제공합니다. 클라이언트 및 서버 구성 요소를 빌드할 때 소스 빌드 버전을 사용해야 합니다.

서버 SDK와 통합된 게임 서버를 빌드한 후 다음 작업을 완료하여 호스팅을 위해 Amazon GameLift Servers에 업로드할 준비를 합니다.

클라우드 배포를 위해 서버 빌드를 준비하려면(Windows)

Unreal 편집기가 기본적으로 서버 빌드 파일을 저장하는 WindowsServer 폴더에 다음을 추가합니다.

- 서버 빌드 설치 스크립트를 **WindowsServer** 폴더의 루트에 복사합니다. 설치 스크립트는 플러그인 다운로드에 포함되어 있습니다. [project-name]/Plugins/Resources/

CloudFormation/extra_server_resources/install.bat 파일을 찾습니다. Amazon GameLift Servers는 이 파일을 사용하여 호스팅 컴퓨팅에 서버 빌드를 설치합니다.

2. **VC_redist.x64.exe** 파일을 **WindowsServer** 폴더 루트로 복사합니다. Unreal Engine 버전 5.6 이상을 사용하는 경우 이 단계를 건너뛸 수 있습니다. 이 파일은 Visual Studio 설치에 포함되어 있습니다. 일반적으로 C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142에 있습니다.
3. OpenSSL 라이브러리 파일을 게임 서버 빌드에 추가합니다. 게임 서버가 서버 SDK 5.3 이상과 통합된 경우 이 단계를 건너뛸 수 있습니다. 이 버전은 Unreal용 Amazon GameLift Servers 플러그인 버전 3.0 이상에 포함되어 있습니다.

OpenSSL 라이브러리를 수동으로 찾아 <YourGame>/Binaries/Win64의 게임 빌드 패키지 디렉터리에 복사합니다. Unreal Engine 5 버전에서 사용하는 것과 동일한 OpenSSL 버전을 반드시 사용해야 합니다. 잘못된 OpenSSL 라이브러리와 함께 배포된 게임 빌드는 Amazon GameLift Servers 서비스와 통신할 수 없습니다.

게임 엔진 소스에서 OpenSSL 라이브러리를 찾습니다. 위치는 개발 환경에 따라 다릅니다.

Windows의 경우:

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

Linux의 경우:

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

클라우드 배포를 위해 서버 빌드를 준비하려면(Linux)

Linux용으로 빌드된 게임 서버를 준비하는 방법에 대한 자세한 지침은 [Amazon Linux에서 Unreal Engine 5 Amazon GameLift Servers용 서버 SDK 빌드](#)를 참조하세요.

1. 작업 디렉터리를 지정하여 빌드 파일을 구성합니다. 작업 디렉터리의 구조는 각 호스팅 컴퓨팅에 있는 그대로 배포됩니다. Linux 빌드 게임 서버와 모든 종속 파일을 추가합니다.

2. 작업 디렉터리의 루트에 서버 빌드 설치 스크립트를 생성합니다. 필요한 경우 `install.sh` 파일을 생성하고 게임 서버 빌드를 올바르게 설치하는 데 필요한 명령을 추가합니다. Amazon GameLift Servers는 이 파일을 사용하여 각 EC2 호스팅 리소스에 서버 빌드를 설치합니다.
3. OpenSSL 라이브러리 파일을 게임 서버 빌드에 추가합니다. 게임 서버가 서버 SDK 5.3 이상과 통합된 경우 이 단계를 건너뛸 수 있습니다.

라이브러리를 수동으로 찾아 복사합니다. Unreal Engine 5 버전에서 사용하는 것과 동일한 OpenSSL 버전을 반드시 사용해야 합니다. 잘못된 OpenSSL 라이브러리와 함께 배포된 게임 빌드는 Amazon GameLift Servers 서비스와 통신할 수 없습니다.

- a. 게임 엔진 소스에서 OpenSSL 라이브러리를 찾습니다. 위치는 개발 환경에 따라 다릅니다.

Windows의 경우:

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

Linux의 경우:

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
 - Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1
- b. OpenSSL 라이브러리를 찾은 경우 <YourGame>/Binaries/Linux의 게임 빌드 패키지 디렉터리에 복사합니다.

3단계: 배포 시나리오 선택

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

- 단일 리전 플릿: 활성 프로파일의 기본 AWS 리전에 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 다중 플릿, 다중 위치 구조와 함께 저비용 스팟 호스팅을 사용하여 가용성을 지속적으로 유지합니다. 이 방식은 호스팅 솔루션용 매치메이커 구성 요소 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오에서는 이 솔루션의 기본 리소스를 만들고 필요에 따라 나중에 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.
- FlexMatch 매치메이킹 구성 및 매치메이킹 규칙은 플레이어 요청을 수락하고 매치를 구성하도록 설정됩니다.
- 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 + `GameSessionQueue` 이벤트를 처리하는 Lambda 함수.

4단계: 게임 파라미터 설정

이 단계에서는 업로드하기 위한 게임을 설명합니다 AWS.

- 서버 빌드 이름: 게임 서버 빌드에 의미 있는 이름을 입력합니다. AWS 는 업로드되어 배포에 사용되는 서버 빌드의 사본을 가리킬 때 이 이름을 사용합니다.
- 서버 빌드 OS: 서버가 실행되도록 빌드된 운영 체제를 입력합니다. 이는 게임을 호스팅하는 데 사용할 컴퓨팅 리소스 유형을 AWS 에 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 몇 분 정도 소요될 수 있습니다.

배포를 시작하려면 CloudFormation 배포를 선택합니다. 여기에서 게임 호스팅 상태를 추적할 수 있습니다. 자세한 내용을 알아보려면에 대한 AWS AWS 관리 콘솔에 로그인하고 이벤트 알림을 볼 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 Amazon GameLift Servers로 호스팅된 멀티플레이어 게임을 시작하고 플레이하는 데 필요한 모든 작업을 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.

Unreal용 플러그인: 관리형 컨테이너 플릿에 게임 배포

이 안내 플러그인 워크플로를 사용하여 게임 서버의 컨테이너 이미지를 생성하고 컨테이너 기반 호스팅 솔루션에 배포합니다. 게임 코드를 아직 통합하지 않은 경우 [Unreal용 플러그인: 게임 코드 통합](#) 섹션을 참조하세요. 이 워크플로를 성공적으로 완료하면 컨테이너화된 게임 서버가 클라우드에서 실행되며 플러그인을 사용하여 게임 클라이언트를 시작하고, 게임 세션에 연결하며, 게임을 플레이할 수 있습니다.

시작하기 전에

이 워크플로는 다음 작업을 완료했다고 가정합니다.

- Amazon GameLift Servers Server SDK를 게임 서버 코드에 통합합니다. 호스팅하는 게임 서버는 새 게임 세션을 시작하고 게임 세션 상태를 보고하라는 요청에 응답할 수 있도록 Amazon GameLift Servers 서비스와 통신할 수 있어야 합니다. 이 작업을 완료하지 않은 경우 먼저 플러그인 워크플로우인 Anywhere로 호스팅을 따르는 것이 좋습니다. 게임 서버 코드 준비에 대한 지침은 [게임 서버 코드 업데이트](#) 섹션을 참조하세요. 관리형 컨테이너 플릿의 경우 게임을 서버 SDK 버전 5.2 이상과 통합해야 합니다.

Note

스타트업 게임 맵을 사용하는 경우 이 작업이 이미 완료되었습니다.

- Linux에서 실행할 게임 서버 실행 파일을 패키징합니다. Windows에서 개발하고 C++ 서버 SDK 버전 5.2.x 이하 버전을 통합하는 경우 [Unreal 크로스 컴파일 툴킷](#)으로 작업해야 합니다. 또는 별도의 Linux 워크스페이스를 설정하거나 Linux용 Windows 하위 시스템(WSL)과 같은 도구를 사용할 수 있습니다.
- 게임 서버 빌드와 함께 배포할 파일을 수집합니다. 로컬 시스템에서 작업 디렉터리를 생성하여 파일을 구성합니다. 해당 파일은 게임 서버 컨테이너 이미지로 구축됩니다. 여기에는 게임 종속성, 컨테이너 시작 시 게임 서버 및 기타 프로세스를 실행하는 스크립트 등을 포함할 수 있습니다.
- 게임 서버 빌드를 위해 OpenSSL 라이브러리 파일을 추가합니다. 게임 서버가 서버 SDK 5.3 이상과 통합된 경우 이 단계를 건너뛸 수 있습니다.

OpenSSL 라이브러리를 수동으로 찾아 <YourGame>/Binaries/Win64의 게임 빌드 패키지 디렉터리에 복사합니다. Unreal Engine 5 버전에서 사용하는 것과 동일한 OpenSSL 버전을 반드시 사용해야 합니다. 잘못된 OpenSSL 라이브러리와 함께 배포된 게임 빌드는 Amazon GameLift Servers 서비스와 통신할 수 없습니다.

OpenSSL 라이브러리를 찾으려면 게임 엔진 소스를 검색합니다. 위치는 개발 환경에 따라 다릅니다.

Windows의 경우:

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

Linux의 경우:

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

- Amazon GameLift Servers를 사용한 게임 클라이언트 코드를 통합합니다. 이 작업을 완료하는 한 가지 방법은 이미 통합된 샘플 자산(플러그인에 포함됨)을 추가하는 것입니다. 게임 클라이언트 코드 준비에 대한 지침은 [클라이언트 게임 맵 통합](#) 섹션을 참조하세요.
- 로컬 컴퓨터에 Docker를 설치합니다. 플러그인이 컨테이너 이미지를 생성하여 ECR 리포지토리로 푸시하도록 하려면 이 도구를 설치해야 합니다. 또는 이러한 작업을 수동으로 수행하고 플러그인에 기존 컨테이너 이미지를 사용하도록 지시할 수 있습니다. 이미지를 수동으로 빌드하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)를 참조하세요.

Amazon GameLift Servers 관리형 컨테이너 워크플로를 시작하려면:

- Unreal 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 관리형 컨테이너를 선택합니다. 이 작업을 수행하면 관리형 컨테이너가 있는 호스트 플러그인 페이지가 열립니다. 이 페이지는 게임 서버 빌드로 컨테이너 이미지를 생성하고, 컨테이너 플릿에 배포하며, 게임을 시작하는 단계별 프로세스를 제공합니다.

0단계: 프로필 설정

이 섹션에는 현재 선택한 사용자 프로필이 표시됩니다. 현재 사용자 프로필이 이 워크플로에 사용할 프로필인지 확인합니다. 이 워크플로에서 생성하는 모든 리소스는 프로필의 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

다음과 같은 경우 선택한 사용자 프로필을 수정해야 할 수 있습니다.

- 현재 선택된 프로필이 없습니다.
- 다른 프로필을 선택하거나 새 프로필을 생성하려고 합니다.
- 선택한 프로필을 부트스트랩해야 합니다(부트스트랩 상태가 비활성인 경우).

선택한 사용자 프로필을 설정하거나 변경하려면

- Amazon GameLift Servers 메뉴에서 AWS 액세스 자격 증명 열기를 선택합니다.

1단계: 컨테이너 준비 상태 평가

게임 서버를 컨테이너 플릿에 배포하기 전에 컨테이너 이미지에 패키징하고 Amazon ECR 리포지토리에 저장해야 합니다. 플러그인은 이러한 작업을 사용자를 대신하여 완료할 수 있으며 이러한 작업을 수

동으로 수행할 수 있습니다. 이 단계에서는 컨테이너 이미지 및 ECR 리포지토리의 상태에 대한 정보를 제공합니다.

평가 질문을 사용하여 플러그인에 수행해야 할 단계를 알려줍니다.

- 새 컨테이너 이미지를 생성합니다. 이 옵션을 선택하면 다음 단계에서 게임 서버 빌드 디렉터리의 위치와 빌드 실행 파일을 묻는 메시지가 표시됩니다. 플러그인은 Dockerfile 템플릿(Amazon GameLift Servers에서 제공)을 사용하고 게임에 맞게 자동으로 구성합니다. [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)에서 템플릿을 확인할 수 있습니다. 이 옵션을 선택한 후 플러그인이 새 이미지를 저장할 위치를 지정합니다.
- 새 Amazon ECR 리포지토리를 생성하고 여기에 컨테이너 이미지를 푸시합니다. 플러그인은 선택한 사용자 프로필 AWS 리전 에서 AWS 계정 및 기본값을 사용하여 프라이빗 ECR 리포지토리를 생성합니다.
- 이전에 생성된 Amazon ECR 리포지토리에 컨테이너 이미지를 푸시합니다. 이 옵션을 선택하면 다음 단계에서 목록에서 기존 Amazon ECR 리포지토리를 선택하라는 메시지가 표시됩니다. 목록에는 선택한 사용자 프로필의 AWS 계정 및 기본값 AWS 리전 에 대한 모든 Amazon ECR 리포지토리가 포함됩니다. 퍼블릭 또는 프라이빗 리포지토리를 선택할 수 있습니다.
- 기존 컨테이너 이미지를 사용합니다. 이미지를 수동으로 빌드한 경우 Amazon GameLift Servers에서 제공하는 Dockerfile 템플릿을 사용하는 것이 좋으며, 이 템플릿은 [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)에서 사용할 수 있습니다. 이 옵션을 선택한 후 이미지의 위치를 지정합니다.
- 로컬에 저장된 Docker 생성 이미지입니다. 이 옵션을 선택하면 플러그인이 새 Amazon ECR 프라이빗 리포지토리를 생성하고 로컬 이미지 파일을 푸시합니다. 다음 단계에서는 플러그인이 이미지 파일을 찾는 데 사용하는 이미지 ID를 묻는 메시지가 표시됩니다.
- Amazon ECR 리포지토리에 이미 저장된 컨테이너 이미지입니다. 이 옵션을 선택하면 다음 단계에서 목록의 기존 Amazon ECR 리포지토리 및 이미지를 선택하라는 메시지가 표시됩니다. 목록에는 선택한 사용자 프로필의 AWS 계정 및 기본값 AWS 리전 에 대한 모든 Amazon ECR 리포지토리가 포함됩니다. 퍼블릭 또는 프라이빗 리포지토리를 선택할 수 있습니다.

2단계: 이미지 배포 구성

이 단계에서는 플러그인이 컨테이너 플릿에 컨테이너 이미지를 배포하는 데 필요한 정보를 제공합니다. 이 단계에서는 다음 정보를 요청합니다.

- 1단계에서 선택한 항목을 기반으로 한 게임 서버 빌드, 컨테이너 이미지 또는 Amazon ECR 리포지토리의 위치입니다.
- 관리형 컨테이너 배포에 사용할 시나리오입니다.

- 클라이언트 구성 출력 경로입니다. 클라이언트 빌드에서 AWS 구성이 포함된 폴더를 선택합니다. 다음 위치에서 찾습니다. `[client-build]/[project-name]/Content/CloudFormation`
- 선택적 배포 설정. 이 섹션에는 플러그인이 기본적으로 사용하는 구성 설정이 있습니다. 이를 수정하거나 기본값을 유지할 수 있습니다.
- 기본적으로 게임 이름은 게임 프로젝트 이름으로 설정됩니다. 플러그인이 생성하는 모든 AWS 리소스는 게임 이름 값을 참조합니다.
- 포트 범위, 메모리 제한, vCPU 제한은 컨테이너 플릿의 구성 설정입니다. 이러한 값을 사용자 지정하는 방법에 대한 자세한 내용은 연결 포트 범위는 [네트워크 연결 구성](#) 섹션을 참조하고 리소스 제한은 [리소스 제한 설정](#) 섹션을 참조하세요.
- 컨테이너 이미지 태그는 Amazon ECR에서 컨테이너 이미지를 분류하는 데 사용됩니다. 기본값은 `unreal-gamelift-plugin`입니다.
- Amazon ECR 리포지토리의 이름입니다. 플러그인이 ECR 리포지토리를 생성하는 경우에만 사용자 지정 이름을 제안하도록 이 필드를 편집할 수 있습니다. 기본값은 `unreal-game lift-plugin-ecr-repository`입니다.

배포 시나리오 옵션

단일 리전 컨테이너 플릿

이 시나리오에서는 게임 서버를 단일 컨테이너 플릿에 배포합니다. 이는 AWS 및 컨테이너 구성과의 서버 통합을 테스트하기 위한 좋은 시작점입니다. 다음과 같은 리소스를 배포합니다.

- Amazon GameLift Servers 컨테이너 그룹 정의는 컨테이너 플릿에서 컨테이너 이미지를 배포하고 실행하는 방법을 설명합니다.
- 별칭을 사용하여 게임 서버 컨테이너가 설치되고 실행 중인 Amazon GameLift Servers 컨테이너 플릿(온디맨드).
- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
- 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
- API Gateway에 대한 과도한 플레이어 호출을 스로틀링하기 위한 웹 액세스 제어 목록(ACL).
- 게임 세션을 요청하고 게임에 참여하는 등 게임 클라이언트를 대신하여 Amazon GameLift Servers 서비스에 요청하는 백엔드 서비스:
 - 플레이어가 게임 세션 슬롯을 요청할 수 있는 API Gateway + Lambda 함수. 이 함수는 슬롯이 확보되지 않는 경우 `CreateGameSession()`을 호출합니다.
 - 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 API Gateway + Lambda 함수.

FlexMatch가 있는 단일 리전 컨테이너 플릿

이 시나리오에서는 게임 서버를 컨테이너 플릿에 배포하고, 게임 세션 배치를 구성하며, FlexMatch 매치메이킹을 설정합니다. 이 시나리오는 호스팅 솔루션용 사용자 지정 매치메이커 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오를 사용하여 이 솔루션의 기본 리소스를 만들고 필요에 따라 나중에 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.

- Amazon GameLift Servers 컨테이너 그룹 정의는 컨테이너 플릿에서 컨테이너 이미지를 배포하고 실행하는 방법을 설명합니다.
- 별칭을 사용하여 게임 서버 컨테이너가 설치되고 실행 중인 Amazon GameLift Servers 컨테이너 플릿(온디맨드).
- FlexMatch 매치메이킹 구성 및 매치메이킹 규칙은 플레이어 요청을 수락하고 매치를 구성하도록 설정.
- Amazon GameLift Servers 게임 세션 대기열은 (실행 가능성, 비용, 플레이어 지연 시간 등을 기반으로) 가능한 최상의 호스팅 리소스를 찾고 게임 세션을 시작하여 제안된 매치에 대한 요청을 충족시킵니다.
- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
- 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
- API Gateway에 대한 과도한 플레이어 호출을 스로틀링하기 위한 웹 액세스 제어 목록(ACL).
- 게임 세션을 요청하고 게임에 참여하는 등 게임 클라이언트를 대신하여 Amazon GameLift Servers 서비스에 요청하는 백엔드 서비스:
 - 플레이어가 게임 세션 슬롯을 요청할 수 있는 API Gateway + Lambda 함수. 이 함수는 슬롯이 확보되지 않는 경우 StartMatchmaking()을 호출합니다.
 - 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 API Gateway + Lambda 함수.
- 플레이어용 매치메이킹 티켓과 게임 세션 정보를 저장하는 DynamoDB 테이블.
- GameSessionQueue 이벤트를 처리하는 Amazon SNS 주제 + Lambda 함수.

컨테이너 플릿 배포

플릿 구성이 완료되면 컨테이너 플릿 배포 버튼을 선택하여 배포를 시작합니다. 플러그인이 컨테이너 이미지를 생성하여 ECR에 푸시하고, 컨테이너 플릿에 대한 호스팅 리소스를 프로비저닝하며, 선택한 호스팅 솔루션 시나리오에 대해 플릿 및 기타 AWS 리소스를 배포하는 동안 이 프로세스는 몇 분 정도 소요될 수 있습니다.

배포를 시작하면 각 단계의 진행 상황을 추적할 수 있습니다. 구성에 따라 단계에 다음이 포함될 수 있습니다.

- 컨테이너 이미지 구성
- Amazon ECT 리포지토리 생성
- 이미지 빌드 및 Amazon ECR로 푸시
- 컨테이너 그룹 정의 생성
- 컨테이너 플릿 생성

자세한 배포 정보를 보려면 AWS Management Console에서 보기를 선택합니다. 컨테이너 플릿이 활성 상태에 도달하면 플릿은 게임 세션을 호스팅할 준비가 된 서버 프로세스로 컨테이너를 적극적으로 실행하고 있습니다.

배포가 완료되면 게임 세션을 호스팅하고 플레이어 연결을 수락할 준비가 된 작동 컨테이너 플릿이 있습니다.

진행 중인 배포는 중지할 수 없습니다. 배포가 잘못된 상태가 되거나 실패하는 경우 배포 재설정 옵션을 사용하여 다시 시작할 수 있습니다.

클라이언트 시작

이제 Amazon GameLift Servers로 호스팅된 멀티플레이어 게임을 시작하고 플레이하는 데 모든 작업을 완료했습니다. 게임을 플레이하려면 클라이언트 시작을 선택하여 게임 클라이언트의 로컬 인스턴스를 시작합니다.

- 단일 플릿 시나리오를 배포한 경우 한 명의 플레이어가 있는 게임 클라이언트의 하나의 인스턴스를 열고 서버 맵을 입력하여 이동할 수 있습니다. 또한 게임 클라이언트의 두 번째 인스턴스를 열어 동일한 서버 게임 맵에 두 번째 플레이어를 추가할 수 있습니다.
- FlexMatch 시나리오를 배포한 경우 호스팅 솔루션은 두 개 이상의 게임 클라이언트가 매치메이킹 요청을 할 때까지 기다립니다. 플레이어 한 명과 함께 게임 클라이언트의 인스턴스를 두 개 이상 엽니다. 두 플레이어가 매칭되고 매치를 위해 게임 세션에 참여하라는 메시지가 표시됩니다.

컨테이너 플릿 업데이트

관리형 컨테이너 호스팅 솔루션을 성공적으로 배포한 경우 배포 업데이트 기능을 사용할 수 있습니다. 이 옵션을 사용하면 새 플릿을 생성할 필요 없이 배포된 컨테이너 플릿의 구성 설정을 업데이트할 수 있습니다.

배포를 업데이트할 때 다른 게임 서버 빌드로 컨테이너 이미지를 배포하고, Amazon ECR 리포지토리를 변경하며, 다른 배포 시나리오를 선택하고, 선택적 구성 설정을 사용자 지정할 수 있습니다.

변경사항을 배포할 준비가 되면 업데이트를 선택합니다. 배포 업데이트에 필요한 시간은 전체 배포와 유사합니다. 자세한 배포 정보를 보려면 AWS 관리 콘솔에서 보기를 선택합니다.

배포된 리소스를 정리합니다.

더 이상 필요하지 않은 관리형 컨테이너 솔루션의 AWS 리소스를 정리하는 것이 가장 좋습니다. 리소스를 제거하지 않으면 이러한 리소스에 대한 비용이 계속 발생할 수 있습니다.

다음 리소스를 삭제합니다.

- 관리형 컨테이너 리소스 스택. 이 스택의 리소스는 선택한 배포 시나리오에 따라 달라집니다. 전체 스택을 삭제하려면 CloudFormation 콘솔을 사용합니다. Amazon GameLift Servers 플러그인에서 생성된 스택은 다음 명령 규칙을 사용합니다. `GameLiftPluginForUnreal-{GameName}-Containers`. 플러그인에서 새 관리형 컨테이너 배포를 시작하기 전에 스택 삭제 프로세스가 완료될 때까지 기다립니다. 자세한 내용은 [CloudFormation 콘솔에서 스택 삭제](#)를 참조하세요.
- Amazon ECR 리포지토리. 플러그인을 사용하여 컨테이너 이미지에 대한 리포지토리를 생성한 경우 더 이상 필요하지 않은 저장소는 삭제하는 것이 좋습니다. 관리형 컨테이너 배포를 재설정하기 전에는 리포지토리를 삭제할 필요가 없습니다. 배포를 업데이트하거나 재설정하면 플러그인은 다른 리포지토리를 사용하도록 지시되지 않는 한 동일한 리포지토리를 자동으로 사용합니다. 자세한 내용은 [Amazon ECR에서 프라이빗 리포지토리 삭제](#)를 참조하세요.

Amazon GameLift Servers Unity용 플러그인(서버 SDK 5.x)

이 플러그인은 Unity 편집기에 Amazon GameLift Servers C# 서버 SDK 및 도구를 추가합니다. 안내 UI 워크플로를 사용하여 서버 SDK 기능을 게임 프로젝트에 통합하고 게임 서버에 Amazon GameLift Servers 호스팅 솔루션을 배포합니다.

플러그인 기능을 사용하면 기본 작업 호스팅 솔루션을 빌드한 다음 필요에 따라 최적화하고 사용자 지정할 수 있습니다. 로컬 워크스테이션을 호스트로 사용하여 Amazon GameLift Servers Anywhere 플릿을 설정합니다. 관리형 EC2 또는 관리형 컨테이너 플릿을 사용한 클라우드 호스팅의 경우 게임 세션 요청 및 클라이언트 연결을 관리할 수 있는 완전한 솔루션으로 게임 서버를 배포합니다.

주제

- [Unity 게임 프로젝트에 대한 플러그인 설치](#)
- [Unity용 플러그인: AWS 사용자 프로필 설정](#)
- [Unity용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬 테스트 설정](#)
- [Unity용 플러그인: 관리형 EC2 플릿에 게임 배포](#)

- [Unity용 플러그인: 관리형 컨테이너 플릿에 게임 배포](#)

Unity 게임 프로젝트에 대한 플러그인 설치

[GitHub에서 Unity용 Amazon GameLift Servers 플러그인 가져오기](#)

게임 프로젝트의 플러그인을 설치하는 방법에 대한 자세한 내용은 GitHub 리포지토리 Readme 파일을 참조하세요.

플러그인에는 다음과 같은 구성 요소가 포함됩니다.

- Unity 편집기용 플러그인 모듈 플러그인이 설치되면 새 기본 메뉴 항목을 통해 Amazon GameLift Servers 기능에 액세스할 수 있습니다.
- 클라이언트 측 기능을 갖춘 Amazon GameLift Servers 서비스 API용 C# 라이브러리.
- Amazon GameLift Servers 서버 SDK(버전 5.x)용 C# 라이브러리.
- 에셋 및 씬을 포함한 게임 콘텐츠를 샘플링하여 빌드 지원 멀티플레이어 게임이 없더라도 Amazon GameLift Servers를 사용해볼 수 있습니다.
- 플러그인이 호스팅을 위해 게임 서버를 클라우드에 배포할 때 사용하는 CloudFormation 템플릿으로 제공되는 솔루션 구성입니다.

이 플러그인은 AWS CloudFormation 템플릿을 사용하여 일반적인 게임 시나리오를 위한 호스팅 솔루션을 배포합니다. 이러한 솔루션은 제공된 대로 사용하거나 게임에 필요한 대로 사용자 지정합니다.

Unity용 플러그인: AWS 사용자 프로필 설정

플러그인을 설치한 후 유효한 AWS 계정으로 사용자 프로필을 설정합니다. 프로필을 여러 개 유지할 수 있지만 한 번에 한 프로필만 선택할 수 있습니다. 플러그인에서 작업할 때마다 사용할 프로필을 선택합니다. 각 워크플로 페이지에는 현재 선택한 프로필이 표시됩니다.

프로필을 여러 개 유지하면 서로 다른 호스팅 배포 간에 전환할 수 있습니다. 예를 들어 동일한 AWS 계정을 사용하지만 다른 AWS 리전에 배포하는 프로필을 설정할 수 있습니다. 또는 AWS 계정이나 사용자 및 권한 세트가 다른 프로필을 설정할 수 있습니다.

Note

워크스테이션에 AWS CLI를 설치하고 프로필이 이미 구성된 경우 Amazon GameLift Servers 플러그인이 CLI를 감지하여 기존 프로필로 나열합니다. 플러그인은 [default] 이름이 지정

된 프로필을 자동으로 선택합니다. 기존의 데이터 세트를 사용하거나 새 프로필을 생성할 수 있습니다.

계정 사용자 아래에 일부 필수 AWS 리소스를 설정하려면 모든 프로필을 부트스트래핑해야 합니다.

AWS 프로필을 관리하려면

1. Unity 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 AWS 자격 증명 액세스를 선택합니다. 이 작업을 수행하면 사용자 프로필 설정 페이지에 대한 Amazon GameLift Servers 플러그인이 열립니다.
2. 버튼을 사용하여 새 AWS 계정을 생성하거나 이미 보유한 AWS 계정에 대한 사용자 프로필을 설정합니다.
3. 사용자 프로필이 아직 없는 경우 프로필 세부 정보를 입력하고 새 프로필을 생성하라는 메시지가 표시됩니다. 다음 정보를 제공합니다.
 - AWS 계정. 새 AWS 계정을 생성하는 경우에 대한 링크를 사용하고 프롬프트를 AWS Management Console 따릅니다. 자세한 내용은 [AWS 계정 생성](#)을 참조하세요.
 - Amazon GameLift Servers 및 기타 필수 AWS 서비스를 사용할 권한이 있는 AWS 사용자입니다. 장기 자격 증명을 사용하여 Amazon GameLift Servers 권한 및 프로그래밍 방식으로 액세스할 수 있는 AWS Identity and Access Management (IAM) 사용자를 설정하는 방법에 [AWS 사용자 계정 설정](#) 대한 지침은 섹션을 참조하세요.
 - AWS 사용자에게 대한 자격 증명입니다. 이러한 자격 증명은 AWS 액세스 키 ID와 AWS 보안 키로 구성됩니다. 자세한 내용은 [액세스 키 가져오기](#)를 참조하세요.
 - AWS region. 호스팅을 위해 AWS 리소스를 생성하려는 지리적 위치입니다. 개발 중에는 실제 위치와 가까운 리전을 사용하는 것이 좋습니다. [지원되는 리전 목록에서 AWS 리전](#)을 선택합니다.
4. 플러그인이 기존 프로필을 감지하면 사용 가능한 프로필 목록이 표시됩니다. 목록에서 기존 프로필을 선택하거나 다른 프로필 추가를 선택하여 새 프로필을 생성합니다.

사용자 프로필 부트스트랩

Amazon GameLift Servers 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.

프로필을 부트스트랩하려면

1. AWS 액세스 자격 증명 페이지에서 사용하려는 사용자 프로필의 부트스트랩 상태를 확인합니다. 프로필의 부트스트랩 상태가 '비활성'이고 나열된 S3 버킷이 없는 경우 프로필을 부트스트랩해야 합니다.
2. 사용할 프로필을 선택하고 프로필 부트스트랩을 선택합니다.
3. 부트스트랩 상태가 "활성"으로 변경될 때까지 기다립니다. 몇 분 정도 소요될 수 있습니다.

Unity용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬 테스트 설정

이 워크플로에서는 Amazon GameLift Servers 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 Servers Anywhere 워크플로를 시작하려면:

- Unity 편집기 기본 도구 모음에서 Amazon GameLift Servers > Anywhere와 호스트를 선택합니다. 이 작업을 수행하면 @Anywhere 플릿으로 게임을 설정할 수 있는 플러그인 페이지가 열립니다. 이 페이지에는 게임 구성 요소를 통합, 빌드 및 시작하는 5단계 프로세스가 나와 있습니다.

프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필의 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 생성하려면 Amazon GameLift Servers 메뉴로 이동하여 AWS 계정 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 "활성"이 아닌 경우, 프로필 부트스트랩을 선택하고 상태가 "활성"으로 변경될 때까지 기다립니다.

C# 서버 SDK를 게임 코드에 통합

Note

샘플 게임을 가져온 경우 이 단계를 건너뛸 수 있습니다. 샘플 게임 에셋에는 이미 필요한 서버 및 클라이언트 코드가 있습니다.

워크플로의 이 단계에서는 게임 프로젝트의 클라이언트 및 서버 코드를 업데이트합니다.

- 게임 서버는 Amazon GameLift Servers 서비스와 통신하여 게임 세션을 시작하고, 게임 세션 연결 정보를 제공하고, 상태를 보고하라는 프롬프트를 받을 수 있어야 합니다.
- 게임 클라이언트는 게임 세션에 대한 정보를 얻고, 게임 세션에 연결하거나 시작하고, 게임에 연결하기 위한 연결 정보를 얻을 수 있어야 합니다.

서버 코드 통합

사용자 지정 씬에서 자체 게임 프로젝트를 사용하는 경우 제공된 샘플 코드를 사용하여 게임 프로젝트에 필요한 서버 코드를 추가합니다.

1. 게임 프로젝트 파일에서 Assets/Scripts/Server 폴더를 엽니다. 파일이 없으면 새로 생성합니다.
2. GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity)로 이동하여 Samples~/SampleGame/Assets/Scripts/Server 경로를 엽니다.
3. GameLiftServer.cs 파일을 찾아 게임 프로젝트의 Server 폴더에 복사합니다. 서버 실행 파일을 빌드할 때 이 파일을 빌드 대상으로 사용합니다.

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

- Amazon GameLift Servers API 클라이언트를 초기화합니다. Amazon GameLift Servers Anywhere 플릿에는 서버 파라미터를 사용한 InitSDK() 호출이 필요합니다. 이러한 설정은 플러그인에서 사용하도록 자동으로 설정됩니다.
- OnStartGameSession, OnProcessTerminate, onHealthCheck를 포함한 Amazon GameLift Servers 서비스의 요청에 응답하는 필수 콜백 함수를 구현합니다.
- 지정된 포트와 함께 ProcessReady()를 호출하여 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 알립니다.

샘플 서버 코드를 사용자 지정하려면 다음 리소스를 참조하세요.

- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)
- [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)

클라이언트 코드 통합

사용자 지정 실행과 함께 자체 게임 프로젝트를 사용하는 경우 기본 기능을 게임 클라이언트에 통합해야 합니다. 또한 플레이어가 게임 세션에 로그인하고 연결할 수 있도록 UI 요소를 추가해야 합니다. Amazon GameLift Servers (AWS SDK에서)용 서비스 API를 사용하여 게임 세션 정보를 가져오거나, 새 게임 세션을 생성하거나, 기존 게임 세션에 참여합니다.

Anywhere 플릿을 사용하여 로컬 테스트를 위한 클라이언트를 구축할 때 Amazon GameLift Servers 서비스에 직접 호출을 추가할 수 있습니다. 클라우드 호스팅을 위한 게임을 개발할 때 또는 프로덕션 호스팅을 위해 Anywhere 플릿을 사용할 계획인 경우 게임 클라이언트와 Amazon GameLift Servers 서비스 간의 모든 통신을 처리하기 위해 클라이언트 측 백엔드 서비스를 생성해야 합니다.

Amazon GameLift Servers를 클라이언트 코드에 통합하려면 다음 리소스를 가이드로 사용하세요.

- GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](#)에서 클라이언트를 GameLiftCoreApi 클래스와 통합합니다. 이 클래스는 플레이어 인증 및 게임 세션 정보 검색에 대한 제어를 제공합니다.
- GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](#), `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`에서 사용할 수 있는 샘플 게임 통합을 봅니다.
- Unity 게임 클라이언트에 Amazon GameLift Servers 추가의 지침을 따릅니다.

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

- FleetId - Anywhere 플릿의 고유 식별자입니다.
- FleetLocation - Anywhere 플릿의 사용자 지정 위치입니다.
- AwsRegion - Anywhere 플릿이 호스팅되는 AWS 리전입니다. 사용자 프로필에 설정한 리전입니다.
- ProfileName - 용 AWS SDK에 대한 액세스를 허용하는 로컬 시스템의 AWS 자격 증명 프로파일 Amazon GameLift Servers입니다. 게임 클라이언트는 이러한 자격 증명을 사용하여 Amazon GameLift Servers 서비스에 대한 요청을 인증합니다.

Note

자격 증명 프로파일은 플러그인에서 생성되어 로컬 시스템에 저장됩니다. 따라서 로컬 시스템(또는 프로파일 동일한 시스템)에서 클라이언트를 실행해야 합니다.

Anywhere 플릿 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 프로필에 대한 AWS 리전의 Anywhere 플릿만 표시됩니다.
- 기존 플릿이 없거나 새 플릿을 생성하려는 경우 새 Anywhere 플릿 생성을 선택하고 플릿 이름을 입력합니다.

프로젝트에 Anywhere 플릿을 선택하면 Amazon GameLift Servers는 플릿 상태가 활성 상태인지 확인하고 플릿 ID를 표시합니다. Unity 편집기의 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. Unity 편집기의 출력 로그에서 이 요청의 진행 상황을 추적할 수 있습니다.

플러그인이 IP 주소가 localhost(127.0.0.1)로 설정된 로컬 워크스테이션을 등록합니다. 이 설정은 동일한 시스템에서 게임 클라이언트와 서버를 실행한다고 가정합니다.

이 작업에 대한 응답으로 Amazon GameLift Servers는 컴퓨팅에 연결할 수 있는지 확인하고 새로 등록된 컴퓨팅에 대한 정보를 반환합니다.

게임 시작

이 단계에서는 게임 구성 요소를 빌드하고 시작하여 게임을 플레이합니다. 다음 단계를 완료합니다.

1. 게임 클라이언트를 구성합니다. 이 단계에서는 플러그인에 게임 프로젝트의 GameLiftClientSettings 에셋을 업데이트하라는 메시지를 표시합니다. 플러그인은 이 에셋을 사용하여 게임 클라이언트가 Amazon GameLift Servers 서비스에 연결하는 데 필요한 특정 정보를 저장합니다.
 - a. 샘플 게임을 가져오고 초기화하지 않은 경우 새 GameLiftClientSettings 에셋을 생성합니다. Unity 편집기 기본 메뉴에서 Assets, Create, Amazon GameLift, Client Settings를 선택

합니다. 프로젝트에 GameLiftClientSettings 사본을 여러 개 생성하면 플러그인이 이를 자동으로 감지하고 플러그인이 업데이트할 에셋을 알려줍니다.

- b. 게임 시작에서 클라이언트 구성: Anywhere 설정 적용을 선택합니다. 이 작업은 방금 설정한 Anywhere 플릿을 사용하도록 게임 클라이언트 설정을 업데이트합니다.
2. 게임 클라이언트를 빌드하고 실행합니다.
 - a. 표준 Unity 빌드 프로세스를 사용하여 클라이언트 실행 파일을 빌드합니다. File, Build Settings에서 플랫폼을 Windows, Mac, Linux로 전환합니다. 샘플 게임을 가져와 설정을 초기화하면 빌드 목록과 빌드 대상이 자동으로 업데이트됩니다.
 - b. 새로 빌드된 게임 클라이언트 실행 파일의 인스턴스를 하나 이상 시작합니다.
3. Anywhere 플릿에서 게임 서버를 시작합니다. Editor에서 서버: 서버 시작을 선택합니다. 이 작업은 Unity 편집기가 열려 있는 한 클라이언트가 연결할 수 있는 라이브 서버를 시작합니다.
4. 게임 세션을 시작하거나 연결합니다. 게임 클라이언트 인스턴스에서 UI를 사용하여 각 클라이언트가 게임 세션에 연결합니다. 클라이언트에 기능을 추가한 방법에 따라 작동 방식은 달라집니다.

샘플 게임 클라이언트를 사용하는 경우 다음과 같은 특성이 있습니다.

- 플레이어 로그인 구성 요소 Anywhere 플릿의 게임 서버에 연결할 때는 플레이어 검증이 없습니다. 아무 값이나 입력하여 게임 세션에 연결할 수 있습니다.
- 간단한 게임 연결 UI 클라이언트가 게임에 연결하려고 하면 클라이언트가 사용 가능한 플레이어 슬롯이 있는 활성 게임 세션을 자동으로 찾습니다. 게임 세션을 사용할 수 없는 경우 클라이언트가 새 게임 세션을 요청합니다. 게임 세션을 사용할 수 있는 경우 클라이언트가 사용 가능한 게임 세션에 연결하도록 요청합니다. 동시에 여러 클라이언트로 게임을 테스트할 때 첫 번째 클라이언트는 게임 세션을 시작하고 나머지 클라이언트는 자동으로 기존 게임 세션에 연결합니다.
- 플레이어 슬롯 4개가 있는 게임 세션 최대 4개의 게임 클라이언트 인스턴스를 동시에 시작할 수 있으며 동일한 게임 세션에 연결합니다.

서버 실행 파일에서 시작(선택 사항)

Anywhere 플릿에서 테스트할 게임 서버 실행 파일을 빌드하고 시작할 수 있습니다.

1. 표준 Unity 빌드 프로세스를 사용하여 서버 실행 파일을 빌드합니다. 파일, 빌드 설정에서 플랫폼을 전용 서버로 전환하고 빌드합니다.
2. Anywhere 플릿 ID 및 AWS 리전을 사용하여 AWS CLI 명령 [get-compute-auth-token](#)을 호출하여 단기 인증 토큰을 가져옵니다. 플릿을 생성할 때 플릿 ID가 Connect to a Anywhere Fleet에 표시됩니다. 활성 프로필을 선택하면 AWS 리전이 프로필 설정에 표시됩니다.

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

3. 명령줄에서 새로 빌드된 게임 서버 실행 파일을 시작하고 유효한 인증 토큰을 전달합니다.

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

Unity용 플러그인: 관리형 EC2 플릿에 게임 배포

이 워크플로에서는 플러그인을 사용하여 Amazon GameLift Servers에서 관리하는 클라우드 기반 컴퓨팅 리소스에서 호스팅하도록 게임을 준비합니다. Amazon GameLift Servers 기능을 위한 클라이언트 및 서버 게임 코드를 추가한 다음, 서버 빌드를 Amazon GameLift Servers 서비스에 업로드하여 호스팅합니다. 이 워크플로가 완료되면 클라우드에서 실행하는 게임 서버가 생기고 서버를 연결할 게임 클라이언트가 작동합니다.

Amazon GameLift Servers 관리형 Amazon EC2 워크플로를 시작하려면

- Unity 편집기 기본 도구 모음에서 Amazon GameLift Servers > 관리형 EC2와 호스트를 선택합니다. 이 워크플로는 게임 구성 요소를 통합, 빌드 및 시작하는 6단계 프로세스로 이루어집니다.

프로필 설정

이 워크플로를 따를 때 사용할 프로필을 선택합니다. 선택한 프로필은 워크플로의 모든 단계에 영향을 줍니다. 생성하는 모든 리소스는 프로필의 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 생성하려면 Amazon GameLift Servers 메뉴로 이동하여 AWS 계정 프로필 설정을 선택합니다.
2. 부트스트랩 상태가 “활성”이 아닌 경우, 프로필 부트스트랩을 선택하고 상태가 “활성”으로 변경될 때까지 기다립니다.

게임을 Amazon GameLift Servers에 통합

이 작업에서는 게임 프로젝트의 클라이언트 및 서버 코드를 업데이트합니다.

- 게임 서버는 Amazon GameLift Servers 서비스와 통신하여 게임 세션을 시작하고, 게임 세션 연결 정보를 제공하고, 상태를 보고하라는 프롬프트를 받을 수 있어야 합니다.

- 게임 클라이언트는 게임 세션에 대한 정보를 얻고, 게임 세션에 연결하거나 시작하고, 게임에 연결하기 위한 연결 정보를 얻을 수 있어야 합니다.

Note

샘플 게임을 가져온 경우 이 단계를 건너뛸 수 있습니다. 샘플 게임 에셋에는 이미 필요한 서버 및 클라이언트 코드가 있습니다.

서버 코드 통합

사용자 지정 씬에서 자체 게임 프로젝트 사용 시 제공된 샘플 코드를 사용하여 게임 프로젝트에 필요한 서버 코드를 추가합니다. Anywhere 플릿을 사용하여 테스트할 게임 프로젝트를 통합한 경우 이 단계의 지침은 이미 완료된 상태입니다.

1. 게임 프로젝트 파일에서 Assets/Scripts/Server 폴더를 엽니다. 파일이 없으면 새로 생성합니다.
2. GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity)로 이동하여 Samples~/SampleGame/Assets/Scripts/Server 경로를 엽니다.
3. GameLiftServer.cs 파일을 찾아 게임 프로젝트의 Server 폴더에 복사합니다. 서버 실행 파일을 빌드할 때 이 파일을 빌드 대상으로 사용합니다.

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

- Amazon GameLift Servers API 클라이언트를 초기화합니다. Amazon GameLift Servers Anywhere 플릿에는 서버 파라미터를 사용한 InitSDK() 호출이 필요합니다. 이러한 설정은 플러그인에서 사용하도록 자동으로 설정됩니다.
- OnStartGameSession, OnProcessTerminate, onHealthCheck를 포함한 Amazon GameLift Servers 서비스의 요청에 응답하는 필수 콜백 함수를 구현합니다.
- 지정된 포트와 함께 ProcessReady()를 호출하여 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 알립니다.

샘플 서버 코드를 사용자 지정하려면 다음 리소스를 참조하세요.

- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)

• [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)

클라이언트 코드 통합

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

관리형 EC2 플릿에서 호스팅하기 위한 플러그인 워크플로에서 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 세트 이러한 구성 요소는 API 게이트웨이를 프록시로 사용합니다.
- 고유한 플레이어 ID를 생성하고 플레이어 연결을 인증하는 Amazon Cognito 사용자 풀

이러한 구성 요소를 사용하려면 게임 클라이언트가 다음을 수행하기 위해 백엔드 서비스에 요청을 보내는 기능이 필요합니다.

- AWS Cognito 사용자 풀에서 플레이어 사용자를 생성하고 인증합니다.
- 게임 세션에 연결하고 연결 정보를 수신합니다.
- 매치메이킹을 사용하여 게임에 연결합니다.

다음 리소스를 가이드로 사용합니다.

- GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](#)에서 클라이언트를 [GameLiftCoreApi](#) 클래스와 통합합니다. 이 클래스는 플레이어 인증 및 게임 세션 정보 검색에 대한 제어를 제공합니다.
- 샘플 게임 통합에 대한 내용은 GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](#), Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs에서 확인하세요.
- [Amazon GameLift Servers 게임 클라이언트 기능 통합](#).

(Unity 6.3 이상만 해당) install.sh 스크립트 통합

Linux의 Unity 6.3에는 glibc 2.35가 필요합니다. Amazon Linux 2023에는 glibc 2.34가 포함되어 있습니다. 소스에서 glibc 2.35를 빌드하고 glibc 2.35를 사용하도록 Unity 바이너리를 구성하는 작업을 처리하는 install.sh 스크립트가 제공됩니다.

1. GitHub 리포지토리 [aws/amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity)로 이동하여 ExtraServerResources 경로를 엽니다.
2. 파일을 찾아 서버 빌드가 포함된 폴더에 install.sh 복사합니다.
3. 기존에 있는 경우 두 스크립트의 내용을 단일 로 install.sh 병합합니다 install.sh.

배포 시나리오 선택

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

- 단일 리전 플릿: 활성 프로파일의 기본 AWS 리전에 있는 단일 호스팅 리소스 플릿에 게임 서버를 배포합니다. 이 시나리오는 AWS 및 서버 빌드 구성과의 서버 통합을 테스트하기 위한 좋은 출발점입니다. 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 다중 플릿, 다중 위치 구조와 함께 저비용 스팟 호스팅을 사용하여 가용성을 지속적으로 유지합니다. 이 방식은 호스팅 솔루션용 매치메이커 구성 요소 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오에서는 이 솔루션의 기본 리소스를 만들고 필요에 따라 나중에 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.
 - FlexMatch 매치메이킹 구성 및 매치메이킹 규칙은 플레이어 요청을 수락하고 매치를 구성하도록 설정됩니다.
 - 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 + GameSessionQueue 이벤트를 처리하는 Lambda 함수.

게임 파라미터 설정

이 단계에서는 업로드할 게임을 설명합니다 AWS.

- 게임 이름: 게임 프로젝트에 유의미한 이름을 제공합니다. 이 이름은 플러그인 내에서 사용됩니다.
- 플릿 이름: 관리형 EC2 플릿에 의미 있는 이름을 제공합니다. Amazon GameLift Servers는 AWS 콘솔에 리소스를 나열할 때 이 이름(플릿 ID와 함께)을 사용합니다.
- 빌드 이름: 서버 빌드에 의미 있는 이름을 제공합니다. 이 이름을 AWS 사용하여 업로드 Amazon GameLift Servers되고 배포에 사용되는 서버 빌드의 사본을 참조합니다.
- 시작 파라미터: 관리형 EC2 플릿 인스턴스에서 서버 실행 파일을 시작할 때 실행할 선택적 지침을 입력합니다. 최대 길이는 1024자입니다.
- 게임 서버 폴더: 서버 빌드를 포함하는 로컬 폴더의 경로를 제공합니다.
- 게임 서버 파일: 서버 실행 파일 이름을 지정합니다.

시나리오 배포

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 몇 분 정도 소요될 수 있습니다.

배포를 시작하려면 CloudFormation 배포를 선택합니다. 여기에서 게임 호스팅 상태를 추적할 수 있습니다. 자세한 내용을 알아보려면에 대한 AWS AWS 관리 콘솔에 로그인하고 이벤트 알림을 볼 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 서버를 실행하고 게임 세션을 호스팅할 수 있습니다. 이제 클라이언트를 빌드하고 시작하여 게임 세션을 할 수 있습니다.

1. 게임 클라이언트를 구성합니다. 이 단계에서는 플러그인에 게임 프로젝트의 GameLiftClientSettings 에셋을 업데이트하라는 메시지를 표시합니다. 플러그인은 이 에셋을 사용하여 게임 클라이언트가 Amazon GameLift Servers 서비스에 연결하는 데 필요한 특정 정보를 저장합니다.
 - a. 샘플 게임을 가져오고 초기화하지 않은 경우 새 GameLiftClientSettings 에셋을 생성합니다. Unity 편집기 기본 메뉴에서 Assets, Create, Amazon GameLift, Client Settings를 선택합니다. 프로젝트에 GameLiftClientSettings 사본을 여러 개 생성하면 플러그인이 이를 자동으로 감지하고 플러그인이 업데이트할 에셋을 알려줍니다.
 - b. 게임 시작에서 클라이언트 구성: 관리형 EC2 설정 적용을 선택합니다. 이 작업은 방금 배포한 관리형 EC2 플릿을 사용하도록 게임 클라이언트 설정을 업데이트합니다.
2. 게임 클라이언트를 빌드합니다. 표준 Unity 빌드 프로세스를 사용하여 클라이언트 실행 파일을 빌드합니다. File, Build Settings에서 플랫폼을 Windows, Mac, Linux로 전환합니다. 샘플 게임을 가져와 설정을 초기화하면 빌드 목록과 빌드 대상이 자동으로 업데이트됩니다.
3. 새로 빌드된 게임 클라이언트 실행 파일을 시작합니다. 게임 플레이를 시작하려면 2~4개의 클라이언트 인스턴스를 시작하고 각각의 UI로 게임 세션에 연결합니다.

샘플 게임 클라이언트를 사용하는 경우 다음과 같은 특성이 있습니다.

- 플레이어 로그인 구성 요소 Anywhere 플릿의 게임 서버에 연결할 때는 플레이어 검증이 없습니다. 아무 값이나 입력하여 게임 세션에 연결할 수 있습니다.
- 간단한 게임 연결 UI 클라이언트가 게임에 연결하려고 하면 클라이언트가 사용 가능한 플레이어 슬롯이 있는 활성 게임 세션을 자동으로 찾습니다. 게임 세션을 사용할 수 없는 경우 클라이언트가 새 게임 세션을 요청합니다. 게임 세션을 사용할 수 있는 경우 클라이언트가 사용 가능한 게임 세션에 연결하도록 요청합니다. 동시에 여러 클라이언트로 게임을 테스트할 때 첫 번째 클라이언트는 게임 세션을 시작하고 나머지 클라이언트는 자동으로 기존 게임 세션에 연결합니다.
- 플레이어 슬롯 4개가 있는 게임 세션 최대 4개의 게임 클라이언트 인스턴스를 동시에 시작할 수 있으며 동일한 게임 세션에 연결합니다.

Unity용 플러그인: 관리형 컨테이너 플릿에 게임 배포

이 안내 플러그인 워크플로를 사용하여 게임 서버의 컨테이너 이미지를 생성하고 컨테이너 기반 호스팅 솔루션에 배포합니다. 이 워크플로를 성공적으로 완료하면 컨테이너화된 게임 서버가 클라우드에서 실행되며 플러그인을 사용하여 게임 클라이언트를 시작하고, 게임 세션에 연결하며, 게임을 플레이할 수 있습니다.

시작하기 전에

이 워크플로는 다음 작업을 완료했다고 가정합니다.

- 게임 서버 코드를 Amazon GameLift Servers 서버 SDK와 통합합니다. 호스팅하는 게임 서버는 새 게임 세션을 시작하고 게임 세션 상태를 보고하라는 요청에 응답할 수 있도록 Amazon GameLift Servers 서비스와 통신할 수 있어야 합니다. 이 작업을 완료하지 않은 경우 먼저 플러그인 워크플로 우인 Anywhere로 호스팅을 따르는 것이 좋습니다. 게임 서버 코드 준비에 대한 지침은 [서버 코드 통합](#) 섹션을 참조하세요. 관리형 컨테이너 플릿의 경우 게임을 서버 SDK 버전 5.2 이상과 통합해야 합니다.

Note

샘플 게임을 가져온 경우 이 작업이 이미 완료됩니다.

- Linux에서 실행할 게임 서버 실행 파일을 패키징합니다.
- 게임 서버 빌드와 함께 배포할 파일을 수집합니다. 로컬 시스템에서 작업 디렉터리를 생성하여 파일을 구성합니다. 해당 파일은 게임 서버 컨테이너 이미지로 구축됩니다. 여기에는 게임 종속성, 컨테이너 시작 시 게임 서버 및 기타 프로세스를 실행하는 스크립트 등을 포함할 수 있습니다.
- Amazon GameLift Servers를 사용한 게임 클라이언트 코드를 통합합니다. 이 작업을 완료하는 한 가지 방법은 이미 통합된 샘플 자산(플러그인에 포함됨)을 추가하는 것입니다. 게임 클라이언트 코드 준비에 대한 지침은 [클라이언트 코드 통합](#) 섹션을 참조하세요.
- 로컬 컴퓨터에 Docker를 설치합니다. 플러그인이 컨테이너 이미지를 생성하여 ECR 리포지토리로 푸시하도록 하려면 이 도구를 설치해야 합니다. 또는 이러한 작업을 수동으로 수행하고 플러그인에 기존 컨테이너 이미지를 사용하도록 지시할 수 있습니다. 이미지를 수동으로 빌드하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)를 참조하세요.

Amazon GameLift Servers 관리형 컨테이너 워크플로를 시작하려면:

- Unity 편집기 기본 도구 모음에서 Amazon GameLift Servers 메뉴를 선택하고 관리형 컨테이너를 선택합니다. 이 작업을 수행하면 관리형 컨테이너가 있는 호스트 플러그인 페이지가 열립니다. 이

페이지는 게임 서버 빌드로 컨테이너 이미지를 생성하고, 컨테이너 플릿에 배포하며, 게임을 시작하는 단계별 프로세스를 제공합니다.

0단계: 프로필 설정

이 섹션에는 현재 선택한 사용자 프로필이 표시됩니다. 현재 사용자 프로필이 이 워크플로에 사용할 프로필인지 확인합니다. 이 워크플로에서 생성하는 모든 리소스는 프로필의 AWS 계정과 연결되며 프로필의 기본 AWS 리전에 배치됩니다. 프로필 사용자의 권한에 따라 AWS 리소스 및 작업에 대한 액세스 권한이 결정됩니다.

다음과 같은 경우 선택한 사용자 프로필을 수정해야 할 수 있습니다.

- 현재 선택된 프로필이 없습니다.
- 다른 프로필을 선택하거나 새 프로필을 생성하려고 합니다.
- 선택한 프로필을 부트스트랩해야 합니다(부트스트랩 상태가 비활성인 경우).

선택한 사용자 프로필을 설정하거나 변경하려면

- Amazon GameLift Servers 메뉴에서 AWS 액세스 자격 증명 열기를 선택합니다.

1단계: 컨테이너 준비 상태 평가

게임 서버를 컨테이너 플릿에 배포하기 전에 컨테이너 이미지에 패키징하고 Amazon ECR 리포지토리에 저장해야 합니다. 플러그인은 이러한 작업을 사용자를 대신하여 완료할 수 있으며 이러한 작업을 수동으로 수행할 수 있습니다. 이 단계에서는 컨테이너 이미지 및 ECR 리포지토리의 상태에 대한 정보를 제공합니다.

평가 질문을 사용하여 플러그인에 수행해야 할 단계를 알려줍니다.

- 새 컨테이너 이미지를 생성합니다. 이 옵션을 선택하면 다음 단계에서 게임 서버 빌드 디렉터리의 위치와 빌드 실행 파일을 묻는 메시지가 표시됩니다. 플러그인은 Dockerfile 템플릿(Amazon GameLift Servers에서 제공)을 사용하고 게임에 맞게 자동으로 구성합니다. [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)에서 템플릿을 확인할 수 있습니다. 이 옵션을 선택한 후 플러그인이 새 이미지를 저장할 위치를 지정합니다.
- 새 Amazon ECR 리포지토리를 생성하고 여기에 컨테이너 이미지를 푸시합니다. 플러그인은 선택한 사용자 프로필 AWS 리전에서 AWS 계정 및 기본값을 사용하여 프라이빗 ECR 리포지토리를 생성합니다.

- 이전에 생성된 Amazon ECR 리포지토리에 컨테이너 이미지를 푸시합니다. 이 옵션을 선택하면 다음 단계에서 목록에서 기존 Amazon ECR 리포지토리를 선택하라는 메시지가 표시됩니다. 목록에는 선택한 사용자 프로필의 AWS 계정 및 기본값 AWS 리전 에 대한 모든 Amazon ECR 리포지토리가 포함됩니다. 퍼블릭 또는 프라이빗 리포지토리를 선택할 수 있습니다.
- 기존 컨테이너 이미지를 사용합니다. 이미지를 수동으로 빌드한 경우 Amazon GameLift Servers에서 제공하는 Dockerfile 템플릿을 사용하는 것이 좋으며, 이 템플릿은 [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)에서 사용할 수 있습니다. 이 옵션을 선택한 후 이미지의 위치를 지정합니다.
- 로컬에 저장된 Docker 생성 이미지입니다. 이 옵션을 선택하면 플러그인이 새 Amazon ECR 프라이빗 리포지토리를 생성하고 로컬 이미지 파일을 푸시합니다. 다음 단계에서는 플러그인이 이미지 파일을 찾는 데 사용하는 이미지 ID를 묻는 메시지가 표시됩니다.
- Amazon ECR 리포지토리에 이미 저장된 컨테이너 이미지입니다. 이 옵션을 선택하면 다음 단계에서 목록의 기존 Amazon ECR 리포지토리 및 이미지를 선택하라는 메시지가 표시됩니다. 목록에는 선택한 사용자 프로필의 AWS 계정 및 기본값 AWS 리전 에 대한 모든 Amazon ECR 리포지토리가 포함됩니다. 퍼블릭 또는 프라이빗 리포지토리를 선택할 수 있습니다.
- Linux의 Unity 6.3에는 glibc 2.35가 필요합니다. Amazon Linux 2023에는 glibc 2.34가 포함되어 있습니다. 소스에서 glibc 2.35를 빌드하고 glibc 2.35를 사용하도록 Unity 바이너리를 구성하는 작업을 처리하는 Dockerfile 템플릿이 아래에 제공됩니다.

Unity 6.3 게임 서버 컨테이너 이미지용 Dockerfile 템플릿

이 템플릿은 Amazon GameLift Servers 플릿에서 게임 서버 컨테이너를 사용하는 데 필요한 최소 지침을 포함합니다. 게임 서버의 필요에 따라 콘텐츠를 수정합니다.

```
# 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.
FROM public.ecr.aws/amazonlinux/amazonlinux

# Game build directory
# -----
# Add your game build directory in the 'GAME_BUILD_DIRECTORY' env variable below.
#
# Game executable
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
```

```
# The game build provided over here needs to be integrated with gamelift server sdk.
# This template assumes that the executable path is relative to the game build
directory.
#
# Launch parameters
# -----
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
#
# Default directory
# -----
# The value provided in 'HOME_DIR' below will be where the game executable and logs
exist.
#
ARG GAME_BUILD_DIRECTORY
ARG GAME_EXECUTABLE
ARG LAUNCH_PARAMS

ENV GAME_BUILD_DIRECTORY=$GAME_BUILD_DIRECTORY \
    GAME_EXECUTABLE=$GAME_EXECUTABLE \
    LAUNCH_PARAMS=$LAUNCH_PARAMS \
    HOME_DIR="/local/game"

# install dependencies as necessary
RUN yum install -y shadow-utils bison wget gcc make patchelf tar gzip && \
    yum clean all && rm -fr /var/cache

RUN mkdir -p $HOME_DIR
COPY .$GAME_BUILD_DIRECTORY/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR

# Build glibc 2.35 and patch Unity binaries
RUN GLIBC_VERSION="2.35" && \
    INSTALL_PREFIX="/opt/glibc-${GLIBC_VERSION}" && \
    ARCH=$(uname -m) && \
    if [ "$ARCH" = "x86_64" ]; then \
        INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-x86-64.so.2"; \
        MONO_ARCH="x86_64"; \
    elif [ "$ARCH" = "aarch64" ]; then \
        INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-aarch64.so.1"; \
        MONO_ARCH="aarch64"; \
    else \
```

```

    echo "ERROR: Unsupported architecture: $ARCH"; exit 1; \
fi && \
cd /tmp && \
wget -q "https://ftp.gnu.org/gnu/glibc/glibc-`${GLIBC_VERSION}`.tar.gz" && \
tar -xzf "glibc-`${GLIBC_VERSION}`.tar.gz" && \
cd "glibc-`${GLIBC_VERSION}`" && mkdir glibc-build && cd glibc-build && \
touch /etc/ld.so.conf && \
../configure --prefix="`${INSTALL_PREFIX}`" && \
make -s all && make -s install && \
cd / && rm -rf /tmp/glibc-* && \
GLIBC_LIB="`${INSTALL_PREFIX}`/lib" && \
EXECUTABLE_NAME=$(echo "$GAME_EXECUTABLE" | sed 's/\.[^\.]*$//') && \
DATA_DIR="$HOME_DIR/`${EXECUTABLE_NAME}`_Data" && \
patchelf --set-interpreter "$INTERPRETER" "$HOME_DIR/`${GAME_EXECUTABLE}`" && \
patchelf --set-rpath "$GLIBC_LIB:$HOME_DIR:/lib64:`${DATA_DIR}`/MonoBleedingEdge/
$MONO_ARCH" "$HOME_DIR/`${GAME_EXECUTABLE}`" && \
patchelf --set-rpath "$GLIBC_LIB:/lib64" "$HOME_DIR/UnityPlayer.so" && \
MONO_LIB="`${DATA_DIR}`/MonoBleedingEdge/`${MONO_ARCH}`/libmonobdwgc-2.0.so" && \
if [ -f "`${MONO_LIB}`" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64" "`${MONO_LIB}`";
fi && \
GAME_ASSEMBLY="`${HOME_DIR}`/GameAssembly.so" && \
if [ -f "`${GAME_ASSEMBLY}`" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64"
"`${GAME_ASSEMBLY}`"; fi

RUN useradd -m gamescale && \
    echo "gamescale ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
    chown -R gamescale:gamescale $HOME_DIR

# Add permissions to game build
RUN chmod +x ./`${GAME_EXECUTABLE}`

USER gamescale

# Check directory before starting the container
RUN ls -lhrt .

# Check path before starting the container
RUN echo $PATH

# Start the game build
ENTRYPOINT ["/bin/sh", "-c", "`.`/`${GAME_EXECUTABLE}` $LAUNCH_PARAMS"]

```

2단계: 이미지 배포 구성

이 단계에서는 플러그인이 컨테이너 플릿에 컨테이너 이미지를 배포하는 데 필요한 정보를 제공합니다. 이 단계에서는 다음 정보를 요청합니다.

- 1단계에서 선택한 항목을 기반으로 한 게임 서버 빌드, 컨테이너 이미지 또는 Amazon ECR 리포지토리의 위치입니다.
- 관리형 컨테이너 배포에 사용할 시나리오입니다.
- 선택적 배포 설정. 이 섹션에는 플러그인이 기본적으로 사용하는 구성 설정이 있습니다. 이를 수정하거나 기본값을 유지할 수 있습니다.
 - 기본적으로 게임 이름은 게임 프로젝트 이름으로 설정됩니다. 플러그인이 생성하는 모든 AWS 리소스는 게임 이름 값을 참조합니다.
 - 포트 범위, 메모리 제한, vCPU 제한은 컨테이너 플릿의 구성 설정입니다. 이러한 값을 사용자 지정하는 방법에 대한 자세한 내용은 연결 포트 범위는 [네트워크 연결 구성](#) 섹션을 참조하고 리소스 제한은 [리소스 제한 설정](#) 섹션을 참조하세요.
- 컨테이너 이미지 태그는 Amazon ECR에서 컨테이너 이미지를 분류하는 데 사용됩니다. 기본값은 unity-gamelift-plugin입니다.

배포 시나리오 옵션

단일 리전 컨테이너 플릿

이 시나리오에서는 게임 서버를 단일 컨테이너 플릿에 배포합니다. 이는 AWS 및 컨테이너 구성과의 서버 통합을 테스트하기 위한 좋은 시작점입니다. 다음과 같은 리소스를 배포합니다.

- Amazon GameLift Servers 컨테이너 그룹 정의는 컨테이너 플릿에서 컨테이너 이미지를 배포하고 실행하는 방법을 설명합니다.
- 별칭을 사용하여 게임 서버 컨테이너가 설치되고 실행 중인 Amazon GameLift Servers 컨테이너 플릿(온디맨드).
- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
- 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
- API Gateway에 대한 과도한 플레이어 호출을 스로틀링하기 위한 웹 액세스 제어 목록(ACL).
- 게임 세션을 요청하고 게임에 참여하는 등 게임 클라이언트를 대신하여 Amazon GameLift Servers 서비스에 요청하는 백엔드 서비스:
 - 플레이어가 게임 세션 슬롯을 요청할 수 있는 API Gateway + Lambda 함수. 이 함수는 슬롯이 확보되지 않는 경우 CreateGameSession()을 호출합니다.

- 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 API Gateway + Lambda 함수.

FlexMatch가 있는 단일 리전 컨테이너 플릿

이 시나리오에서는 게임 서버를 컨테이너 플릿에 배포하고, 게임 세션 배치를 구성하며, FlexMatch 매치메이킹을 설정합니다. 이 시나리오는 호스팅 솔루션용 사용자 지정 매치메이커 설계를 시작할 준비가 되었을 때 유용합니다. 이 시나리오를 사용하여 이 솔루션의 기본 리소스를 만들고 필요에 따라 나중에 사용자 지정할 수 있습니다. 다음과 같은 리소스를 배포합니다.

- Amazon GameLift Servers 컨테이너 그룹 정의는 컨테이너 플릿에서 컨테이너 이미지를 배포하고 실행하는 방법을 설명합니다.
- 별칭을 사용하여 게임 서버 컨테이너가 설치되고 실행 중인 Amazon GameLift Servers 컨테이너 플릿(온디맨드).
- FlexMatch 매치메이킹 구성 및 매치메이킹 규칙은 플레이어 요청을 수락하고 매치를 구성하도록 설정.
- Amazon GameLift Servers 게임 세션 대기열은 (실행 가능성, 비용, 플레이어 지연 시간 등을 기반으로) 가능한 최상의 호스팅 리소스를 찾고 게임 세션을 시작하여 제안된 매치에 대한 요청을 충족시킵니다.
- 플레이어가 게임을 인증하고 시작할 수 있는 Amazon Cognito 사용자 풀 및 클라이언트.
- 사용자 풀을 API와 연결하는 API 게이트웨이 권한 부여자.
- API Gateway에 대한 과도한 플레이어 호출을 스로틀링하기 위한 웹 액세스 제어 목록(ACL).
- 게임 세션을 요청하고 게임에 참여하는 등 게임 클라이언트를 대신하여 Amazon GameLift Servers 서비스에 요청하는 백엔드 서비스:
 - 플레이어가 게임 세션 슬롯을 요청할 수 있는 API Gateway + Lambda 함수. 이 함수는 슬롯이 확보되지 않는 경우 StartMatchmaking()을 호출합니다.
 - 플레이어가 게임 요청에 대한 연결 정보를 얻을 수 있는 API Gateway + Lambda 함수.
- 플레이어용 매치메이킹 티켓과 게임 세션 정보를 저장하는 DynamoDB 테이블.
- GameSessionQueue 이벤트를 처리하는 Amazon SNS 주제 + Lambda 함수.

컨테이너 플릿 배포

플릿 구성이 완료되면 컨테이너 플릿 배포 버튼을 선택하여 배포를 시작합니다. 플러그인이 컨테이너 이미지를 생성하여 ECR에 푸시하고, 컨테이너 플릿에 대한 호스팅 리소스를 프로비저닝하며, 선택한 호스팅 솔루션 시나리오에 대해 플릿 및 기타 AWS 리소스를 배포하는 동안 이 프로세스는 몇 분 정도 소요될 수 있습니다.

배포를 시작하면 각 단계의 진행 상황을 추적할 수 있습니다. 구성에 따라 단계에 다음이 포함될 수 있습니다.

- 컨테이너 이미지 구성
- Amazon ECT 리포지토리 생성
- 이미지 빌드 및 Amazon ECR로 푸시
- 컨테이너 그룹 정의 생성
- 컨테이너 플릿 생성

자세한 배포 정보를 보려면 AWS Management Console에서 보기를 선택합니다. 컨테이너 플릿이 활성 상태에 도달하면 플릿은 게임 세션을 호스팅할 준비가 된 서버 프로세스로 컨테이너를 적극적으로 실행하고 있습니다.

배포가 완료되면 게임 세션을 호스팅하고 플레이어 연결을 수락할 준비가 된 작동 컨테이너 플릿이 있습니다.

진행 중인 배포는 중지할 수 없습니다. 배포가 잘못된 상태가 되거나 실패하는 경우 배포 재설정 옵션을 사용하여 다시 시작할 수 있습니다.

클라이언트 시작

이제 Amazon GameLift Servers로 호스팅된 멀티플레이어 게임을 시작하고 플레이하는 데 모든 작업을 완료했습니다. 게임을 플레이하려면 클라이언트 시작을 선택하여 게임 클라이언트의 로컬 인스턴스를 시작합니다.

- 단일 플릿 시나리오를 배포한 경우 한 명의 플레이어가 있는 게임 클라이언트의 하나의 인스턴스를 열고 서버 맵을 입력하여 이동할 수 있습니다. 또한 게임 클라이언트의 두 번째 인스턴스를 열어 동일한 서버 게임 맵에 두 번째 플레이어를 추가할 수 있습니다.
- FlexMatch 시나리오를 배포한 경우 호스팅 솔루션은 두 개 이상의 게임 클라이언트가 매치메이킹 요청을 할 때까지 기다립니다. 플레이어 한 명과 함께 게임 클라이언트의 인스턴스를 두 개 이상 엽니다. 두 플레이어가 매칭되고 매치를 위해 게임 세션에 참여하라는 메시지가 표시됩니다.

컨테이너 플릿 업데이트

관리형 컨테이너 호스팅 솔루션을 성공적으로 배포한 경우 배포 업데이트 기능을 사용할 수 있습니다. 이 옵션을 사용하면 새 플릿을 생성할 필요 없이 배포된 컨테이너 플릿의 구성 설정을 업데이트할 수 있습니다.

배포를 업데이트할 때 다른 게임 서버 빌드로 컨테이너 이미지를 배포하고, Amazon ECR 리포지토리를 변경하며, 다른 배포 시나리오를 선택하고, 선택적 구성 설정을 사용자 지정할 수 있습니다.

변경사항을 배포할 준비가 되면 업데이트를 선택합니다. 배포 업데이트에 필요한 시간은 전체 배포와 유사합니다. 자세한 배포 정보를 보려면 AWS 관리 콘솔에서 보기를 선택합니다.

배포된 리소스를 정리합니다.

더 이상 필요하지 않은 관리형 컨테이너 솔루션의 AWS 리소스를 정리하는 것이 가장 좋습니다. 리소스를 제거하지 않으면 이러한 리소스에 대한 비용이 계속 발생할 수 있습니다.

다음 리소스를 삭제합니다.

- 관리형 컨테이너 리소스 스택. 이 스택의 리소스는 선택한 배포 시나리오에 따라 달라집니다. 전체 스택을 삭제하려면 CloudFormation 콘솔을 사용합니다. Amazon GameLift Servers 플러그인에서 생성된 스택은 다음 명명 규칙을 사용합니다. `GameLiftPluginForUnity-{GameName}-Containers`. 플러그인에서 새 관리형 컨테이너 배포를 시작하기 전에 스택 삭제 프로세스가 완료될 때까지 기다립니다. 자세한 내용은 [CloudFormation 콘솔에서 스택 삭제](#)를 참조하세요.
- Amazon ECR 리포지토리. 플러그인을 사용하여 컨테이너 이미지에 대한 리포지토리를 생성한 경우 더 이상 필요하지 않은 저장소는 삭제하는 것이 좋습니다. 관리형 컨테이너 배포를 재설정하기 전에 리포지토리를 삭제할 필요가 없습니다. 배포를 업데이트하거나 재설정하면 플러그인은 다른 리포지토리를 사용하도록 지시되지 않는 한 동일한 리포지토리를 자동으로 사용합니다. 자세한 내용은 [Amazon ECR에서 프라이빗 리포지토리 삭제](#)를 참조하세요.

Unity용 Amazon GameLift Servers 플러그인(서버 SDK 4.x)

Note

이 주제에서는 이전 버전의 Unity용 Amazon GameLift Servers 플러그인에 대한 정보를 제공합니다. 버전 1.x는 Amazon GameLift Servers 4.x 이하용 서버 SDK를 사용합니다. 서버 SDK 5.x를 사용하고 Amazon GameLift Servers Anywhere 및 관리형 컨테이너 호스팅과 같은 최신 기능을 지원하는 최신 플러그인 버전에 대한 설명서는 [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 섹션을 참조하세요.

Amazon GameLift Servers에는 멀티플레이어와 게임 서버를 Amazon GameLift Servers에서 실행하도록 준비할 때 필요한 도구가 갖추어져 있습니다. Unity용 Amazon GameLift Servers 플러그인을 사용하면 간편하게 Amazon GameLift Servers를 Unity 게임에 통합하고 클라우드 호스팅용 Amazon

GameLift Servers 리소스를 배포할 수 있습니다. Unity용 플러그인을 사용하여 Amazon GameLift Servers API에 액세스하고 일반적인 게임 시나리오에 맞는 AWS CloudFormation 템플릿을 배포할 수 있습니다.

플러그인을 설정한 후에는 GitHub에서 [Amazon GameLift Servers Unity 샘플](#)을 사용해 볼 수 있습니다.

주제

- [Unity 게임 서버 프로젝트와 Amazon GameLift Servers 통합](#)
- [Unity 게임 클라이언트 프로젝트와 Amazon GameLift Servers 통합](#)
- [플러그인 설치 및 설정](#)
- [로컬에서 게임 테스트](#)
- [시나리오 배포](#)
- [Unity에서 Amazon GameLift Servers와 게임 통합](#)
- [샘플 게임 가져오기 및 실행](#)

Unity 게임 서버 프로젝트와 Amazon GameLift Servers 통합

Note

이 주제에서는 이전 버전의 Unity용 Amazon GameLift Servers 플러그인에 대한 정보를 제공합니다. 버전 1.x는 Amazon GameLift Servers 4.x 이하용 서버 SDK를 사용합니다. 서버 SDK 5.x를 사용하고 Amazon GameLift Servers Anywhere 및 관리형 컨테이너 호스팅과 같은 최신 기능을 지원하는 최신 플러그인 버전에 대한 설명서는 [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 섹션을 참조하세요.

이 주제는 Amazon GameLift Servers에서 호스팅하기 위한 사용자 지정 게임 서버를 준비하는 데 도움을 줍니다. 게임 서버가 해당 상태를 Amazon GameLift Servers에 알려, 메시지가 표시될 때 게임 세션을 시작 및 중지하고, 다른 작업을 수행할 수 있어야 합니다. 자세한 정보는 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#)을 참조하십시오.

사전 조건

게임 서버를 통합하기 전에 먼저 다음 작업을 완료합니다.

- [Amazon GameLift Servers의 IAM 서비스 역할 설정](#)

• 플러그인 설치 및 설정

새로운 서버 프로세스 설정

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Amazon GameLift Servers와의 통신을 설정하고 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 보고합니다.

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. OnGameSession(), OnGameSessionUpdate(), OnProcessTerminate(), OnHealthCheck()와 같이 Amazon GameLift Servers 서비스가 간접 호출하는 콜백 함수의 이름을 포함합니다. Amazon GameLift Servers가 콜백을 제공하는 데 몇 분 정도 소요될 수 있습니다.
3. Amazon GameLift Servers는 서버 프로세스의 상태를 ACTIVE으로 업데이트합니다.
4. Amazon GameLift Servers는 onHealthCheck를 주기적으로 호출합니다.

다음 코드 예에서는 Amazon GameLift Servers를 사용하여 간단한 서버 프로세스를 설정하는 방법에 대해 보여줍니다.

```
//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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

게임 초기화가 완료된 후 게임 세션을 시작할 수 있습니다.

1. `onStartGameSession` 콜백 함수를 구현합니다. Amazon GameLift Servers는 이 메서드를 호출하여 서버 프로세스에서 새 게임 세션을 시작하고 플레이어 연결을 수신합니다.

2. 게임 세션을 활성화하려면 `ActivateGameSession()`을 호출합니다. SDK에 대한 자세한 내용은 [Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업](#) 섹션을 참조하세요.

다음 코드 예에서는 Amazon GameLift Servers로 게임 세션을 시작하는 방법에 대해 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();
}
```

게임 세션 종료

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

게임 세션이 종료되면 Amazon GameLift Servers에 알립니다. 호스팅 리소스를 재활용하고 새로 고치려면 게임 세션이 완료된 후 서버 프로세스를 종료하는 것이 가장 좋습니다.

1. Amazon GameLift Servers로부터 요청을 수신하도록 `onProcessTerminate`로 이름이 지정된 함수를 설정하고 `ProcessEnding()`을 호출합니다.
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 Amazon GameLift Servers에 업로드

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

게임 서버를 Amazon GameLift Servers와 통합한 후 Amazon GameLift Servers가 게임 호스팅에 배포할 수 있도록 빌드 파일을 플릿에 업로드합니다. 서버를 Amazon GameLift Servers에 업로드하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers의 게임 서버 빌드 생성](#) 섹션을 참조하세요.

Unity 게임 클라이언트 프로젝트와 Amazon GameLift Servers 통합

Note

이 주제에서는 이전 버전의 Unity용 Amazon GameLift Servers 플러그인에 대한 정보를 제공합니다. 버전 1.x는 Amazon GameLift Servers 4.x 이하용 서버 SDK를 사용합니다. 서버 SDK 5.x를 사용하고 Amazon GameLift Servers Anywhere 및 관리형 컨테이너 호스팅과 같은 최신 기능을 지원하는 최신 플러그인 버전에 대한 설명서는 [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 섹션을 참조하세요.

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

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

백엔드 서버는 다음 게임 클라이언트 작업을 처리합니다.

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

주제

- [사전 조건](#)
- [게임 클라이언트 초기화](#)
- [특정 플릿에서 게임 세션 생성](#)
- [게임 세션에 플레이어 추가](#)
- [게임 세션에서 플레이어를 제거합니다.](#)

사전 조건

Amazon GameLift Servers 클라이언트와의 게임 서버 통신을 설정하기 전에, 먼저 다음 작업을 완료해야 합니다.

- [AWS 사용자 계정 설정](#)
- [플러그인 설치 및 설정](#)
- [Unity 게임 서버 프로젝트와 Amazon GameLift Servers 통합](#)
- [Amazon GameLift Servers용 호스팅 플릿 배포](#)

게임 클라이언트 초기화

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

게임 클라이언트를 초기화하는 코드를 추가합니다. 시작 시 이 코드를 실행하며, 이 코드는 다른 Amazon GameLift Servers 함수에 필요합니다.

1. AmazonGameLiftClient를 초기화합니다. 기본 클라이언트 구성 또는 사용자 지정 구성으로 AmazonGameLiftClient를 호출합니다. 클라이언트를 구성하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers API를 설정합니다.](#) 섹션을 참조하세요.
2. 각 플레이어가 게임 세션에 연결할 수 있도록 고유한 플레이어 ID를 생성합니다. 자세한 내용은 [플레이어 ID 생성](#)을 참조하세요.

다음 예제에서는 Amazon GameLift Servers 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 Servers service by setting up a configuration.
        //The default configuration specifies a location.
        var config = new AmazonGameLiftConfig();
        config.RegionEndpoint = Amazon.RegionEndpoint.USEast1;

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

특정 플릿에서 게임 세션 생성

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

코드를 추가하여 배포된 플릿에서 새 게임 세션을 시작하고 플레이어가 사용할 수 있게 허용합니다. Amazon GameLift Servers가 새 게임 세션을 만들고 GameSession을 반환한 후에는 플레이어를 참여시킬 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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Amazon GameLift Servers가 새 게임 세션을 만들고 `GameSession` 객체를 반환한 후에는 플레이어를 참여시킬 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 playerId, string playerSessionId)
{
    //Call ConnectPlayer with player ID and player session ID.
    return server.ConnectPlayer(playerId, playerSessionId);
}
```

게임 세션에서 플레이어를 제거합니다.

 Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

플레이어가 게임을 떠나면 게임 세션에서 플레이어를 제거할 수 있습니다.

1. 플레이어가 서버 프로세스와의 연결이 끊겼음을 Amazon GameLift Servers에 알립니다. 플레이어의 세션 ID로 RemovePlayerSession을 호출합니다.
2. RemovePlayerSession이 Success를 반환하는지 확인합니다. Amazon GameLift Servers는 Amazon GameLift Servers가 새 플레이어에 할당할 수 있도록 플레이어 슬롯을 사용 가능하도록 변경합니다.

다음 예는 플레이어 세션을 제거하는 방법을 보여줍니다.

```
public void DisconnectPlayer(int playerId)
{
    //Receive the player session ID.
    string playerIdSessionId = playerSessions[playerId];
    var outcome = GameLiftServerAPI.RemovePlayerSession(playerIdSessionId);
    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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

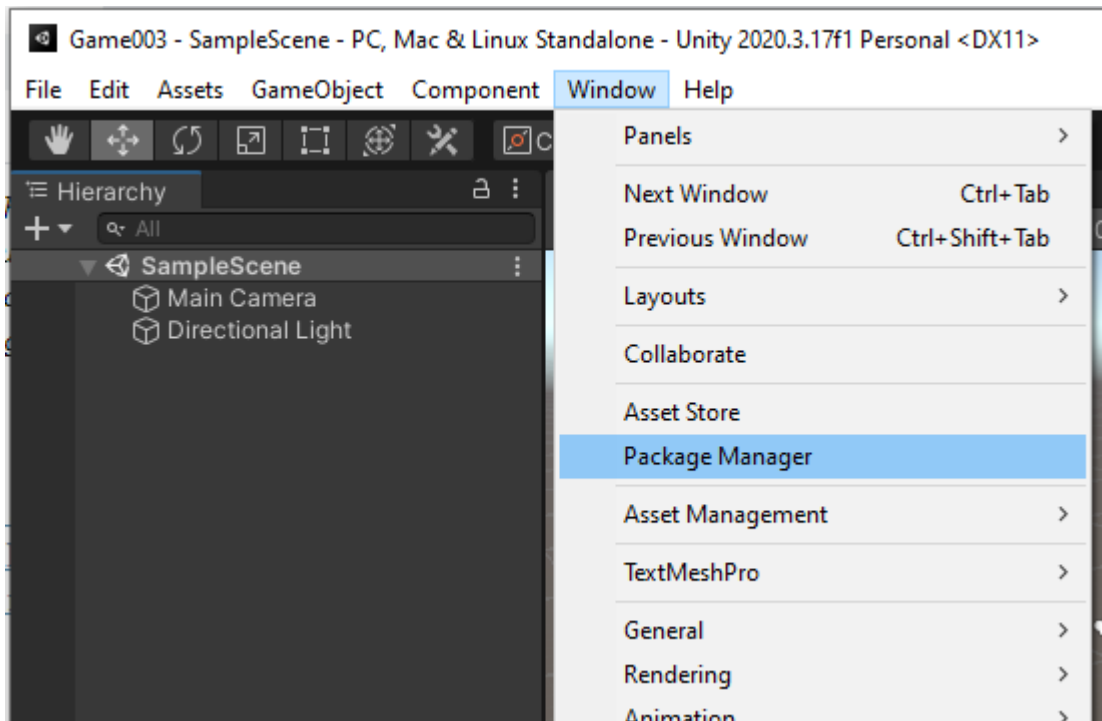
이 섹션에서는 Unity용 Amazon GameLift Servers 플러그인 버전 1.0.0을 다운로드, 설치 및 설정하는 방법을 설명합니다.

사전 조건

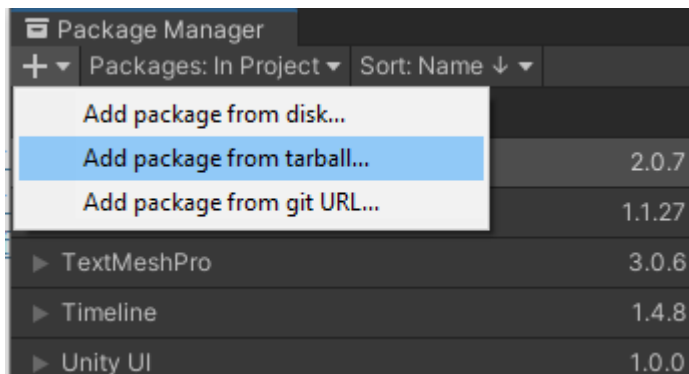
- Windows 2019.4 LTS, Windows 2020.3 LTS,용 Unity 또는 MacOS Unity
- Java의 최신 버전
- .NET 4.x의 현재 버전

Unity용 플러그인을 다운로드하고 설치하려면

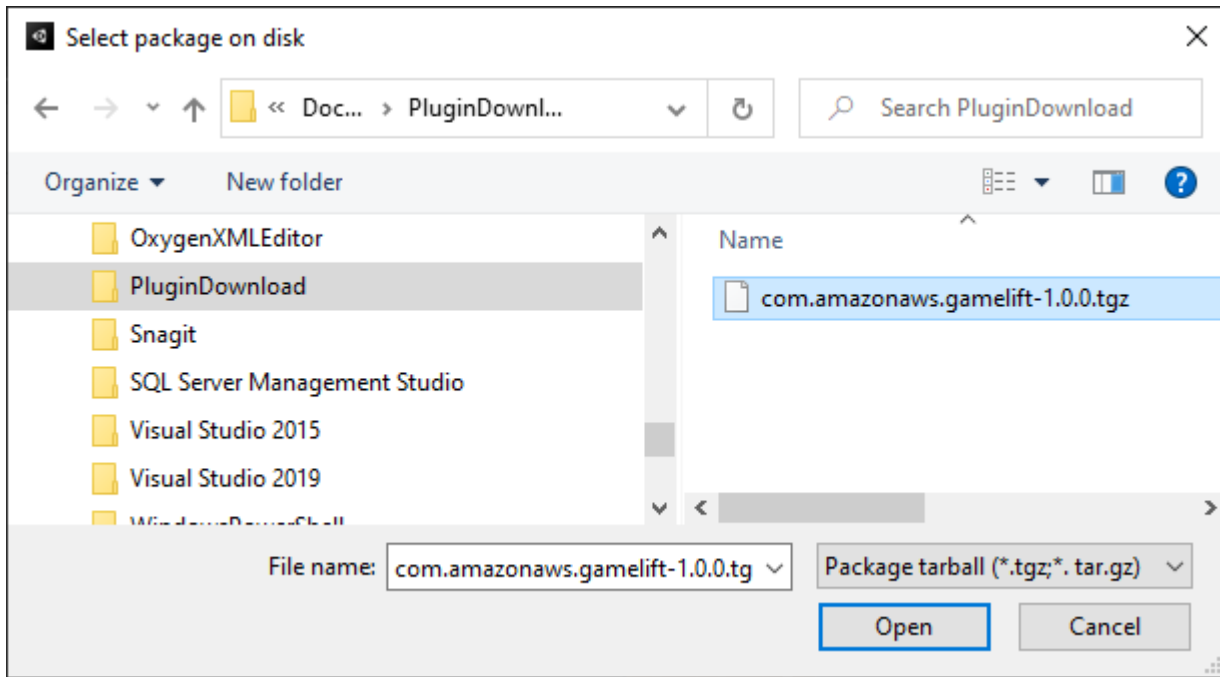
1. Unity용 Amazon GameLift Servers 플러그인을 다운로드합니다. [Unity 리포지토리용 Amazon GameLift Servers 플러그인](#) 페이지에서 최신 버전을 찾을 수 있습니다. [최신 릴리스](#)에서 자산을 선택한 다음 `com.amazonaws.gamelift-version.tgz` 파일을 다운로드합니다.
2. Unity를 시작하고 프로젝트를 선택합니다.
3. 상단 탐색 모음의 창에서 패키지 관리자를 선택합니다.



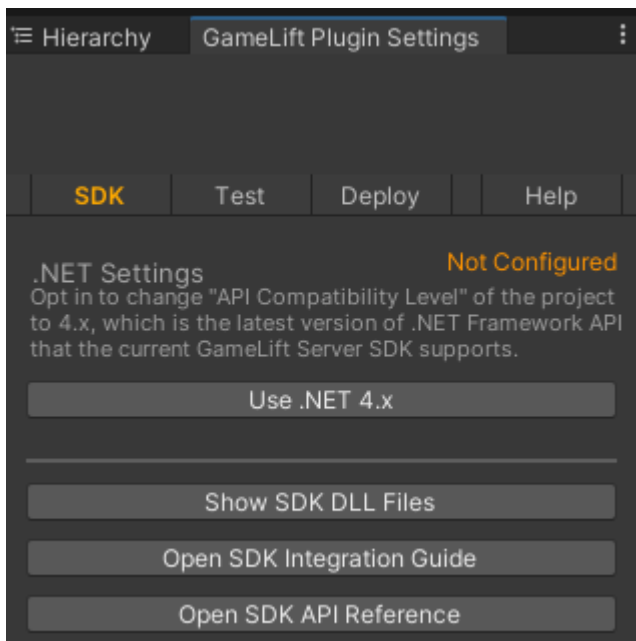
4. 패키지 관리자 탭에서 +를 선택한 다음 tarball에서 패키지 추가...를 선택합니다.



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



6. Unity가 플러그인을 로드한 후, Amazon GameLift Servers가 Unity 메뉴에 새 항목으로 표시됩니다. 스크립트를 설치하고 다시 컴파일하는 데 몇 분 정도 걸릴 수 있습니다. Amazon GameLift Servers 플러그인 설정 탭이 자동으로 열립니다.



7. SDK 창에서 .NET 4.x 사용을 선택합니다.

구성되면 상태가 구성되지 않음에서 구성됨으로 변경됩니다.

로컬에서 게임 테스트

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Amazon GameLift Servers Local을 사용하여 로컬 디바이스에서 Amazon GameLift Servers를 실행합니다. Amazon GameLift Servers Local을 사용하면 네트워크 연결 없이 몇 초 만에 코드 변경을 확인할 수 있습니다.

로컬 테스트 구성

1. Unity용 플러그인 창에서 테스트 탭을 선택합니다.
2. 테스트 창에서 Amazon GameLift Servers 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 Servers 플러그인 설정 창에서 Amazon GameLift Servers Local 경로를 선택하고 압축이 풀린 폴더로 이동하여 GameLiftLocal.jar 파일을 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 게임 실행

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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

시나리오는 CloudFormation 템플릿을 사용하여 게임에 클라우드 호스팅 솔루션을 배포하는 데 필요한 리소스를 생성합니다. 이 섹션에서는 Amazon GameLift Servers가 제공하는 시나리오와 이를 사용하는 방법에 대해 설명합니다.

사전 조건

시나리오를 배포하려면 Amazon GameLift Servers 서비스에 대한 IAM 역할이 필요합니다. Amazon GameLift Servers의 역할 생성 방법에 대한 자세한 내용은 [AWS 사용자 계정 설정](#) 섹션을 참조하세요.

각 시나리오에는 다음 리소스에 대한 권한이 필요합니다.

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

시나리오

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Unity용 Amazon GameLift Servers 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 보는 Amazon API Gateway REST 엔드포인트 지원 AWS Lambda 핸들러입니다.

단일 리전 플릿

이 시나리오에서는 단일 Amazon GameLift Servers 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 시나리오에서는 Amazon GameLift Servers 대기열과 사용자 지정 매치메이커를 사용하여 대기 중인 풀에서 가장 오래된 플레이어를 그룹화하여 매치를 구성합니다. 다음 리소스를 생성합니다.

- Amazon GameLift Servers가 메시지를 게시하는 Amazon Simple Notification Service 주제입니다. 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 시나리오에서는 Amazon GameLift Servers 대기열과 사용자 지정 매치메이커를 사용하여 매치를 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 [리소스 사용자 지정 호스팅](#) 섹션을 참조하세요.

- 서버 가용성을 높게 유지하고 비용을 낮게 유지하는 Amazon GameLift Servers 대기열입니다. 대기열에 대한 자세한 내용과 모범 사례에 대해서는 [게임 세션 대기열 사용자 지정](#) 섹션을 참조하세요.

FlexMatch

이 시나리오에서는 관리형 매치메이킹 서비스인 FlexMatch를 사용하여 게임 플레이어를 매칭합니다. FlexMatch에 대한 자세한 정보는 [Amazon GameLift Servers FlexMatch란 무엇입니까?](#)를 참조하세요. 이 시나리오에서는 다음 리소스를 생성합니다.

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

에 불필요한 요금이 부과되지 않도록 각 시나리오에서 생성한 리소스를 사용한 후 AWS 계정제거합니다. 해당 CloudFormation 스택을 삭제합니다.

AWS 자격 증명 업데이트

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Unity용 Amazon GameLift Servers 플러그인을 사용하려면 시나리오를 배포하기 위해 보안 자격 증명 이 필요합니다. 새 자격 증명을 생성하거나 기존 자격 증명을 사용할 수 있습니다.

자격 증명 구성에 대한 자세한 내용은 [AWS 자격 증명 이해 및 가져오기](#)를 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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

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

Amazon S3 버킷에 대한 자세한 내용은 [Amazon Simple Storage Service 버킷 생성, 구성, 작업](#)을 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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

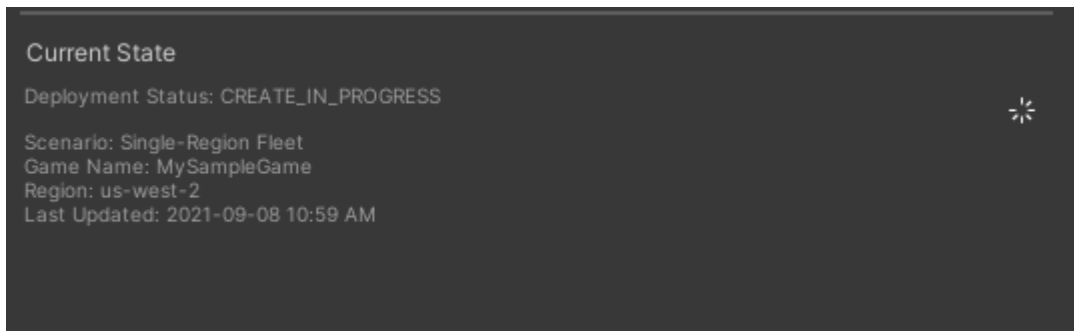
시나리오를 사용하여 Amazon GameLift Servers에서 게임을 테스트할 수 있습니다. 각 시나리오는 CloudFormation 템플릿을 사용하여 필요한 리소스로 스택을 생성합니다. 대부분의 시나리오에는 게임

서버 실행 파일과 빌드 경로가 필요합니다. 시나리오를 배포할 때 Amazon GameLift Servers는 배포의 일환으로 게임 애셋을 부트스트랩 위치에 복사합니다.

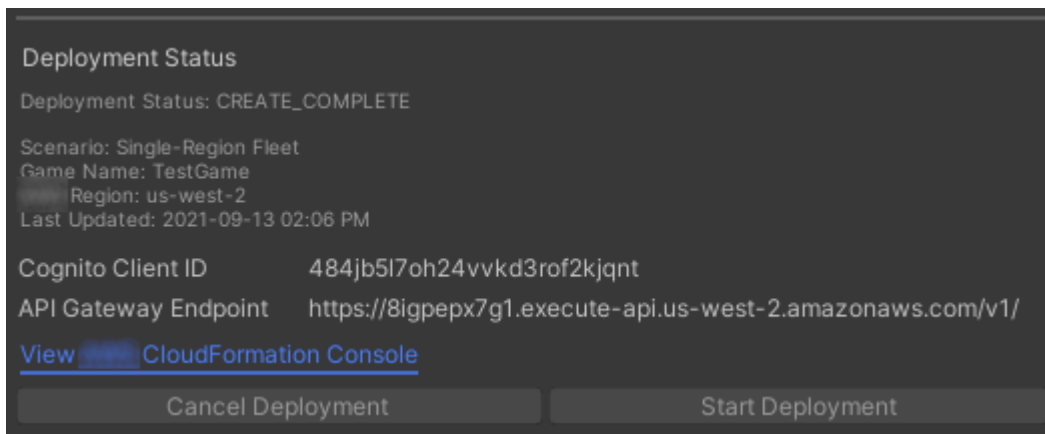
시나리오를 배포하려면 AWS 자격 증명과 AWS 계정 부트스트랩을 구성해야 합니다.

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 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 몇 분이 소요될 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 Cognito 클라이언트 ID를 입력합니다. 시나리오 배포에 AWS 리전 사용한 것과 동일한 항목을 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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

시나리오용으로 생성된 리소스를 삭제하려면 해당 CloudFormation 스택을 삭제합니다.

시나리오에서 생성된 리소스를 삭제하려면

1. Unity 배포용 Amazon GameLift Servers 플러그인 창에서 AWS CloudFormation 콘솔 보기를 선택하여 CloudFormation 콘솔을 엽니다.
2. 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. 가 시나리오에서 사용하는 스택을 CloudFormation 삭제하면 상태가 로 변경됩니다 ROLLBACK_COMPLETE.

Unity에서 Amazon GameLift Servers와 게임 통합

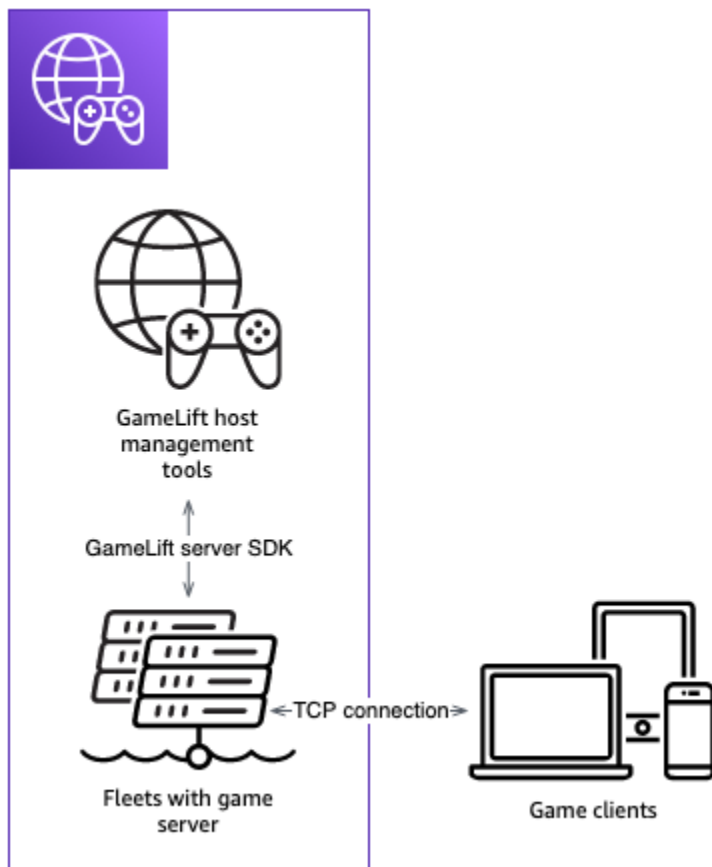
Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

다음 작업을 완료하여 Unity 게임을 Amazon GameLift Servers와 통합합니다.

- [Unity 게임 서버 프로젝트와 Amazon GameLift Servers 통합](#)
- [Unity 게임 클라이언트 프로젝트와 Amazon GameLift Servers 통합](#)

다음 다이어그램은 게임을 통합하는 흐름의 예를 보여 줍니다. 다이어그램에서는 게임 서버가 있는 플릿이 Amazon GameLift Servers에 배포되어 있습니다. 게임 클라이언트는 게임 서버와 통신하며, 게임 서버는 Amazon GameLift Servers와 통신합니다.



샘플 게임 가져오기 및 실행

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

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

사전 조건

- [AWS 사용자 계정 설정](#)
- [플러그인 설치 및 설정](#)

샘플 게임 서버 빌드 및 실행

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

샘플 게임의 게임 서버 파일을 설정합니다.

1. Unity의 메뉴에서 Amazon GameLift Servers를 선택한 다음 샘플 게임 가져오기를 선택합니다.
2. 샘플 게임 가져오기 창에서 가져오기를 선택하여 게임, 애셋, 종속 항목을 가져옵니다.
3. 게임 서버를 빌드합니다. Unity의 메뉴에서 Amazon GameLift Servers를 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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

샘플 게임의 게임 클라이언트 파일을 설정합니다.

1. Unity의 메뉴에서 Amazon GameLift Servers를 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 Servers 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 Amazon GameLift Servers와 어떻게 상호 작용하는지 확인합니다.

로컬에서 샘플 게임 테스트

Note

이 주제에서는 서버 SDK 4.x 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

Amazon GameLift Servers 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.exe입니다.
4. 배포 및 실행을 선택합니다. Unity용 플러그인은 게임 서버를 시작하고 Amazon GameLift Servers 로컬 로그 창을 엽니다. 창에는 게임 서버와 Amazon GameLift Servers Local 간에 전송된 메시지를 비롯한 로그 메시지가 포함되어 있습니다.
5. 게임 클라이언트를 실행합니다. 샘플 게임 클라이언트로 빌드 위치를 찾고 실행 파일을 선택합니다.
6. Amazon GameLift Servers 샘플 게임에서 이메일과 암호를 입력한 다음 로그인을 선택합니다. 이 이메일과 암호는 검증되거나 사용되지 않습니다.
7. Amazon GameLift Servers 샘플 게임에서 시작을 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: playerId
0 joined
2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerId
1 joined
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 0 Msg:
CONNECT: server IP localhost
```

```

2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 0:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 0
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 1 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 1:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 1

```

Amazon GameLift Servers Local 로그 샘플

```

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 이하를 사용하는 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 설명합니다.

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.

Amazon GameLift Servers와 게임 서버 통합

호스팅 리소스에 배포하고 실행할 때 사용자 지정 게임 서버가 Amazon GameLift Servers와 상호 작용할 수 있어야 합니다. 이 섹션의 주제에서는 게임 서버 프로젝트를 Amazon GameLift Servers용 서버 SDK와 통합하여 필수 및 선택적 기능을 추가하는 방법을 설명합니다. 이 섹션에는 언어별 서버 SDK 중 어느 것과도 함께 사용할 수 있는 일반적인 지침이 포함되어 있습니다 또한 Unreal Engine 및 Unity 게임 프로젝트 통합에 대한 자세한 지침도 포함되어 있습니다.

주제

- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)
- [게임 엔진용 서버 SDK 통합](#)

서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.

이 주제에서는 게임 서버 코드에 추가해야 하는 서버 SDK 기능에 대한 일반적인 지침을 제공합니다. 이 기능을 사용하면 게임 서버 프로세스가 Amazon GameLift Servers 서비스와 통신할 수 있습니다. Amazon GameLift Servers 관리형 EC2 플릿, 관리형 컨테이너 플릿 또는 Anywhere 플릿에 배포하려는 게임 서버에 이 지침을 사용합니다.

게임 서버 프로세스는 Amazon GameLift Servers 서비스와 통신하여 서비스로부터 지침을 수신하고 서버 프로세스 상태 및 게임 세션 상태를 보고합니다. 게임 호스팅 솔루션 구성 요소(게임 서버, 백엔드 서비스, 게임 클라이언트, Amazon GameLift Servers) 간의 상호 작용에 대한 자세한 내용은 [Amazon GameLift Servers와 게임 클라이언트/서버 상호 작용](#) 섹션을 참조하세요.

서버 SDK 가져오기

호스팅을 위해 게임을 준비하려면 게임 서버 프로젝트에 Amazon GameLift Servers용 서버 SDK를 추가합니다. Server SDK는 여러 언어로 제공됩니다. 서버 SDK를 포함하여 게임 서버에 대한 도구 지원에 대한 자세한 내용은 [Amazon GameLift Servers 개발 도구 가져오기](#) 섹션을 참조하세요.

github.com/amazon-gamelift 개발 언어용 서버 SDK를 다운로드합니다. 서버 SDK 버전, 플러그인, 기타 개발 도구에 대한 리포지토리를 찾을 수 있습니다.

Note

Unreal Engine 또는 Unity를 사용하여 개발하는 경우 이러한 게임 엔진용 Amazon GameLift Servers 플러그인을 다운로드합니다. 서버 SDK가 내장되어 바로 사용할 수 있습니다. 추가 정보는 다음 링크를 참조하세요.

- Unreal Engine([플러그인 다운로드](#))([통합 가이드](#))
- Unity([플러그인 다운로드](#))([통합 가이드](#))

Server SDK API 참조:

- [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers용 Go 서버 SDK - 작업](#)

서버 프로세스 초기화

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

1. `InitSdk()`를 호출하여 Amazon GameLift Servers API 클라이언트를 초기화합니다. Amazon GameLift Servers 관리형 EC2 플릿에서 실행되도록 게임 서버를 준비하는 경우 파라미터 없이 기본 `InitSDK()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 를 사용합니다. API 클라이언트는 Amazon GameLift Servers 서비스에 대한 연결을 처리합니다.

i Amazon GameLift Servers Anywhere 플릿에서 사용할 게임 서버를 준비하는 경우: 다음 `ServerParameters`로 `InitSdk()`를 호출하여 Amazon GameLift Servers API 클라이언트를 초기화합니다.

- 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Servers Anywhere 컴퓨팅이 포함된 플릿의 ID입니다.

- Amazon GameLift Servers 작업 [GetComputeAuthToken](#)을 통해 생성된 인증 토큰입니다.

2. 서비스에 게임 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 다음 정보 `ProcessParameters`와 함께 `ProcessReady()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 호출합니다. 각 게임 서버 프로세스는 `ProcessReady()`를 오직 한 번만 호출해야 합니다.

- 서버 프로세스의 포트 번호입니다. 서버 프로세스가 게임 세션을 시작하면 Amazon GameLift Servers 서비스에 포트를 제공하여 게임 세션 정보를 업데이트합니다. 게임은 이 정보를 검색하고 게임 클라이언트에 제공할 수 있으며, 이 클라이언트는 이를 사용하여 서버 프로세스에 연결하고 게임 세션에 참여합니다.
- Amazon GameLift Servers가 저장할 파일의 위치입니다. 여기에는 게임 세션 로그 및 서버 프로세스가 게임 세션 중에 생성하는 기타 파일이 포함될 수 있습니다. Amazon GameLift Servers가 이러한 파일들을 서버 프로세스가 실행 중인 컴퓨팅 장치에 일시적으로 저장하지만, 이 파일들은 인스턴스가 종료될 때까지만 사용할 수 있습니다. [Amazon GameLift Servers 콘솔](#)을 통하거나 Amazon GameLift Servers API 작업 [GetGameSessionLogUrl\(\)](#)을 호출하여 저장된 파일에 액세스할 수 있습니다.

 Amazon GameLift Servers 관리형 컨테이너 플릿에서 사용할 게임 서버를 준비하는 경우:

컨테이너 플릿에 대한 로그 파라미터를 지정할 필요가 없습니다. 대신 게임 세션 및 기타 로그 데이터를 표준 출력으로 전송합니다. 컨테이너 플릿은 모든 컨테이너 표준 출력을 로그 스트림으로 자동으로 캡처합니다.

- Amazon GameLift Servers가 게임 서버 프로세스에 메시지 또는 프롬프트를 전송하도록 허용하는 다음 콜백 함수입니다. 게임 서버 코드에 이러한 각각의 함수를 구현해야 합니다. 자세한 내용은 `ProcessParameters`([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 참조하세요.
 - (선택 사항) `onHealthCheck` - Amazon GameLift Servers는 이 함수를 정기적으로 호출하여 서버에서 상태 보고서를 요청합니다.
 - `onStartGameSession` - Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession\(\)](#)에 대한 응답으로 이 함수를 호출합니다.
 - `onProcessTerminate` - Amazon GameLift Servers는 서버 프로세스를 강제로 중지하여 정상적으로 종료합니다.

- (선택 사항) `onUpdateGameSession` - Amazon GameLift Servers가 업데이트된 게임 세션 객체를 게임 서버에 전달하거나 매치 채우기 요청에 대한 상태 업데이트를 제공합니다. [FlexMatch 채우기](#) 기능은 이 콜백이 필요합니다.

소유하거나 제어하는 다른 AWS 리소스에 안전하게 액세스하도록 게임 서버를 설정할 수도 있습니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

(선택 사항) 서버 프로세스 상태 보고

게임 서버에 코드를 추가하여 `onHealthCheck()` 콜백 함수를 구현합니다. Amazon GameLift Servers는 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 비정상적으로 보고할 수 있습니다.
- 상태 평가를 완료하고 60초 내에 콜백에 응답합니다. Amazon GameLift Servers가 60초 내에 응답을 수신하지 못하면 자동으로 서버 프로세스를 비정상적으로 간주합니다.
- 정상에 대해 `true`, 비정상에 대해 `false` 부울 값을 반환합니다.

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

Amazon GameLift Servers 서비스는 서버 프로세스 상태를 사용하여 비정상 프로세스를 종료하고 리소스를 정리합니다. 서버 프로세스가 계속 비정상적으로 보고되거나 세 번 연속 상태 확인에 응답이 없으면 이 서비스는 해당 프로세스를 종료하고 새 프로세스를 시작할 수 있습니다. 해당 서비스는 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](#)) ([Go](#)) 을 호출합니다.

게임 세션 시작

onStartGameSession 콜백 함수를 구현하는 코드를 추가합니다. Amazon GameLift Servers는 이 콜백을 호출하여 서버 프로세스에서 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 GameLift Servers 서비스에 알립니다. 서버 API 작업 ActivateGameSession() ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 호출합니다. 성공적인 호출에 대한 응답으로 해당 서비스가 게임 세션 상태를 ACTIVE으로 변경합니다.

(선택 사항) 새 플레이어 확인

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

검증을 위해 게임 세션에 참여하려는 게임 클라이언트에는 플레이어 세션 ID가 포함되어야 합니다. Amazon GameLift Servers는 게임이 [StartGameSessionPlacement\(\)](#) 또는 [StartMatchmaking\(\)](#)을 호출하여 새 게임 세션을 시작할 때 이 ID를 생성합니다. 이러한 요청에서 게임 세션의 열린 슬롯은 플레이어 세션용으로 예약됩니다.

게임 서버 프로세스가 게임 클라이언트 연결 요청을 받으면 플레이어 세션 ID로 AcceptPlayerSession() ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 호출합니다. 이에 대한 응답으로 Amazon GameLift Servers는 플레이어 세션 ID가 게임 세션에 예약된 열린 슬롯과 일치하는지 확인합니다. Amazon GameLift Servers가 플레이어 세션 ID를 검증하면 서버 프로세스가 연결을 수락합니다. 그러면 플레이어가 게임 세션에 참여할 수 있습니다. Amazon GameLift Servers가 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 GameLift Servers에 알리는 코드를 추가합니다. 이 코드는 서버 프로세스가 연결 끊김을 감지할 때마다 실행되어야 합니다.

Amazon GameLift Servers는 이 알림을 사용하여 게임 세션에서 현재 플레이어와 사용 가능한 슬롯을 추적합니다.

끊긴 연결을 코드에서 처리하려면 서버 API 작업 `RemovePlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 에 대한 호출을 해당 플레이어 세션 ID와 함께 추가합니다.

게임 세션 종료

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

서버 프로세스 종료 코드를 시작할 때 서버 API 작업 `ProcessEnding()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 호출합니다. 이 호출은 서버 프로세스가 종료되었음을 Amazon GameLift Servers에 알립니다. Amazon GameLift Servers는 게임 세션 상태와 서버 프로세스 상태를 `TERMINATED`으로 변경합니다. `ProcessEnding()`을 호출한 후에는 프로세스를 안전하게 종료할 수 있습니다.

서버 프로세스 종료 알림에 응답

Amazon GameLift Servers 서비스의 알림에 대한 응답으로 서버 프로세스를 종료하는 코드를 추가합니다. 서버 프로세스가 지속적으로 비정상 상태를 보고하거나 서버 프로세스가 실행 중인 인스턴스가 종료되는 경우 서비스가 이 알림을 보냅니다. Amazon GameLift Servers는 용량 스케일 다운 이벤트의 일부로 또는 스팟 인스턴스 중단에 대한 응답으로 인스턴스를 중지할 수 있습니다. 스팟 인스턴스 중단은 2분짜리 알림을 제공하므로 서버 프로세스에서 플레이어를 정상적으로 연결 해제하고, 게임 상태 데이터를 보존하며, 기타 정리 작업을 수행할 수 있습니다.

종료 알림을 처리하려면 게임 서버 코드를 다음과 같이 변경합니다.

- `onProcessTerminate()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 콜백 함수를 구현합니다. 이 함수는 서버 프로세스를 종료하는 코드를 호출해야 합니다.
- 게임 서버 종료 코드에서 서버 API 작업 `GetTerminationTime()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) 을 호출합니다. Amazon GameLift Servers가 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](#)) ([Go](#)) 을 호출합니다. 이 호출은 서버 프로세스가 종료 중임을 Amazon GameLift Servers 서비스에 알립니다. 그러면 서비스가 서버 프로세스 상태를 `TERMINATED`로 변경합니다. `ProcessEnding()`을 호출한 후에는 프로세스를 안전하게 종료할 수 있습니다.

게임 엔진용 서버 SDK 통합

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

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

- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#) – Amazon GameLift Servers를 게임 서버에 통합하는 방법에 대한 자세한 지침.
- [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) – 게임 세션을 생성하고 플레이어를 게임에 참가시키는 등 게임과의 통합에 대한 자세한 지침.

Unreal Engine

게임 서버

[Unreal Engine Amazon GameLift Servers용 서버 SDK](#)를 프로젝트에 추가하고 필요한 서버 기능을 구현하여 Amazon GameLift Servers에서 호스팅할 게임 서버를 준비합니다. Unreal Engine 플러그인 설정 및 Amazon GameLift Servers 코드 추가에 대한 도움말은 [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#) 단원을 참조하십시오.

게임 클라이언트와 서비스

게임 클라이언트 및/또는 게임 서비스가 사용할 수 있는 게임 세션 찾기 또는 새 게임 세션 생성 등 Amazon GameLift Servers 서비스와 상호 작용하고 게임에 플레이어를 추가할 수 있습니다. 핵심 클라이언트 기능은 [C++용AWS SDK](#)에서 제공됩니다. Amazon GameLift Servers를 Unreal Engine 게임 프로젝트에 통합하려면 [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 단원을 참조하십시오.

Unity

게임 서버

[C# Amazon GameLift Servers용 서버 SDK](#)를 프로젝트에 추가하고 필요한 서버 기능을 구현하여 Amazon GameLift Servers에서 호스팅할 게임 서버를 준비합니다. Unity를 통한 설정 및 Amazon GameLift Servers 코드 추가에 대한 도움말은 [Amazon GameLift Servers를 Unity 프로젝트에 통합](#) 단원을 참조하십시오.

게임 클라이언트와 서비스

게임 클라이언트 및/또는 게임 서비스가 사용할 수 있는 게임 세션 찾기 또는 새 게임 세션 생성 등 Amazon GameLift Servers 서비스와 상호 작용하고 게임에 플레이어를 추가할 수 있습니다. 핵심 클라이언트 기능은 [AWS SDK for .NET](#)에서 제공됩니다. Amazon GameLift Servers를 Unity 게임 프로젝트에 통합하려면 [Amazon GameLift Servers 게임 클라이언트 기능 통합](#) 단원을 참조하십시오.

기타 엔진

게임 서버 및 클라이언트에 사용할 수 있는 전체 Amazon GameLift Servers SDK 목록은 [the section called “개발 도구 가져오기”](#) 단원을 참조하십시오.

Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합

Unreal Engine용 Amazon GameLift Servers SDK를 게임 프로젝트에 통합하여 전체 서버 SDK 기능 세트에 액세스하는 방법을 알아봅니다.

도움말

더 빠른 배포를 위해 Unreal Engine용 Amazon GameLift Servers 독립 실행형 플러그인을 사용해 보세요. 최소한의 설정으로 게임 서버를 빠르게 배포할 수 있는 안내 UI 워크플로를 제공하므로 게임 구성 요소를 실제로 사용해 볼 수 있습니다. [Unreal Engine용 Amazon GameLift Servers 플러그인](#)을(를) 참조하세요.

추가 리소스:

- [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers 개발 도구 가져오기](#)

Unreal용 서버 SDK 설치

[GitHub](#)에서 Unreal Engine용 오픈 소스 Amazon GameLift Servers SDK를 가져옵니다. 리포지토리의 Readme 파일에는 사전 조건과 설치 지침이 포함되어 있습니다.

빌드 대상 및 모듈 규칙 설정

게임 프로젝트 파일을 수정하여 Amazon GameLift Servers에 사용할 빌드 구성 요소를 올바르게 생성합니다.

클라이언트 및 서버 빌드 대상을 추가하려면:

1. 게임 프로젝트의 코드 파일을 열고 `.../Games/[your application name]Source/[your application name]Target.cs` 파일을 찾습니다. 예시: `.../Source/GameLiftUnrealAppTarget.cs`. (Visual Studio를 사용하는 경우 프로젝트의 `.sln` 파일을 엽니다.)
2. 이 파일을 복사하여 `Source/` 디렉터리에 두 개의 새 대상 파일을 생성합니다.
 - 클라이언트 대상 - 새 파일의 이름을 `[your application name]Client.Target.cs`로 변경합니다. 다음 샘플 코드에 설명한 대로 콘텐츠를 편집하여 클래스 이름과 대상 유형 값을 업데이트합니다.

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

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

- 서버 대상 - 새 파일의 이름을 `[your application name]Server.Target.cs`로 변경합니다. 다음 샘플 코드에 설명한 대로 콘텐츠를 편집하여 클래스 이름과 대상 유형 값을 업데이트합니다.

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

public class GameLiftUnrealAppServerTarget : TargetRules
{
    public GameLiftUnrealAppServerTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Server;
        DefaultBuildSettings = BuildSettingsVersion.V2;
    }
}
```

```

        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp");
    }
}

```

3. 프로젝트 파일을 다시 생성합니다. Visual Studio를 사용하는 경우 게임 프로젝트의 .uproject 파일을 마우스 오른쪽 버튼으로 클릭하고 Visual Studio 프로젝트 파일 생성을 선택할 수 있습니다.

게임 프로젝트 모듈 규칙을 업데이트하려면:

플러그인에 종속되도록 게임 프로젝트의 모듈 규칙을 업데이트합니다.

1. 게임 프로젝트의 코드 파일을 열고 .../Games/[*your application name*]Source/[*your application name*].Build.cs 파일을 찾습니다. 예시: .../Source/GameLiftUnrealApp.Build.cs. (Visual Studio를 사용하는 경우 프로젝트의 .sln 파일을 엽니다.)
2. ModuleRules 클래스를 찾고 다음 샘플 코드에 표시된 대로 업데이트합니다.

```

using UnrealBuildTool;

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

```

3. 새 대상 파일을 생성하고 모듈 규칙을 수정한 후 게임 프로젝트를 다시 빌드합니다.

서버 코드에 게임 호스팅 기능 추가

서버 SDK 설치 및 설정 후 다음 단계는 게임 호스팅 기능을 서버 코드에 통합하는 것입니다. 서버 SDK를 사용하면 게임 서버가 Amazon GameLift Servers 서비스와 통신하고, 게임 세션에 대한 지침을 수신하며, 상태 및 운영 상태를 보고하고, 기타 작업을 수행할 수 있습니다.

이 주제에서는 Amazon GameLift Servers를 사용하여 게임을 호스팅하는 데 필요한 최소 기능을 추가하는 샘플 코드를 제공합니다.

1단계: **GameMode** 헤더 파일 업데이트

1. 게임 프로젝트의 코드 파일을 열고 *Your-application-name*GameMode.h 파일을 찾습니다.
예시: GameLiftUnrealAppGameMode.h. Visual Studio를 사용하는 경우 게임 프로젝트의 .sln 파일을 엽니다.
2. 다음 예제 코드를 포함하도록 헤더 파일을 변경합니다. 반드시 "GameLiftUnrealApp"을 사용자 고유의 애플리케이션 이름으로 변경해야 합니다.

GameMode.h용 샘플 코드

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

#pragma once

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

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

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

public:
    AGameLiftUnrealAppGameMode();
```

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

private:
    void InitGameLift();

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

2단계: 게임 서버 코드에 필요한 서버 SDK 호출 추가

이 섹션의 샘플 코드를 사용하여 Amazon GameLift Servers에 사용할 게임 서버 코드를 통합합니다. 코드의 기능에 대한 자세한 내용은 [서버 프로세스 초기화](#) 및 [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#) 섹션을 참조하세요.

Note

WITH_GAMELIFT 프리프로세서 플래그는 다음 두 가지 용도로 사용됩니다.

- Amazon GameLift Servers 백엔드 API 호출을 Unreal 서버 빌드로만 제한합니다.
- 다양한 Unreal 빌드 대상 간의 호환성 보장

1. 관련 소스 파일인 *Your-application-name*GameMode.cpp 파일을 엽니다. 이 예제에서는 GameLiftUnrealAppGameMode.cpp입니다.
2. 코드를 다음 예제 코드에 맞게 변경합니다. 반드시 'GameLiftUnrealApp'의 모든 인스턴스를 사용자 고유의 애플리케이션 이름으로 변경해야 합니다.

제공된 코드 샘플은 Amazon GameLift Servers와의 통합에 필요한 요소를 추가하는 방법을 보여줍니다. 다음이 포함됩니다.

- Amazon GameLift Servers API 클라이언트를 초기화합니다.
- OnStartGameSession, OnProcessTerminate, onHealthCheck를 포함한 Amazon GameLift Servers 서비스의 요청에 응답하는 콜백 함수를 구현합니다.
- 게임 세션을 호스팅할 준비가 되면 ProcessReady()를 호출하여 Amazon GameLift Servers 서비스에 알립니다.

GameMode.cpp용 샘플 코드

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

#include "GameLiftUnrealAppGameMode.h"

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

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

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

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

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

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

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

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
```

```

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

    FString glAnywhereSecretKey = "";

```

```

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

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

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

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

    //InitSDK will establish a local connection with GameLift's agent to enable further
communication.
    FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule-
>InitSDK(ServerParametersForAnywhere);
    if (InitSdkOutcome.IsSuccess())

```

```

{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("GameLift InitSDK succeeded!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: InitSDK failed : ("));
    FGameLiftError GameLiftError = InitSdkOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"), *GameLiftError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    return;
}

ProcessParameters = MakeShared<FProcessParameters>();

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

//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback
before shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to shut
down.
ProcessParameters->OnTerminate.BindLambda([=]()
{
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    // First call ProcessEnding()
    FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule-
>ProcessEnding();

```

```

        // Then call Destroy() to free the SDK from memory
        FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();
        // Exit the process with success or failure
        if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess()) {
            UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
        }
        else {
            if (!processEndingOutcome.IsSuccess()) {
                const FGameLiftError& error = processEndingOutcome.GetError();
                UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error:
%s"),
                    error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
            }
            if (!destroyOutcome.IsSuccess()) {
                const FGameLiftError& error = destroyOutcome.GetError();
                UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
                    error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
            }
        }
    });

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)

```

```

{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

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

//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

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

```

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

3단계: 게임 프로젝트 재구축

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

Note

솔루션을 다시 빌드할 필요는 없습니다. 대신 앱의 /Games/ 폴더 아래에 프로젝트를 빌드합니다. 그렇지 않으면 Visual Studio에서 전체 UE5 프로젝트를 다시 빌드하므로 최대 1 시간 정도 소요될 수 있습니다.

호스팅을 위해 게임 서버를 패키징합니다.

이제 게임 서버 코드가 최소 필수 서버 SDK 기능과 통합되어 Unreal Editor를 사용하여 게임 서버 빌드를 패키징할 준비가 되었습니다.

게임 서버 빌드를 패키징하려면

- Unreal Editor에서 게임 프로젝트를 엽니다.
- Unreal Editor 단계에 따라 게임 서버를 패키징합니다.
 - 대상 플랫폼(Windows 또는 Linux)을 선택합니다.
 - 서버 빌드 대상(*[your application name]*Server)을 선택합니다.

패키징 프로세스는 게임 서버 실행 파일 *[your application name]*Server.exe를 생성합니다.

- 호스팅 리소스에 배포할 게임 서버 빌드를 준비합니다. 빌드에는 다음 파일이 포함되어야 합니다.
 - 게임 서버 실행 파일
 - Unreal Engine 버전 5.5 이상을 사용하는 경우 Windows 빌드에 다음 파일을 포함합니다. Unreal Engine의 소스 빌드 버전에서 찾을 수 있습니다.
 - VC_redist.x64.exe (UnrealEngine\Engine\Source\Programs\PrereqInstaller\Resources\VCRedist\)

- UEPrereqSetup_x64.exe or UE5PrereqSetup_x64.exe (UnrealEngine\Engine\Extras\Redist\en-us\)
- 게임 서버에 대한 필요한 기타 모든 종속성입니다.
- 필요한 경우 OpenSSL 라이브러리. 게임 서버가 Amazon GameLift Servers 서버 SDK 버전 5.3 이상과 통합된 경우 이 단계를 건너뛸 수 있습니다. [최신 서버 SDK 버전은 여기에서 확인할 수 있습니다.](#)

Unreal용 서버 SDK, 버전 5.2 이하로 생성된 게임 서버에 필요한 OpenSSL 라이브러리

게임 서버를 Unreal로 패키징할 때 사용한 것과 동일한 버전의 OpenSSL 라이브러리를 포함해야 합니다. 이러한 라이브러리는 게임 엔진 소스에 있습니다. 위치는 개발 환경에 따라 다릅니다.

Windows의 경우:

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

Linux의 경우:

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

OpenSSL 라이브러리를 게임 서버 실행 파일과 동일한 디렉터리의 게임 빌드 패키지에 복사합니다.

다음 단계

이제 Amazon GameLift Servers를 사용하여 호스팅하는 데 필요한 최소 기능으로 게임 서버 빌드를 준비했으므로 다음 단계를 고려하세요.

- 테스트 및 개발을 위해 통합 게임 서버를 배포합니다. Anywhere 플릿을 사용하면 로컬 시스템을 호스팅 리소스로 설정하고 이를 사용하여 게임 서버 및 게임 클라이언트 연결을 테스트할 수 있습니다. 클라우드 기반 호스팅의 경우 관리형 EC2 또는 관리형 컨테이너 플릿에 게임 서버를 배포합니다. 지침에 대해서는 다음 주제를 참조하세요.
- [Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정](#)
- [Amazon GameLift Servers Anywhere 플릿](#)

- [Amazon GameLift Servers 관리형 EC2 플릿](#)
- [Amazon GameLift Servers 관리형 컨테이너 플릿](#)
- 선택적 기능을 추가하여 게임 서버 통합을 사용자 지정합니다. 예를 들어 고유한 플레이어 IDs로 플레이어 세션을 추가하거나, 매치메이킹 채우기를 설정하거나, 다른 AWS 리소스(예: 데이터베이스 또는 콘텐츠 스토리지 서비스)에 대한 게임 서버 액세스를 관리할 수 있습니다. 지침에 대해서는 다음 주제를 참조하세요.
- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)
- [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
- 게임 클라이언트 구성 요소를 사용자 지정하여 게임 세션을 요청하고, 연결 정보를 수신하며, 게임 서버에 직접 연결하여 게임을 플레이합니다. 지침에 대해서는 다음 주제를 참조하세요.
- [Amazon GameLift Servers 게임 클라이언트 기능 통합](#)

Amazon GameLift Servers를 Unity 프로젝트에 통합

Unity용 Amazon GameLift Servers SDK를 게임 프로젝트에 통합하여 전체 서버 SDK 기능 세트에 액세스하는 방법을 알아봅니다.

도움말

더 빠른 배포를 위해 Unity용 Amazon GameLift Servers 독립 실행형 플러그인을 사용해 보세요. 최소한의 설정으로 게임 서버를 빠르게 배포할 수 있는 안내 UI 워크플로를 제공하므로 게임 구성 요소를 실제로 사용해 볼 수 있습니다. [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#)을(를) 참조하세요.

추가 리소스:

- [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)
- [the section called “개발 도구 가져오기”](#)

Unity용 서버 SDK 설치

[GitHub](#)에서 Unity용 오픈 소스 Amazon GameLift Servers를 가져옵니다. 리포지토리의 Readme 파일에는 사전 조건과 설치 지침이 포함되어 있습니다.

테스트용 Amazon GameLift Servers Anywhere 플릿 설정

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

워크스테이션 설정에 대한 지침은 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#) 섹션을 참조하여 다음 단계를 완료합니다.

1. 워크스테이션에 대한 사용자 지정 위치를 만듭니다.
2. 새 사용자 지정 위치를 사용하여 Amazon GameLift Servers Anywhere 플릿을 생성합니다. 요청이 성공하면 플릿 ID가 반환됩니다. 나중에 필요하므로 이 값을 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 값을 기록해 둡니다.
4. 워크스테이션 컴퓨팅을 위한 인증 토큰을 생성합니다. 이 단수명 인증에는 토큰과 만료 날짜가 포함됩니다. 게임 서버는 이를 사용하여 Amazon GameLift Servers 서비스와의 통신을 인증합니다. 워크스테이션 컴퓨팅에 인증을 저장하여 실행 중인 게임 서버 프로세스가 액세스할 수 있도록 합니다.

Amazon GameLift Servers 서버 코드를 Unity 프로젝트에 추가

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

제공된 코드 예제는 기본적인 필수 통합 요소를 보여줍니다. MonoBehaviour를 사용하여 Amazon GameLift Servers로 간단한 게임 서버 초기화를 설명합니다. 이 예제에서는 테스트를 위해 게임 서버가 Amazon GameLift Servers Anywhere 플릿에서 실행된다고 가정합니다. 여기에는 다음과 같은 코드가 포함됩니다.

- Amazon GameLift Servers API 클라이언트를 초기화합니다. 이 샘플은 Anywhere 플릿 및 컴퓨팅용 서버 파라미터가 있는 `InitSDK()` 버전을 사용합니다. 이전 주제 [테스트용 Amazon GameLift Servers Anywhere 플릿 설정](#)에서 정의한 대로 WebSocket URL, 플릿 ID, 컴퓨팅 이름(호스트 ID) 및 인증 토큰을 사용합니다.
- `OnStartGameSession`, `OnProcessTerminate`, `onHealthCheck`를 포함한 Amazon GameLift Servers 서비스의 요청에 응답하는 콜백 함수를 구현합니다.

- 지정된 포트와 함께 ProcessReady()를 호출하여 프로세스가 게임 세션을 호스팅할 준비가 되면 Amazon GameLift Servers 서비스에 알립니다.

제공된 샘플 코드는 Amazon GameLift Servers 서비스와의 통신을 설정합니다. 또한 Amazon GameLift Servers 서비스의 요청에 응답하는 콜백 함수 집합을 구현합니다. 각 함수와 코드가 수행하는 작업에 대한 자세한 내용은 [서버 프로세스 초기화](#)를 참조하세요. 이 코드에 사용된 SDK 작업 및 데이터 유형에 대한 자세한 내용은 [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#) 섹션을 참조하세요.

샘플 코드는 [게임 서버에 Amazon GameLift Servers 추가](#)에 설명된 대로 필요한 기능을 추가하는 방법을 보여줍니다. 서버 SDK 작업에 대한 자세한 내용은 [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#) 섹션을 참조하세요.

통합 샘플 코드

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

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

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

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

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

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

        //Authorization token for this host process
```

```

var authToken = "myAuthToken";

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

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

```

```

        //It gives the game server a chance to save its state, communicate
with services, etc.,
        //and initiate shut down. When the game server is ready to shut
down, it invokes the
        //server SDK call ProcessEnding() to tell Amazon GameLift Servers
it is shutting down.
        GameLiftServerAPI.ProcessEnding();
    },
    () => {
        //Implement callback function OnHealthCheck
        //Amazon GameLift Servers invokes this callback approximately every
60 seconds.

        //A game server might want to check the health of dependencies,
etc.

        //Then it returns health status true if healthy, false otherwise.
        //The game server must respond within 60 seconds, or Amazon
GameLift Servers records 'false'.
        //In this example, the game server always reports healthy.
        return true;
    },
    //The game server gets ready to report that it is ready to host game
sessions
    //and that it will listen on port 7777 for incoming player connections.
    listeningPort,
    new LogParameters(new List<string>()
    {
        //Here, the game server tells Amazon GameLift Servers where to find
game session log files.
        //At the end of a game session, Amazon GameLift Servers uploads
everything in the specified
        //location and stores it in the cloud for access later.
        "/local/game/logs/myserver.log"
    }));

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

```

```

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

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

```

다음 단계

이제 Amazon GameLift Servers를 사용하여 호스팅하는 데 필요한 최소 기능으로 게임 서버 빌드를 준비했으므로 다음 단계를 고려하세요.

- 테스트 및 개발을 위해 통합 게임 서버를 배포합니다. Anywhere 플릿을 사용하면 로컬 시스템을 호스팅 리소스로 설정하고 이를 사용하여 게임 서버 및 게임 클라이언트 연결을 테스트할 수 있습니다. 클라우드 기반 호스팅의 경우 관리형 EC2 또는 관리형 컨테이너 플릿에 게임 서버를 배포합니다. 지침에 대해서는 다음 주제를 참조하세요.
- [Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정](#)
- [Amazon GameLift Servers Anywhere 플릿](#)

- [Amazon GameLift Servers 관리형 EC2 플릿](#)
- [Amazon GameLift Servers 관리형 컨테이너 플릿](#)
- 선택적 기능을 추가하여 게임 서버 통합을 사용자 지정합니다. 예를 들어 고유한 플레이어 IDs로 플레이어 세션을 추가하거나, 매치메이킹 채우기를 설정하거나, 다른 AWS 리소스(예: 데이터베이스 또는 콘텐츠 스토리지 서비스)에 대한 게임 서버 액세스를 관리할 수 있습니다. 지침에 대해서는 다음 주제를 참조하세요.
- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)
- [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
- 게임 클라이언트 구성 요소를 사용자 지정하여 게임 세션을 요청하고, 연결 정보를 수신하며, 게임 서버에 직접 연결하여 게임을 플레이합니다. 지침에 대해서는 다음 주제를 참조하세요.
- [Amazon GameLift Servers 게임 클라이언트 기능 통합](#)

배포를 위해 게임 서버 빌드 패키징

게임 서버 소프트웨어를 호스팅 리소스에 배포할 준비를 합니다. 배포되면 소프트웨어가 각 호스팅 리소스에 설치된 다음 하나 이상의 게임 서버 프로세스가 시작되고 준비되어 플레이어를 위한 게임 세션을 호스팅합니다.

게임 서버 빌드를 준비하는 방법은 사용 중인 Amazon GameLift Servers 호스팅 옵션 유형에 따라 다릅니다. 모든 게임 서버 빌드는 [Amazon GameLift Servers와 게임 서버 통합](#)에 설명된 대로 Amazon GameLift Servers용 서버 SDK와 통합되어야 합니다.

이 주제에서는 다음 시나리오에 소프트웨어를 배포할 수 있도록 준비하는 방법에 대한 지침을 제공합니다.

- 관리형 EC2 호스팅의 경우 서버 소프트웨어를 패키징하고 배포를 위해 Amazon GameLift Servers에 업로드합니다.
- 관리형 컨테이너 호스팅의 경우 서버 소프트웨어로 컨테이너 이미지를 빌드하고 배포를 위해 Amazon Elastic 컨테이너 레지스트리에 저장합니다.
- Amazon GameLift Servers Anywhere로 호스팅할 경우 자체 호스팅 리소스에 설치하기 위해 필요에 따라 서버 소프트웨어를 패키징합니다.

Note

Amazon GameLift Servers Realtime 구성된 스크립트를 배포하는 경우 [Amazon GameLift Servers Realtime용 스크립트 업로드](#)를 참조하세요.

주제

- [Amazon GameLift Servers의 게임 서버 빌드 생성](#)
- [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)

Amazon GameLift Servers의 게임 서버 빌드 생성

게임 서버를 Amazon GameLift Servers와 통합한 후([Amazon GameLift Servers를 사용하여 호스팅할 게임 준비](#) 참조) 호스팅을 위해 게임 서버 소프트웨어를 컴퓨팅 리소스에 설치합니다. 이 프로세스는 사용 중인 Amazon GameLift Servers 호스팅 유형에 따라 달라집니다.

관리형 호스팅을 위한 배포

Amazon GameLift Servers 관리형 EC2 호스팅을 사용하는 경우 게임 서버 소프트웨어를 패키징하여 Amazon GameLift Servers에 업로드해야 합니다. 관리형 플릿을 생성하면 Amazon GameLift Servers가 자동으로 각 플릿 인스턴스에 배포합니다.

이 섹션의 주제에서는 업로드를 위해 빌드 파일을 패키징하고, 선택적 빌드 설치 스크립트를 생성한 다음, [AWS Command Line Interface \(AWS CLI\)](#) 또는 AWS SDK를 사용하여 파일을 업로드하는 방법을 설명합니다.

Anywhere 호스팅 배포

자체 관리형 호스팅을 위해 Amazon GameLift Servers Anywhere 플릿을 사용하는 경우 플릿의 각 컴퓨팅에 게임 서버 소프트웨어를 설치하고 업데이트를 유지하는 것은 사용자의 책임입니다.

통합 게임 서버 프로세스가 실행되기 시작하면 자동으로 초기화되고 Amazon GameLift Servers 서비스와 통신이 설정됩니다. 서버 프로세스가 Amazon GameLift Servers의 프롬프트에서 게임 세션을 시작하고 활동을 서비스에 다시 보고합니다.

주제

- [게임 빌드 파일 패키징하기](#)
- [빌드 설치 스크립트 추가](#)

• [관리형 호스팅을 위한 Amazon GameLift Servers 빌드 리소스 생성](#)

게임 빌드 파일 패키징하기

구성된 게임 서버를 Amazon GameLift Servers에 업로드하기 전에 게임 빌드 파일을 빌드 디렉터리에 패키징합니다. 이 프로세스는 EC2 관리형 플릿으로 호스팅할 때의 요구 사항이며 Anywhere 플릿으로 호스팅할 때의 모범 사례입니다. 빌드 디렉터리에는 게임 서버를 실행하고 게임 세션을 호스팅하는 데 필요한 모든 구성 요소가 포함되어야 합니다. 이러한 단계에는 다음 작업이 포함되어 있을 수 있습니다.

- 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 [Amazon GameLift Servers 개발 도구 가져오기](#) 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 Servers의 서버 SDK로 만든 게임 빌드(Unreal 플러그인으로 만든 빌드 포함)의 경우, 서버 SDK를 빌드할 때 사용한 것과 동일한 버전의 OpenSSL용 OpenSSL DLL을 포함합니다. 자세한 내용은 Server SDK README 파일을 참조하세요.

- 설치 스크립트(선택 사항) - Amazon GameLift Servers 호스팅 서버에 게임 빌드를 설치하는 작업을 처리하는 스크립트 파일입니다. 이 파일을 빌드 디렉터리의 루트에 배치합니다. Amazon GameLift Servers는 플릿 생성의 일부로 설치 스크립트를 실행합니다.

설치 스크립트를 포함하여 빌드의 모든 애플리케이션을 설정하여 다른 AWS 서비스의 리소스에 안전하게 액세스할 수 있습니다. 이를 수행하는 자세한 방법은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 섹션을 참조하세요.

빌드 파일을 패키징한 후에는 모든 필요한 종속성을 포함하고 설치 스크립트가 정확한지 확인하기 위해 게임 서버가 새로 설치된 대상 OS에서 실행될 수 있도록 합니다.

빌드 설치 스크립트 추가

게임 빌드의 운영 체제(OS)에 맞게 다음과 같이 설치 스크립트를 생성합니다.

- Windows: install.bat이라는 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 GameLift Servers는 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 `sudo`를 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 GameLift Servers는 `bash` 같은 공통 shell 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 [플릿 인스턴스에 연결](#) 섹션을 참조하세요.
- 설치 스크립트는 VPC 피어링 연결을 사용할 수 없습니다. VPC 피어링 연결은 Amazon GameLift Servers가 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 `bash`를 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 GameLift Servers는 서비스 VPC에서 실행되므로 사용자를 대신하여 AWS Identity and Access Management (IAM) 역할을 맡을 수 있는 권한을 Amazon GameLift Servers에 부여해야 합니다. 가 역할을 수임Amazon GameLift Servers하도록 허용하려면 AWS 관리형 정책이 포함된 역할을 생성하고 `CloudWatchAgentAdminPolicy` 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
```

관리형 호스팅을 위한 Amazon GameLift Servers 빌드 리소스 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 GameLift Servers가 고유의 빌드 ID 및 기타 메타데이터를 사용하여 새 빌드 리소스를 생성합니다. 빌드는 초기화된 상태에서 시작됩니다. Amazon GameLift Servers가 게임 서버 파일을 획득한 후 빌드는 준비 완료 상태로 이동합니다.

빌드가 준비되면 새 Amazon GameLift Servers 플릿에 배포할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 관리형 EC2 플릿 생성](#) 섹션을 참조하세요. Amazon GameLift Servers는 새 플릿을 설정할 때 빌드 파일을 각 플릿 인스턴스에 다운로드하고 빌드 파일을 설치합니다.

파일 디렉터리에서 빌드 생성

로컬 디렉터리를 포함하여 모든 위치에 저장된 게임 빌드를 생성하려면 [upload-build](#) AWS CLI 명령을 사용합니다. 이 명령은 Amazon GameLift Servers에 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 server SDK for Amazon GameLift Servers 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 Servers 서버 SDK 버전입니다. 값을 입력하지 않으면 Amazon GameLift Servers에서 기본값 4.0.2를 사용합니다. 서버 SDK 버전을 잘못 지정하면 Amazon GameLift Servers 서비스에 대한 연결을 설정하도록 InitSdk를 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 region](#)을 사용하여 현재 기본 리전을 확인합니다. 기본 리전을 변경하려면 [aws configure set region *region name*](#) 명령을 사용합니다.

예시

```
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_2022 \  
  --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 GameLift Servers는 업로드 진행 상황을 제공합니다. 업로드에 성공하면 Amazon GameLift Servers는 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 GameLift Servers에 업로드할 수 있습니다. 빌드를 생성할 때 S3 버킷 위치를 지정하고 Amazon S3에서 직접 빌드 파일을 Amazon GameLift Servers 검색합니다.

빌드 리소스를 생성하려면

1. 빌드 파일을 Amazon S3에 저장합니다. 패키지 빌드 파일이 포함된 .zip 파일을 생성하여의 S3 버킷에 업로드합니다 AWS 계정. Amazon GameLift Servers 빌드를 생성할 때 버킷 레이블과 파일 이름을 기록해 둡니다.
2. 빌드 파일에 대한 액세스 권한을 Amazon GameLift Servers에 부여합니다. [Amazon S3의 게임 빌드 파일에 액세스](#)의 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 Servers 콘솔 또는를 사용하여 새 빌드 레코드를 AWS CLI 생성합니다. [Amazon GameLift Servers의 IAM 권한 예제](#)에 설명된 대로 PassRole 권한이 있어야 합니다.

Console

1. [Amazon GameLift Servers 콘솔](#)의 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 GameLift Servers에서 S3 버킷 및 빌드 객체에 액세스할 수 있도록 생성한 역할을 선택합니다.
4. (선택 사항) 태그에서 키 및 값 쌍을 입력하여 빌드에 태그를 추가합니다.
5. 생성(Create)을 선택합니다.

Amazon GameLift Servers는 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 server SDK for Amazon GameLift Servers version \
  --operating-system supported OS \
  --build-version user-defined build number \
  --storage-location "Bucket"=S3 bucket label,"Key"=Build .zip file name,"RoleArn"=Access role ARN} \
  --region region name
```

- name - 새 빌드를 설명하는 이름입니다.
- server-sdk-version - 게임 서버를 Amazon GameLift Servers와 통합하는 데 사용되는 Amazon GameLift Servers의 서버 SDK 버전입니다. 값을 입력하지 않으면 Amazon GameLift Servers에서 기본값 4.0.2를 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". 정책 예제는 [Amazon S3의 게임 빌드 파일에 액세스](#)를 참조하세요.

- **region** - 플릿을 배포하려는 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.

예제

```
aws gamelift create-build \
  --operating-system WINDOWS_2022 \
  --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.

Amazon GameLift Servers용 컨테이너 이미지 빌드

이 주제에서는 Amazon GameLift Servers에서 사용할 게임 서버 소프트웨어로 컨테이너 이미지를 생성하는 방법을 설명합니다. 게임 서버 컨테이너 이미지에는 게임 서버 실행 파일과 실행해야 하는 모든 종속성이 포함됩니다. 게임 서버 컨테이너 이미지는 Amazon GameLift Servers 관리형 컨테이너 호스팅 솔루션과 함께 사용됩니다. 전체 솔루션 구축에 대한 자세한 내용은 다음을 참조하세요.

- [Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers의 컨테이너 작동 방식](#)

다음 작업을 완료하여 게임 서버 컨테이너 이미지를 Amazon GameLift Servers 컨테이너 플릿에 배포할 수 있도록 준비합니다. 이러한 작업을 시작하기 전에 게임 서버 코드와 Amazon GameLift Servers 서버 SDK의 통합을 완료합니다.

주제

- [게임 서버 컨테이너 이미지 생성](#)

- [Amazon ECR에 컨테이너 이미지 푸시](#)

게임 서버 컨테이너 이미지 생성

Linux 기반 플랫폼에서 작업하거나 Docker가 설치된 Windows Subsystem for Linux(WSL)를 사용하는 경우 다음 지침을 따릅니다.

게임 서버 컨테이너 이미지를 생성하려면

1. 게임 서버 소프트웨어를 사용하여 작업 디렉터리를 준비합니다. 로컬 시스템에서 작업 디렉터리를 생성하여 게임 서버 컨테이너의 파일을 구성합니다. 컨테이너 이미지는 호스팅을 위해 컨테이너를 Amazon GameLift Servers 리소스에 배포할 때 이 파일 구조를 사용합니다. 예제:

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .
.
./glc
./glc/gamebuild
```

Note

이 기능을 사용해 보려고 하지만 작동하는 게임 서버 빌드가 없는 경우 GitHub에서 사용할 수 있는 샘플 게임 서버인 [SimpleServer](#)를 사용해 보세요.

2. 제공된 템플릿을 사용하여 새 Dockerfile을 생성합니다.
3. Dockerfile 템플릿의 지침에 따라 직접 사용할 수 있도록 업데이트합니다.
 - 필요에 따라 기본 이미지를 업데이트합니다.
 - 게임 서버 빌드의 환경 변수를 설정합니다.
4. 컨테이너 이미지를 빌드합니다. 자체 리포지토리 이름을 지정하여 `docker build`를 실행합니다. 예제:

```
[~/work/glc]$ docker build -t <local repository name>:<optional tag> .
```

다음 예제와 같이 `docker images` 명령을 사용하여 리포지토리 및 이미지 ID를 볼 수 있습니다.

게임 서버 컨테이너 이미지용 Dockerfile 템플릿

이 템플릿은 Amazon GameLift Servers 플릿에서 게임 서버 컨테이너를 사용하는 데 필요한 최소 지침을 포함합니다. 게임 서버의 필요에 따라 콘텐츠를 수정합니다.

```
# Base image
# -----
# Add the base image that you want to use,
# Make sure to use an image with the same architecture as the
# Instance type you are planning to use on your fleets.
FROM public.ecr.aws/amazonlinux/amazonlinux
#
# Game build directory
# -----
# Add your gamebuild directory to the env variable below.
# The game build provided here needs to be integrated with server sdk for Amazon
GameLift Servers.
ENV GAME_BUILD_DIRECTORY="<ADD_GAME_BUILD_DIRECTORY>" \
#
# Game executable and launch parameters
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
# The game build provided over here needs to be integrated with server sdk for Amazon
GameLift Servers.
# This template assumes that the executable path is relative to the game build
directory.
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
# Add 'HOME_DIR' to identify where the game executable and logs exist.
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME DIRECTORY>" \
LAUNCH_PARAMS=<ADD LAUNCH PARAMETERS> \
HOME_DIR="/local/game" \

# Install dependencies as necessary
RUN yum install -y shadow-utils

RUN mkdir -p $HOME_DIR
COPY ./GAME_BUILD_DIRECTORY/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR
```

```
# Set up for 'gamelift' user
RUN useradd -m gamelift && \
    echo "gamelift ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
    chown -R gamelift:gamelift $HOME_DIR

# Add permissions to game build
RUN chmod +x ./${GAME_EXECUTABLE}

USER gamelift

# Check directory before starting the container
RUN ls -lhrt .

# Check path before starting the container
RUN echo $PATH

# Start the game build
ENTRYPOINT ["/bin/sh", "-c", "./${GAME_EXECUTABLE}", "${LAUNCH_PARAMS}"]
```

Amazon ECR에 컨테이너 이미지 푸시

Amazon GameLift Servers에 배포할 컨테이너 이미지를 생성한 후 Amazon ECR의 퍼블릭 또는 프라이빗 리포지토리에 이미지를 저장합니다. 이 리포지토리에 Amazon GameLift Servers를 사용하여 컨테이너 플릿에 배포할 이미지의 스냅샷을 생성하는 URI 값이 할당됩니다.

Note

Amazon ECR 프라이빗 리포지토리가 아직 없는 경우 [하나를 생성](#)합니다.

Amazon ECR에 컨테이너 이미지를 푸시하려면

1. Amazon ECR 자격 증명을 가져옵니다. 컨테이너 이미지를 Amazon ECR에 푸시하기 전에 먼저 임시 형식으로 AWS 자격 증명을 획득하여 Docker에 제공합니다. Docker가 로그인하려면 이러한 자격 증명도 필요합니다.

```
[~/work/glc]$ aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
WARNING! Your password will be stored unencrypted in
/home/user-name/.docker/config.json.
Configure a credential helper to remove this warning.
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store
```

Login Succeeded

2. 사용하려는 [Amazon ECR 프라이빗 리포지토리](#)의 URI를 복사합니다.
3. 컨테이너 이미지를 Amazon ECR 태그를 적용합니다.

Example

```
[~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository URI>:<optional tag>
```

4. Amazon ECR에 컨테이너 이미지를 푸시합니다.

Example

```
[~/work/glc]$ docker image push <Amazon ECR private repository URI>
```

Amazon GameLift Servers와 게임 클라이언트/서버 상호 작용

Amazon GameLift Servers 호스팅 솔루션의 구성 요소는 플레이어 수요에 따라 게임 세션을 실행하는 특정 방식으로 서로 상호 작용합니다. 이 주제에서는 Amazon GameLift Servers 관리형 EC2 플릿, 자체 관리형 Amazon GameLift Servers Anywhere 플릿 또는 하이브리드 솔루션에서 게임 서버를 호스팅할 때 구성 요소가 서로 통신하는 방법을 설명합니다.

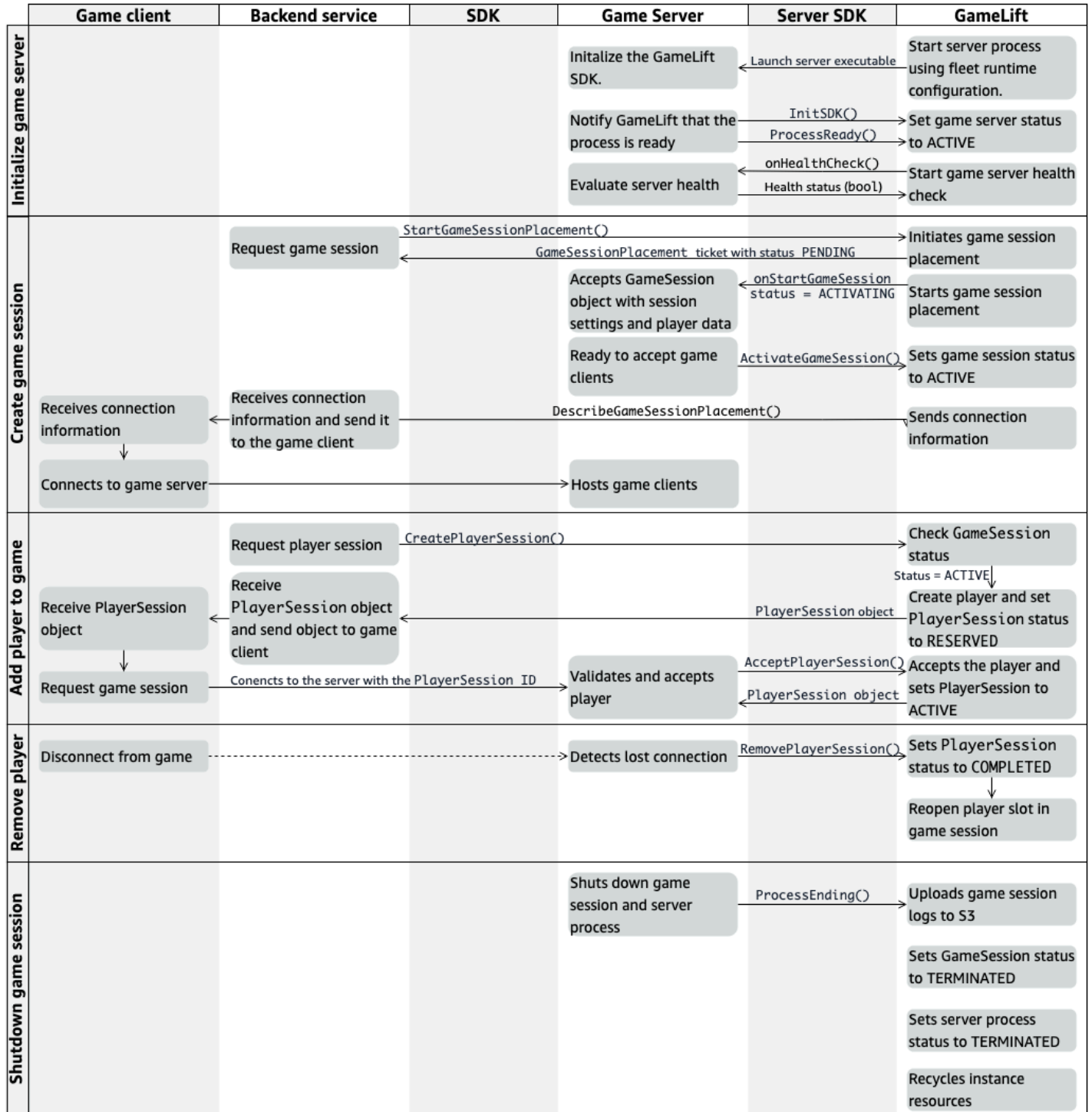
호스팅 솔루션 구성 요소에는 게임 서버, Amazon GameLift Servers 서비스, 클라이언트 측 백엔드 서비스 및 게임 클라이언트가 포함됩니다. 게임 서버는 [Amazon GameLift Servers 서버 SDK](#)를 사용하여 Amazon GameLift Servers 서비스와 상호 작용합니다. 백엔드 서비스는 [Amazon GameLift Servers 서비스 API](#)(AWS SDK의 일부)를 사용하여 게임 클라이언트를 대신하여 서비스와 상호 작용합니다. 게임 세션 연결 시 게임 클라이언트가 게임 세션의 고유한 IP 주소와 포트 번호를 사용하여 게임 세션에 직접 연결합니다.

주제

- [상호 작용 다이어그램](#)
- [상호 작용 동작](#)

상호 작용 다이어그램

다음 다이어그램은 Amazon GameLift Servers 서비스가 게임 서버 가용성 상태를 추적하고 플레이어 수요에 대응하여 게임 세션을 시작할 수 있도록 게임 호스팅 구성 요소가 상호 작용하는 방법을 보여줍니다.



상호 작용 동작

다음 섹션에서는 각 키 상호 작용의 이벤트 시퀀스를 설명합니다.

게임 서버 프로세스 초기화

시작 시 게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 통신을 설정하고 게임 세션을 호스팅할 준비가 되었다고 상태를 보고합니다.

1. 게임 서버 실행 파일의 새 프로세스가 호스팅 리소스에서 실행되기 시작합니다.
2. 게임 서버 프로세스는 다음 서버 SDK 작업을 순서대로 호출합니다.
 - a. `InitSDK()`로 서버 SDK를 초기화하고, 서버 프로세스를 인증하며, Amazon GameLift Servers 서비스와의 통신을 설정합니다.
 - b. `ProcessReady()` 게임 세션을 호스팅할 준비가 되었음을 알립니다. 또한 이 호출은 프로세스의 연결 정보, 게임 클라이언트가 게임 세션에 연결하는 데 사용하는 정보 및 기타 정보를 보고합니다.

그런 다음 서버 프로세스가 Amazon GameLift Servers 서비스의 프롬프트를 기다립니다.

3. Amazon GameLift Servers가 새 게임 세션을 호스팅할 수 있도록 서버 프로세스의 상태를 ACTIVE로 업데이트합니다.
4. Amazon GameLift Servers가 `onHealthCheck` 콜백을 주기적으로 호출하여 서버 프로세스에서 상태를 요청하기 시작합니다. 이러한 호출은 서버 프로세스가 활성 상태로 유지되는 동안 계속됩니다. 서버 프로세스는 1분 내에 정상 또는 비정상을 보고해야 합니다. 서버 프로세스가 정상 상태가 아니거나 응답하지 않는 경우 Amazon GameLift Servers 서비스가 서버 프로세스의 활성 상태를 변경하고 게임 세션 시작 요청 전송을 중지합니다.

게임 세션 생성

Amazon GameLift Servers 서비스는 플레이어의 게임 플레이 요청에 따라 새 게임 세션을 시작합니다.

1. 게임 클라이언트를 사용하는 플레이어가 게임 세션 연결을 요청합니다. 게임이 플레이어 연결 프로세스를 처리하는 방식에 따라 게임 클라이언트가 백엔드 서비스에 요청을 보냅니다.
2. 플레이어 연결 프로세스에서 새 게임 세션을 시작해야 하는 경우 백엔드 서비스가 Amazon GameLift Servers 서비스에 새 게임 세션에 대한 요청을 보냅니다. 이 요청은 서비스 API 작업 `StartGameSessionPlacement()`를 호출합니다. (또는 백엔드 서비스가 `StartMatchmaking()` 또는 `CreateGameSession()`을 호출할 수 있습니다)

3. Amazon GameLift Servers 서비스가 PENDING 상태의 새 GameSessionPlacement 티켓을 생성하여 응답합니다. 티켓 정보를 백엔드 서비스에 반환하여 배치 티켓 상태를 추적하고 게임 세션이 플레이어를 위해 준비된 시점을 확인할 수 있습니다. 자세한 내용은 [게임 세션 배치의 이벤트 알림 설정](#) 단원을 참조하십시오.
4. Amazon GameLift Servers가 게임 세션 배치 프로세스를 시작합니다. 어떤 폴릿을 조회할지 식별하고 해당 폴릿에서 게임 세션을 호스팅하지 않는 활성 서버 프로세스를 검색합니다. 사용 가능한 서버 프로세스를 찾으면 Amazon GameLift Servers 서비스가 다음을 수행합니다.
 - a. 게임 세션 설정과 배치 요청의 플레이어 데이터로 GameSession 객체를 생성하고 상태를 ACTIVATING으로 설정합니다.
 - b. 서버 프로세스에 게임 세션을 시작하라는 메시지를 표시합니다. 서비스가 서버 프로세스의 onStartGameSession 콜백을 호출하고 GameSession 객체를 전달합니다.
 - c. 서버 프로세스의 게임 세션 수를 1로 변경합니다.
5. 서버 프로세스가 onStartGameSession 콜백 함수를 실행합니다. 서버 프로세스가 플레이어 연결을 수락할 준비가 되면 서버 SDK 작업 ActivateGameSession()을 호출하고 플레이어 연결을 기다립니다.
6. Amazon GameLift Servers 서비스는 서버 프로세스의 연결 정보로 GameSession 객체를 업데이트하고(ProcessReady()에 대한 호출이 보고한 대로) 게임 세션 상태를 ACTIVE으로 설정합니다. 또한 GameSessionPlacement 티켓 상태를 FULFILLED로 업데이트합니다.
7. 백엔드 서비스가 DescribeGameSessionPlacement()를 호출하여 티켓 상태를 확인하고 게임 세션 정보를 가져옵니다. 게임 세션이 활성화되면 백엔드 서비스가 게임 클라이언트에 알리고 게임 세션 연결 정보를 전달합니다.
8. 게임 클라이언트가 연결 정보를 사용하여 게임 서버 프로세스에 직접 연결하고 게임 세션에 연결합니다.

게임에 플레이어 추가

게임은 플레이어 세션을 선택적으로 사용하여 게임 세션에 대한 플레이어 연결을 추적할 수 있습니다. 플레이어 세션이 개별적으로 또는 게임 세션 배치 요청의 일부로 생성될 수 있습니다.

1. 백엔드 서비스가 게임 세션 ID를 사용하여 서비스 API 작업 CreatePlayerSession()을 호출합니다.
2. Amazon GameLift Servers 서비스가 게임 세션 상태(반드시 ACTIVE)를 확인하고 게임 세션에서 열린 플레이어 슬롯을 찾습니다. 슬롯이 발견되면 서비스가 다음을 수행합니다.
 - a. 새로운 PlayerSession 객체를 만들고 상태를 RESERVED로 설정합니다.

- b. 플레이어 세션 정보를 사용하여 백엔드 서비스 요청에 응답합니다.
3. 백엔드 서비스가 플레이어 세션 정보와 게임 세션 연결 정보를 게임 클라이언트에 전달합니다.
4. 게임 클라이언트가 연결 정보와 플레이어 세션 ID를 사용하여 게임 서버 프로세스에 직접 연결하고 게임 세션에 연결합니다.
5. 게임 클라이언트 연결 시도에 대한 응답으로 게임 서버 프로세스가 서비스 API 작업 `AcceptPlayerSession()`을 호출하여 플레이어 세션 ID를 검증합니다. 서버 프로세스가 연결을 수락하거나 거부합니다.
6. Amazon GameLift Servers 서비스가 다음 중 하나를 수행합니다.
 - a. 연결이 수락되면 Amazon GameLift Servers가 `PlayerSession` 상태를 `ACTIVE`으로 설정하고 `PlayerSession`을 게임 서버 프로세스에 전달합니다.
 - b. 게임 서버 프로세스가 원래 `CreatePlayerSession()` 요청 후 특정 기간 내에 플레이어 세션 ID에 대한 `AcceptPlayerSession()`을 호출하지 않는 경우 Amazon GameLift Servers 서비스가 `PlayerSession` 상태를 `TIMEDOUT`으로 변경하고 게임 세션의 플레이어 슬롯을 다시 엽니다.

플레이어 제거

플레이어 세션을 사용하는 게임의 경우 플레이어 연결이 끊어지면 게임 서버 프로세스가 Amazon GameLift Servers 서비스에 알립니다. 서비스는 이 정보를 사용하여 게임 세션의 플레이어 슬롯 상태를 추적하고 새 플레이어가 열린 슬롯을 사용하도록 허용할 수 있습니다.

1. 플레이어가 게임 연결을 해제합니다.
2. 게임 서버 프로세스가 끊어진 연결을 찾아내고 서버 SDK 작업 `RemovePlayerSession()`을 호출합니다.
3. Amazon GameLift Servers 서비스가 플레이어 세션 상태를 `COMPLETED`으로 변경하고 게임 세션에서 플레이어 슬롯을 다시 엽니다.

게임 세션 종료

게임 세션이 종료되거나 게임 세션을 종료할 때 서버 프로세스가 Amazon GameLift Servers 서비스에 게임 세션 상태를 알립니다.

1. 게임 서버 프로세스가 게임 세션을 종료하고 서버 SDK 작업 `ProcessEnding()`을 호출하여 프로세스 종료를 시작합니다.

2. 게임 서버 프로세스를 호출 Destroy()하여 서버 SDK를 메모리에서 해제합니다. 이 단계는 정상 프로세스 종료를 충돌로 보고하지 않도록 [원격 측정 지표](#)에 필요합니다.
3. Amazon GameLift Servers 서비스가 다음을 수행합니다.
 - a. 게임 세션 로그를 Amazon Simple Storage Service(S3)에 업로드합니다.
 - b. 게임 세션 상태를 TERMINATED로 변경합니다.
 - c. 서버 프로세스 상태를 TERMINATED로 변경합니다.
 - d. 호스팅 솔루션의 설계 방식에 따라 새로 사용 가능한 호스팅 리소스가 새 게임 서버 프로세스를 실행하도록 할당됩니다.

Amazon GameLift Servers에 대한 백엔드 서비스 구축

플레이어를 인증하고 Amazon GameLift Servers와 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 GameLift Servers가 새 게임 세션 또는 기존 게임 세션에 추가할 플레이어를 그룹화하는 방법을 제어합니다.
- 자체 리소스에서 정보를 수집하여 클라이언트를 신뢰하는 대신 매치메이킹을 위한 스킬 등급과 같은 게임 세션 플레이어 속성을 제공합니다.

또한, 게임 클라이언트 서비스를 사용하면 게임 클라이언트가 Amazon GameLift Servers 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 블로그AWS 용)

Amazon GameLift Servers 게임 클라이언트 기능 통합

게임 세션 정보를 가져오거나 새 게임 세션을 생성하는 등 Amazon GameLift Servers 서비스와 통신해야 하는 모든 솔루션 구성 요소에 Amazon GameLift Servers 게임 호스팅 기능을 통합합니다. 대부분의 경우 이 기능은 백엔드 서비스 구성 요소에 내장되어 있습니다. 다음을 포함한 핵심 작업에 대한 기능을 추가합니다.

- 활성 게임 세션의 정보 및 상태를 요청합니다.
- 기존 게임 세션에서 새 플레이어 슬롯을 예약합니다.
- 플레이어 그룹을 위한 새 게임 세션을 생성합니다.
- 한 명 또는 여러 명의 플레이어에 대한 매치메이킹을 요청합니다.
- 기존 게임 세션에 대해 업데이트된 데이터를 제공합니다.

게임 호스팅 구성 요소가 Amazon GameLift Servers SDK를 통해 상호 작용하는 방식에 대한 자세한 내용은 [Amazon GameLift Servers와 게임 클라이언트/서버 상호 작용](#) 섹션을 참조하세요.

Amazon GameLift Servers API를 설정합니다.

AWS SDK for C++ 를 Amazon GameLift Servers 프로젝트에 추가합니다.

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

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

Amazon GameLift Servers에 문의할 때 클라이언트 구성은 사용할 대상 리전 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 Servers 클라이언트를 초기화합니다. 기본 클라이언트 구성 또는 사용자 지정 클라이언트 구성으로 [Aws::GameLift::GameLiftClient\(\)](#)(C++) 또는 [AmazonGameLiftClient\(\)](#)(C#)를 사용합니다.
3. 플레이어 세션을 사용 중인 경우 각 플레이어에 대한 고유 식별자를 생성하는 메커니즘을 추가합니다. 자세한 내용은 [플레이어 ID 생성](#) 단원을 참조하십시오.
4. 다음 정보를 수집하고 저장합니다.

- 대상 플릿 - 특정 플릿에서 게임 세션을 수동으로 생성하는 경우 대상 플릿을 가리키는 플릿 ID 또는 별칭 ID를 제공합니다. 플릿이 다중 위치인 경우 플릿 위치를 지정합니다. 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 배치 요청을 보낼 대기열 이름을 지정합니다.
- AWS 자격 증명 -에 대한 모든 호출은 게임을 호스팅 AWS 계정 하는에 대한 자격 증명을 제공해야 Amazon GameLift Servers 합니다. [게임의 프로그래밍 방식 액세스 설정](#)에 설명된 대로 플레이어 사용자를 생성하여 이러한 자격 증명을 획득할 수 있습니다. 플레이어 사용자의 액세스를 관리하는 방법에 따라 다음을 수행합니다.
 - 역할을 사용하여 플레이어 사용자 권한을 관리하는 경우 Amazon GameLift Servers API를 호출하기 전에 역할을 수임하는 코드를 추가합니다. 역할 수임 요청은 임시 보안 자격 증명 집합을 반환합니다. 자세한 내용은 [IAM 사용 설명서의 IAM 역할\(AWS API\)로 전환](#)을 참조하세요.
 - 장기 보안 자격 증명에 있는 경우 저장된 자격 증명을 찾아 사용하도록 코드를 구성합니다. SDK 및 도구 참조 안내서의 [장기 자격 증명을 사용한 인증](#)을 참조하세요. AWS SDKs 자격 증명 저장에 대한 자세한 내용은 [\(C++\)](#) 및 [\(.NET\)](#)에 대한 AWS API 참조를 참조하세요.
 - 임시 보안 자격 증명에 있는 경우 IAM 사용 설명서의 [AWS SDK에서 임시 보안 자격 증명 사용](#)에 설명된 대로 AWS Security Token Service (AWS STS)를 사용하여 자격 증명을 정기적으로 새로 고치는 코드를 추가합니다. 이전 자격 증명에 만료되기 전에 코드는 새 자격 증명을 요청해야 합니다.

활성 게임 세션 가져오기

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

활성 게임 세션 검색

[SearchGameSessions](#)를 사용하여 특정 게임 세션, 모든 활성 세션 또는 일련의 검색 기준을 충족하는 세션에 대한 정보를 가져옵니다. 이 호출은 검색 요청과 일치하는 각 활성 게임 세션에 대한 [GameSession](#) 객체를 반환합니다. 이 객체에는 게임 클라이언트가 게임 세션에 연결하는 데 필요한 DNS 이름, IP 주소 및 포트를 포함합니다.

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 데이터 관리

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

- [DescribeGameSessionDetails\(\)](#) - 게임 세션 정보 외에도 게임 세션의 보호 상태를 가져옵니다.
- [UpdateGameSession\(\)](#) - 필요에 따라 게임 세션의 메타데이터와 설정을 변경합니다.
- [GetGameSessionLogUrl](#) - 저장된 게임 세션 로그에 액세스합니다.

게임 세션 만들기

코드를 추가하여 배포된 플릿에서 새 게임 세션을 시작하고 플레이어가 사용할 수 있게 허용합니다. 게임 세션 생성에는 게임 호스팅 솔루션이 게임 세션 배치를 관리하는 방식에 따라 두 가지 옵션이 있습니다.

다중 위치 대기열을 사용하여 게임 세션 만들기

[StartGameSessionPlacement](#)를 사용하여 대기열에 새 게임 세션에 대한 요청을 배치합니다. 이 작업을 사용하려면 대기열을 만듭니다. 이는 Amazon GameLift Servers가 새 게임 세션을 배치할 위치를 결정합니다. 대기열 및 사용 방법에 대한 자세한 내용은 [게임 세션 배치 구성](#) 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 리전의 플레이어 지연 시간 데이터를 포함하는 경우 Amazon GameLift Servers는 이 정보를 사용하여 플레이어에게 이상적인 게임 플레이 경험을 제공하는 새 게임 세션을 플릿에 배치합니다.

정확한 지연 시간 측정값을 얻으려면 Amazon GameLift Servers의 UDP 핑 비콘을 사용합니다. 이러한 엔드포인트를 사용하면 플레이어 디바이스와 잠재적 호스팅 위치 간의 실제 UDP 네트워크 지연 시간을 측정할 수 있으므로 ICMP Ping을 사용하는 것보다 더 정확한 배치 결정을 내릴 수 있습니다. UDP 핑 비콘을 사용하여 지연 시간을 측정하는 방법에 대한 자세한 내용은 [UDP 핑 비콘](#) 섹션을 참조하세요.

게임 세션 배치는 비동기식 프로세스입니다. 요청한 후에는 이 요청을 성공하거나 시간 초과되도록 할 수 있습니다. 또한 언제든지 [StopGameSessionPlacement](#)를 사용하여 요청을 취소할 수 있습니다. 배치 요청의 상태를 확인하려면 [DescribeGameSessionPlacement](#)를 호출합니다.

특정 플릿에서 게임 세션 생성

[CreateGameSession](#)을 사용하여 지정된 플릿에서 새 세션을 만듭니다. 이 동기식 작업을 해당 플릿에 새 게임 세션을 호스팅하는 데 사용할 수 있는 리소스가 있는지 여부에 따라 성공 또는 실패가 결정됩니다. Amazon GameLift Servers가 새 게임 세션을 만들고 [GameSession](#) 객체를 반환한 후에는 플레이어를 해당 객체에 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 카-값 페어의 배열에 정의됩니다.

Amazon GameLift Servers 리소스 보호 기능을 사용하여 한 플레이어가 생성할 수 있는 게임 세션 수를 제한할 경우 게임 세션 생성자의 플레이어 ID를 제공합니다.

게임 세션에 플레이어 참여

코드를 추가하여 활성 게임 세션에서 플레이어 슬롯을 예약하고 게임 클라이언트를 게임 세션에 연결합니다. 이 작업은 게임이 고유한 플레이어 ID를 가진 플레이어 세션을 사용하는 경우에 사용할 수 있습니다. 플레이어 세션에 대한 자세한 내용은 [Amazon GameLift Servers 및 플레이어 경험](#) 섹션을 참조하세요.

1. 게임 세션에서 플레이어 슬롯을 예약합니다.

플레이어 슬롯을 예약하려면 게임 세션에 대해 플레이어 세션을 만듭니다. 이렇게 하는 방법은 두 가지입니다.

- [StartGameSessionPlacement](#) 또는 [StartMatchmaking](#)을 사용하여 게임 세션을 생성하는 경우 요청에 하나 이상의 플레이어를 포함하여 새 게임 세션에서 플레이어 세션을 생성할 수 있습니다.
- 기존 게임 세션에 플레이어를 추가하려면 게임 세션 ID를 사용하여 [CreatePlayerSession](#) 또는 [CreatePlayerSessions](#)를 호출합니다.

플레이어 세션 요청은 고유한 플레이어 ID를 포함해야 합니다. 자세한 내용은 [플레이어 ID 생성](#) 단원을 참조하십시오. 요청을 수신하면 Amazon GameLift Servers는 먼저 게임 세션이 새 플레이어를 수락하고 사용 가능한 플레이어 슬롯을 보유하는지 확인합니다. 성공하면 Amazon GameLift Servers는 플레이어를 위한 슬롯을 예약하고, 새 플레이어 세션을 생성하며, [PlayerSession](#) 객체를 반환합니다.

플레이어 세션이 사용자 지정 플레이어 데이터 집합을 포함할 수 있습니다. 이 데이터는 새로 생성된 플레이어 세션 객체에 저장됩니다. Amazon GameLift Servers는 플레이어가 게임 세션에 직접

연결할 때 이 객체를 게임 서버에 전달합니다. 여러 플레이어 세션을 요청하는 경우 각 플레이어 데이터 문자열을 플레이어 ID에 매핑합니다.

2. 게임 세션에 연결

백엔드 서비스에 코드를 추가하여 `PlayerSession` 객체를 검색하고(예:

[DescribePlayerSessions\(\)](#) 호출) 게임 클라이언트에 다시 전달합니다. 이 객체는 DNS 이름, IP 주소, 포트를 포함합니다. 게임 클라이언트는 이 정보를 사용하여 서버에 직접 연결을 설정할 수 있습니다.

- 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.

연결된 후 Amazon GameLift Servers의 개입 없이 게임 클라이언트와 서버 프로세스가 직접 통신합니다. 서버는 Amazon GameLift Servers와 통신을 유지하여 플레이어 연결 상태, 상태 확인 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.

사용자 지정 게임 데이터를 게임 세션에 전달

게임 클라이언트는 게임 세션에 데이터를 전달할 수 있습니다. 게임 세션을 생성할 때 일련의 게임 속성(키-값 페어) 또는 게임 세션 데이터(문자열 값)를 요청에 포함할 수 있습니다. 기존 게임 세션을 새 게임 데이터 또는 업데이트된 게임 데이터로 업데이트할 수도 있습니다. 이 데이터는 게임 서버 코드에서 사용할 수 있도록 게임 세션을 호스팅하는 게임 서버 프로세스로 전달됩니다. 게임 속성은 삭제할 수 없습니다.

예를 들어 게임에서 난이도 수준을 Novice, Easy, Intermediate, Expert로 정의한다고 가정해 보겠습니다. 플레이어가 Easy 게임에 참여하기로 선택합니다. 게임 클라이언트는 백엔드 서비스를 통해 다음 게임 속성을 사용하여 새 게임 세션을 요청합니다. `{"Key": "Difficulty", "Value": "Easy"}`. 이에 대한 응답으로 Amazon GameLift Servers는 사용 가능한 게임 서버에 새 게임 세션을 시작하라는 메시지를 표시하고 `GameSession` 객체를 전달합니다. 게임 서버 프로세스는 제공된 게임 속성을 사용하여 게임 세션의 난이도를 설정합니다.

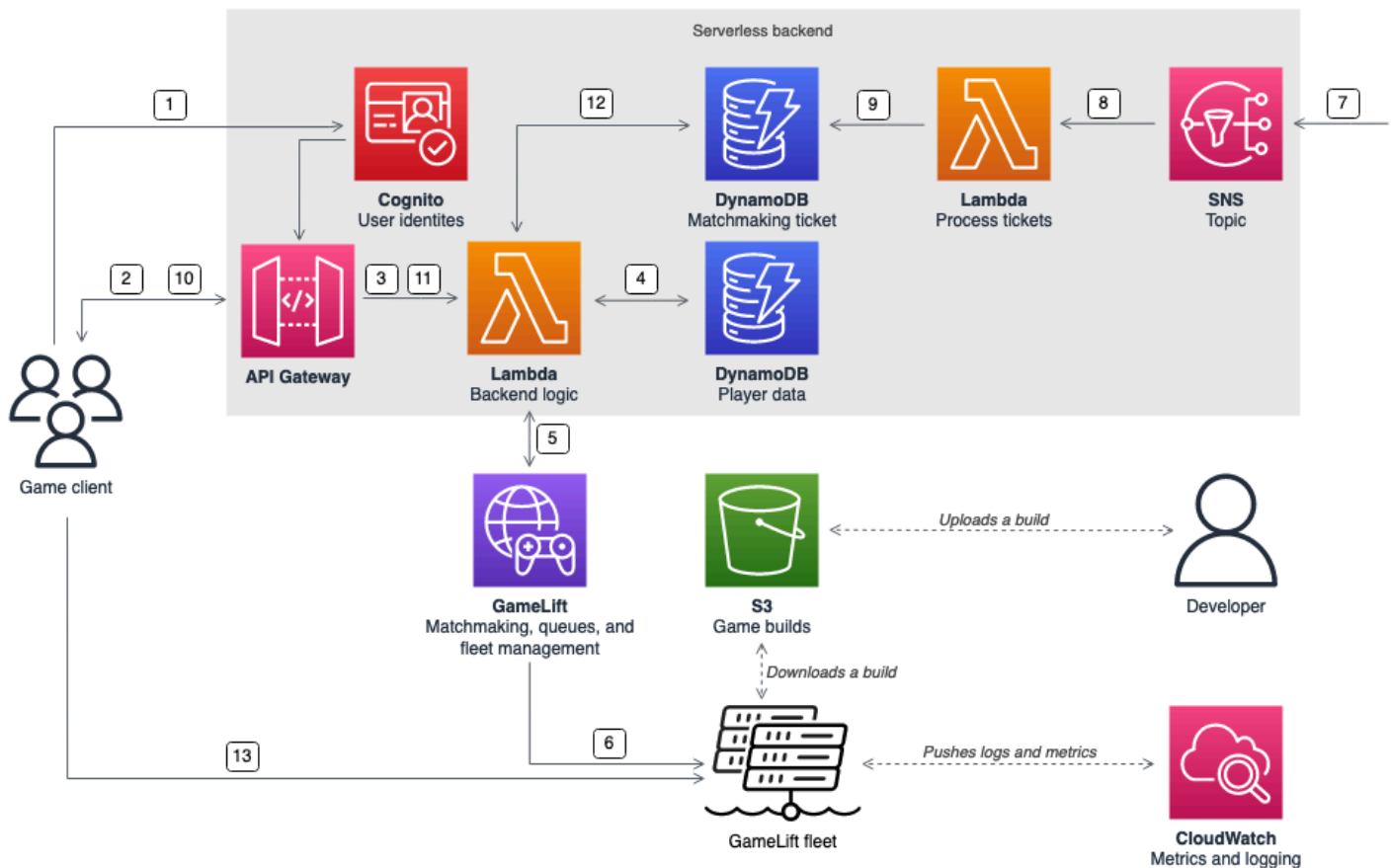
자세히 알아보기

- [GameProperty 데이터 유형](#)
- [SearchGameSessions\(\) 예제](#)
- [UpdateGameSession\(\) GameProperties 파라미터](#)

서버리스 백엔드가 있는 독립형 게임 세션 서버

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

다음 다이어그램은 플레이어를 Amazon GameLift Servers 플릿에서 실행되는 게임에 매칭 AWS 서비스 하는 로 빌드된 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.

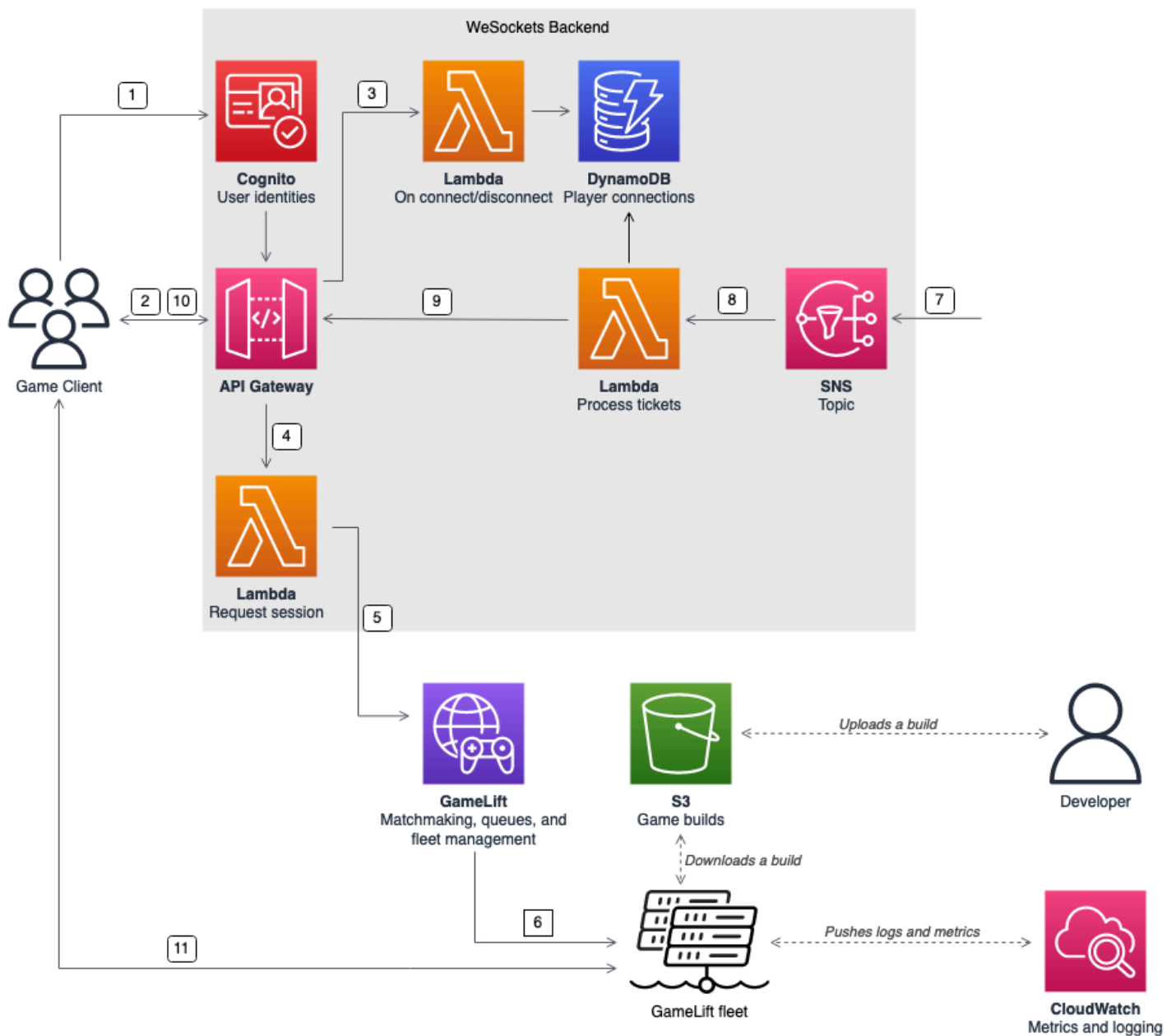
2. 게임 클라이언트는 Amazon API Gateway API를 통해 임시 액세스 자격 증명을 받고 게임 세션을 요청합니다.
3. API Gateway는 AWS Lambda 함수를 호출합니다.
4. Lambda 함수는 Amazon DynamoDB NoSQL 테이블에서 플레이어 데이터를 요청합니다. 이 함수는 요청 컨텍스트 데이터에서 Amazon Cognito 자격 증명을 제공합니다.
5. Lambda 함수는 Amazon GameLift Servers FlexMatch 매치메이킹을 통해 매칭을 요청합니다.
6. FlexMatch는 적절한 지연 시간을 가진 플레이어 그룹을 일치시킨 다음 Amazon GameLift Servers 대기열을 통해 게임 세션 배치를 요청합니다. 대기열에는 하나 이상의 AWS 리전 위치가 있는 플릿이 있습니다.
7. Amazon GameLift Servers가 플릿 위치 중 하나에 세션을 배치하면 Amazon GameLift Servers는 Amazon Simple Notification Service(Amazon 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 Servers용 서버 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 작업](#)을 참조하세요.

다음 다이어그램은 API Gateway 및 기타를 사용하여 플레이어 AWS 서비스 를 Amazon GameLift Servers 플릿에서 실행되는 게임에 매칭하는 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 Servers FlexMatch 매치메이킹을 통해 매칭을 요청합니다.
6. 가 플레이어 그룹과 FlexMatch 일치하면는 Amazon GameLift Servers 대기열을 통해 게임 세션 배치를 FlexMatch 요청합니다.
7. Amazon GameLift Servers가 플릿 위치 중 하나에 세션을 배치하면 Amazon GameLift Servers는 Amazon Simple Notification Service(Amazon 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 Servers용 서버 SDK를 사용하여 ID를 검증합니다.

Amazon GameLift Servers용 호스팅 플릿 배포

호스팅 리소스를 배포하려면 게임 서버를 실행할 컴퓨팅 인프라를 생성하고 구성해야 합니다.는 완전 관리형 AWS 클라우드 리소스부터 클라우드 및 온프레미스 인프라를 결합하는 하이브리드 솔루션에 이르기까지 다양한 호스팅 요구 사항에 맞는 여러 플릿 유형을 Amazon GameLift Servers 제공합니다.

비용, 제어, 확장성, 지리적 배포에 대한 요구 사항에 가장 적합한 플릿 유형을 선택합니다. 또한 단일 호스팅 솔루션에서 여러 플릿 유형을 결합하여 다양한 시나리오 또는 플레이어 수에 맞게 최적화할 수 있습니다.

플릿 특성

Amazon GameLift Servers 플릿은 게임 서버를 실행하고 플레이어용 게임 세션을 호스팅하는 컴퓨팅 리소스 컬렉션입니다. 플릿은 사용하는 컴퓨팅 리소스 유형과 플릿 관리 방식에 따라 다를 수 있습니다. 플릿의 크기, 즉 플릿이 지원할 수 있는 게임 세션 및 플레이어의 수는 사용자가 제공하는 컴퓨팅 리소스의 수에 따라 달라집니다. Amazon GameLift Servers 플릿의 공통적인 특징은 다음과 같습니다.

- 모든 플릿에서 실행되는 게임 서버 프로세스는 Amazon GameLift Servers용 서버 SDK와 통합되어 동일한 방식으로 Amazon GameLift Servers 서비스와 통신합니다. 게임 서버는 게임 세션 및 플레이

어를 호스팅하고, 게임 세션 및 기타 상호 작용을 시작하거나 중지하라는 프롬프트에 응답할 수 있는 가용성을 보고합니다.

- Amazon GameLift Servers는 모든 플릿의 게임 세션 배치를 동일한 방식으로 처리합니다. Amazon GameLift Servers는 플릿의 게임 서버 상태를 계속 추적하고 사용 가능한 게임 서버 중에서 선택하여 새 게임 세션을 호스팅합니다. 이 프로세스는 게임이 단일 플릿에 게임 세션을 배치하든 [게임 세션 대기열](#)을 사용하여 여러 플릿에 걸쳐 호스팅의 균형을 맞추든 관계없이 사용됩니다. 대기열을 사용하면 배치 결정을 사용자 지정하여 리소스 비용 및 지연 시간과 같은 요인을 고려할 수도 있습니다.
- 모든 플릿은 게임 세션 배치 대기열과 공동 작업하여 FlexMatch 매치메이커 사용을 지원합니다. Amazon GameLift Servers 서비스는 플레이어 매치 요청을 수신하고, 매치를 구성하고, 이를 게임 세션 대기열에 전달하여 사용 가능한 게임 서버를 찾습니다.
- Amazon GameLift Servers는 광범위한 플릿 지표를 수집합니다. 여기에는 컴퓨팅 및 서버 프로세스에 대한 상태 지표와 게임 세션 및 플레이어 활동에 대한 사용량 지표가 포함됩니다. 사용 가능한 지표 전체 목록은 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#) 섹션을 참조하세요.

주제

- [결정 가이드: 호스팅 옵션 선택](#)
- [Amazon GameLift Servers 관리형 EC2 플릿](#)
- [Amazon GameLift Servers 관리형 컨테이너 플릿](#)
- [Amazon GameLift Servers Anywhere 플릿](#)
- [하이브리드 호스팅 솔루션 구축](#)

결정 가이드: 호스팅 옵션 선택

Amazon GameLift Servers는 다양한 호스팅 시나리오 및 요구 사항에 맞게 설계된 여러 호스팅 옵션을 제공합니다. 이 가이드는 게임의 기술적 요구 사항, 운영 기본 설정 및 비즈니스 제약 조건에 따라 플릿 유형을 선택하는 데 도움이 됩니다. 자세한 설명은 [Amazon GameLift Servers 게임 호스팅 옵션](#) 섹션을 참조하세요.

플릿 비교

다음 표에서는 각 호스팅 옵션의 주요 특성을 비교합니다.

호스팅 옵션	최적의 용도	관리 레벨	주요 이점
관리형 EC2	전체 AWS 관리를 통한 프로덕션 호스팅	완전관리형	Auto Scaling, 글로벌 도달 범위, 최소 운영 오버헤드
관리형 컨테이너	오케스트레이션이 필요한 컨테이너화된 애플리케이션	완전관리형	컨테이너 오케스트레이션, 리소스 효율성, 최신 배포
Anywhere	온프레미스, 하이브리드 또는 개발 환경	자체 관리형	하드웨어 제어, 규정 준수 요구 사항, 비용 최적화

주요 결정 요인

호스팅 옵션을 선택할 때 다음 요소를 고려합니다.

운영 복잡성

얼마나 많은 인프라 관리를 처리하려고 하나요? 관리형 플릿은 최소 운영 오버헤드를 필요로 하지만, Anywhere 플릿은 전체 제어 기능을 제공하므로 더 많은 관리가 필요합니다.

확장성 요구 사항

플레이어 수요에 따라 자동 규모 조정이 필요하나요? 관리형 플릿은 내장된 Auto Scaling을 제공하는 반면, Anywhere 플릿은 수동 용량 관리가 필요합니다.

지리적 분포

플레이어의 위치는 어디인가요? 관리형 플릿은 전 세계 여러 AWS 리전에 배포할 수 있지만 Anywhere 플릿은 자체 인프라 위치에 따라 달라집니다.

규정 준수 및 보안

특정 규정 준수 요구 사항이 있거나 데이터를 온프레미스에 보관해야 하나요? Anywhere 플릿을 사용하면 인프라 및 데이터 위치를 완전히 제어할 수 있습니다.

비용 고려 사항

예산 및 비용 최적화 전략은 무엇인가요? 옵션을 비교할 때 인프라 비용과 운영 오버헤드를 모두 고려합니다.

하이브리드 솔루션

게임 호스팅 솔루션에 대해 한 가지 유형의 플릿만 선택할 필요는 없습니다. 수많은 게임들은 다음과 같은 다양한 시나리오에 맞게 최적화하도록 여러 플릿 유형을 결합하는 하이브리드 접근 방식을 사용합니다.

- 개발 및 테스트: 개발 및 테스트를 위해 Anywhere 플릿을 사용한 다음 프로덕션을 위해 관리형 플릿에 배포합니다.
- 리전 최적화: 플레이어 밀도가 높은 AWS 리전 의 관리형 플릿과 기존 인프라가 있는 리전의 Anywhere 플릿을 사용합니다.
- 비용 최적화: 기존 용량에 관리형 플릿을 사용하고 오버플로 또는 특정 워크로드에 Anywhere 플릿을 사용합니다.

Amazon GameLift Servers 관리형 EC2 플릿

Amazon GameLift Servers 관리형 EC2 플릿은 프로덕션 호스팅을 위한 클라우드 기반 리소스를 제공합니다. 관리형 플릿을 사용하면 멀티플레이어 게임 호스팅에 최적화된 AWS 클라우드 리소스의 유연성, 보안 및 신뢰성을 얻을 수 있습니다. 강력한 호스트 관리 도구를 Amazon GameLift Servers 제공합니다.

관리형 EC2 플릿은 구성을 기반으로 Amazon GameLift Servers가 소유하고 운영하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 세트입니다. 이러한 인스턴스는 지원되는 AWS 리전 또는 로컬 영역에 물리적으로 위치합니다. 플릿을 생성할 때 컴퓨팅 성능, 메모리, 스토리지, 네트워킹 기능에 대한 게임 서버의 요구 사항을 충족하는 EC2 인스턴스 유형을 선택합니다.

플릿의 각 인스턴스를 시작할 때 Amazon GameLift Servers는 필요한 런타임 환경을 사용하여 게임 서버 빌드를 배포합니다. 런타임 환경은 플릿이 생성될 때 사용할 수 있는 최신 Amazon Machine Image(AMI) 버전을 사용합니다. 플릿의 모든 인스턴스는 동일한 AMI 버전을 사용합니다.

Note

모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 이를 위해서는 새 플릿을 생성하고 플레이어 트래픽을 해당 플릿으로 마이그레이션해야 합니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#) 단원을 참조하십시오.

인스턴스에 런타임 환경과 게임 서버 빌드를 설치한 후 Amazon GameLift Servers는 게임 서버 프로세스를 시작합니다. 각 게임 서버 프로세스는 Amazon GameLift Servers 서비스에 대한 연결을 설정하고, 게임 세션을 호스팅할 준비를 보고하며, 상태 통신을 시작합니다. 그러면 Amazon GameLift Servers가 서버 프로세스에 게임 세션을 시작하라는 메시지를 표시할 수 있습니다.

플릿 배포 외에도 Amazon GameLift Servers는 다음과 같은 호스트 관리 작업을 처리하므로 다음 작업을 별도로 수행할 필요가 없습니다.

- 플릿의 모든 컴퓨팅 상태를 추적하고 오래된 컴퓨팅 또는 비정상 컴퓨팅을 대체합니다.
- 서버 프로세스와 Amazon GameLift Servers 서비스 간의 통신에 대한 인증을 처리합니다.
- 런타임 구성에 따라 각 컴퓨팅에서 게임 서버 프로세스를 자동으로 시작하고 중지합니다.
- 플레이어 수요에 맞게 플릿 용량을 동적으로 조정하도록 오토 스케일링 도구를 제공합니다.
- 플릿의 EC2 인스턴스에 대한 성능 지표를 보고합니다.

관리형 EC2 플릿을 설정하고 유지하는 방법에 대한 다음 주제를 참조하세요.

- [Amazon GameLift Servers 관리형 EC2를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers 관리형 EC2 플릿 생성](#)
- [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)
- [리소스 사용자 지정 호스팅](#)
- [Amazon GameLift Servers 플릿 구성 업데이트](#)

관리형 EC2 플릿 생성 워크플로:

관리형 플릿의 경우 Amazon GameLift Servers는 플릿 리소스를 설정하고 게임 서버 소프트웨어가 설치 및 실행 중인 컴퓨팅 리소스 집합도 배포합니다. 생성 워크플로가 성공적으로 완료되면 플릿에는 활성 EC2 인스턴스가 플릿 홈 리전과 플릿의 원격 위치에 각각 하나씩 있게 됩니다. 모든 인스턴스에 게임 세션을 호스팅할 준비가 된 게임이 있습니다.

1. Amazon GameLift Servers는 플릿의 홈 리전에 플릿 리소스를 생성하고 하나의 인스턴스당 각 위치에 필요한 용량을 설정합니다. 플릿 및 위치 상태는 신규로 설정됩니다.
2. Amazon GameLift Servers는 플릿 이벤트 로그에 이벤트를 기록하기 시작합니다.
3. Amazon GameLift Servers가 플릿 상태를 다운로드 중으로 설정하고 게임 서버 소프트웨어 배포 준비를 시작합니다.
 - a. 업로드된 게임 서버 빌드를 가져오고 압축된 파일을 추출합니다.

- b. 제공된 경우 설치 스크립트를 실행합니다.
 - c. 플릿 상태를 검증 중으로 설정하고 빌드 파일을 다운로드하고 설치할 때 오류가 발생하지 않았는지 확인하기 시작합니다.
4. Amazon GameLift Servers가 플릿 상태를 빌드 중으로 설정하고 플릿 하드웨어를 구성한 후, 각 플릿 인스턴스에 대해 하나의 EC2 인스턴스를 할당합니다.
 5. Amazon GameLift Servers는 플릿 상태를 활성화 중으로 설정합니다. 각 인스턴스에서 게임 서버 프로세스를 시작하고(플릿의 런타임 지침에 따라) 빌드와 Amazon GameLift Servers 서비스 간의 연결을 테스트합니다.
 6. 각 인스턴스에서 게임 서버 프로세스가 연결을 설정하고 게임 세션을 호스팅하기 위한 준비 상태를 보고하면 Amazon GameLift Servers가 플릿 및 위치 상태를 활성으로 설정합니다. 이 시점에서 플릿은 게임 세션을 호스팅할 준비가 된 것으로 간주됩니다.

Amazon GameLift Servers 관리형 EC2 플릿 생성

이 주제에서는 Amazon GameLift Servers 관리형 EC2 플릿을 생성하는 방법을 설명합니다. 관리형 플릿은 멀티플레이어 게임 호스팅에 최적화된 Amazon Elastic Compute Cloud(Amazon EC2) 컴퓨팅 인스턴스를 사용합니다. 에서 지원하는 AWS 리전 및 로컬 영역에 전역적으로 컴퓨팅을 배포하는 관리형 플릿을 생성할 수 있습니다Amazon GameLift Servers.

새 관리형 EC2 플릿 리소스를 생성하면 플릿 생성의 첫 번째 단계를 즉시 시작합니다. 관리형 플릿 생성은 Amazon GameLift Servers가 EC2 인스턴스를 배포하고, 인스턴스에 런타임 환경과 게임 서버 빌드를 설치하며, 게임 서버 시작을 시작할 때 여러 단계를 거칩니다. 게임 서버 빌드에 필요한 런타임 환경에 따라 플릿 생성 시 최신 버전의 Amazon Machine Image(AMI)를 Amazon GameLift Servers 배포합니다(플릿의 향후 모든 인스턴스는 동일한 버전을 사용함).콘솔에서 또는 AWS Command Line Interface ()를 사용하여 플릿의 상태를 모니터링할 수 있습니다AWS CLI. 플릿이 ACTIVE 상태로 변경되면 게임 세션을 호스팅할 준비가 된 것입니다. 관리형 플릿 생성에 대한 자세한 내용은 다음 항목을 참조하세요.

Note

모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 이를 위해서는 새 플릿을 생성하고 플레이어 트래픽을 해당 플릿으로 마이그레이션해야 합니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#) 단원을 참조하십시오.

관리형 EC2 플릿 생성

Amazon GameLift Servers 콘솔 또는 AWS Command Line Interface (AWS CLI)를 사용하여 관리형 EC2 플릿을 생성합니다.

Console

[Amazon GameLift Servers 콘솔](#)의 탐색 창에서 관리형 EC2, 플릿 페이지를 실행합니다. 관리형 EC2 플릿 생성을 선택하여 플릿 생성 워크플로를 시작합니다.

1단계: 관리형 EC2 플릿 세부 정보 정의

최소 플릿 구성:

- 플릿 이름을 입력합니다.
- 바이너리 형식: 빌드를 선택합니다. 이전에 Amazon GameLift Servers에 업로드한 서버 빌드 목록에서 빌드를 선택합니다.
- 기본 설정을 사용하려면 추가 세부 정보 및 태그 섹션을 건너뜁니다.

1. 플릿 세부 정보 섹션을 작성합니다.

- a. 플릿 이름을 입력합니다. 플릿 이름은 설명 목적으로만 사용되며 플릿 목록을 정렬하거나 필터링하는 데 유용합니다. 반드시 고유할 필요는 없습니다.
- b. 플릿에 대해 캡처하려는 사용자 지정 정보를 포함하는 짧은 플릿 설명을 제공합니다.
- c. 바이너리 형식, 빌드를 선택하고 드롭다운 목록을 사용하여 이전에 Amazon GameLift Servers에 업로드한 게임 서버 빌드를 선택합니다.

2. 필요에 따라 추가 세부 정보를 설정합니다.

- a. 게임 서버 실행 파일이 계정의 다른 AWS 리소스에 액세스해야 하는 경우 필요한 권한이 있는 IAM 인스턴스 역할을 지정합니다. 다른 서버 측 애플리케이션(예: CloudWatch 에이전트)을 승인하는 방법 등에 관한 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 섹션을 참조하세요. 플릿을 생성한 후에는 이 설정을 변경할 수 없습니다.

플릿을 생성하기 전에 역할을 먼저 생성해야 합니다. 추가로, 인스턴스 역할이 있는 플릿을 생성하려면 AWS 사용자에게 IAM PassRole 권한이 있어야 합니다([Amazon GameLift Servers의 IAM 권한 예제](#) 참조).

- b. 게임에 대한 인증 및 암호화를 설정하려면 TLS 인증서 생성 옵션을 켭니다. 게임 클라이언트는 이 인증서를 사용하여 모든 클라이언트/서버 통신을 연결하고 암호화할 때 게임 서버를 인증합니다. TLS 활성화 플릿에 있는 각 인스턴스에 대해 Amazon GameLift Servers는 인증서를 사용하여 새 DNS 항목도 생성합니다. 플릿을 생성한 후에는 이 설정을 변경할 수 없습니다.
 - c. Amazon GameLift Servers는 각 개별 플릿에 대한 지표 데이터를 내보냅니다. 여러 플릿의 지표 데이터를 결합하려면 지표 그룹 이름을 지정합니다. 결합하려는 모든 플릿에 동일한 지표 그룹 이름을 사용합니다. CloudWatch를 사용하여 집계된 지표 그룹 데이터를 봅니다.
3. 플릿 리소스에 태그를 추가합니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 용도, 소유자 또는 환경별로 분류하려는 AWS 리소스에 태그를 할당합니다. 추가하려는 각 추가 태그에 대해 새 태그 추가를 선택합니다.
4. 워크플로를 계속하려면 다음을 선택합니다.

2단계 인스턴스 세부 정보 정의

이 단계에서는 게임 서버에 사용할 하드웨어 리소스 유형을 선택하고 이를 배포할 위치를 지정합니다. 여러 위치를 선택하면 게임 서버를 더 넓은 지리적 위치에 배포할 수 있으므로 플레이어에 더 가깝게 배치하고 지연 시간을 최소화할 수 있습니다. 플릿의 모든 EC2 인스턴스는 동일한 유형이어야 합니다. 모든 위치에서 모든 인스턴스 유형을 사용할 수 있는 것은 아닙니다.

최소 플릿 구성:

- 홈 위치에만 배포하는 것으로 시작합니다. 초기 개발 및 테스트를 위해 원격 위치를 추가할 필요가 없습니다.
- 플릿 유형을 “온디맨드”로 설정합니다. 스팟 플릿에는 추가 설정 작업이 필요합니다.
- 인스턴스 유형을 “c5.large”로 설정합니다. 일반적으로 사용되는 이 인스턴스 유형은 모든에서 사용할 수 있습니다 AWS 리전. 나중에 하드웨어 사용량을 최적화할 수 있습니다.

1. 인스턴스 배포에서 플릿 위치 및 유형을 지정합니다.
 - a. 플릿 인스턴스를 배포하려는 위치를 하나 이상 선택합니다. 이러한 원격 위치는 이 플릿을 생성하는 AWS 리전 플릿의 홈 위치(미리 선택됨)에 추가됩니다. 가 Amazon

GameLift Servers 지원하는 모든 AWS 리전 및 로컬 영역에서 원격 위치를 선택할 수 있습니다.

기본적으로 활성화되지 AWS 리전 애플을 사용하는 방법을 포함하여 지원되는 위치에 대한 자세한 내용은 관리형 호스팅 [Amazon GameLift Servers 서비스 위치](#)은 섹션을 참조하세요. 또한 플릿당 위치에 대한 Amazon GameLift Servers [할당량](#)을 검토합니다.

- b. 이 플릿에 온디맨드를 사용할지 또는 스팟 인스턴스를 사용할지 선택합니다. 플릿 유형에 대한 자세한 내용은 [온디맨드 인스턴스 및 스팟 인스턴스 비교](#) 섹션을 참조하세요.
2. 요구 사항을 충족하고 선택한 모든 위치에서 사용할 수 있는 Amazon EC2 인스턴스 구성을 선택합니다. 이 목록은 현재 위치 및 플릿 유형 선택에 따라 필터링됩니다. 인스턴스 유형 패밀리 및 아키텍처와 같은 다른 요인을 기준으로 추가로 필터링할 수 있습니다. 플릿을 생성한 후에는 인스턴스 유형을 변경할 수 없습니다.

일부 위치에는 인스턴스 유형 옵션이 제한됩니다. 원하는 인스턴스 유형을 모든 위치에 사용할 수 없는 경우 위치 가용성 값을 선택하여 전체 세부 정보를 확인합니다. 모든 위치를 수용하려면 인스턴스 유형이 다른 별도의 플릿을 생성해야 할 수 있습니다.

인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 [인스턴스 유형](#) 섹션을 참조하세요. Amazon EC2 Arm 아키텍처에 대한 자세한 내용은 [AWS Graviton 프로세서](#) 및 [Amazon EC2 인스턴스 유형](#)을 참조하세요. Amazon GameLift Servers에서 지원하는 인스턴스 유형의 전체 목록은 [EC2InstanceType](#) (CreateFleet())에 대한 API 참조를 참조하세요.

Note

Graviton Arm 인스턴스의 경우 Linux AMI에서 서버 빌드를 사용해야 합니다. 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.

3. 워크플로를 계속하려면 다음을 선택합니다.

4단계: 런타임 구성

이 단계에서는 플릿의 각 인스턴스가 게임 서버 소프트웨어를 실행하는 방법을 설명합니다. 인스턴스에서 실행할 각 실행 파일에 대해 별도의 서버 프로세스 라인 항목을 정의하고 동시에 실행할 각 서버 프로세스의 수를 결정합니다. 각 인스턴스의 포트를 열어 플레이어가 게임 서버에 직접 연결할 수 있도록 합니다. 플릿 설정은 언제든지 업데이트할 수 있습니다.

i 최소 플릿 구성:

- 게임 서버 실행 파일에 대한 단일 서버 프로세스 라인 항목을 정의합니다. 게임 서버에서 다른 프로세스를 실행해야 하는 경우 이러한 각 프로세스에 대한 정의도 생성합니다.
- 각 라인 항목에 대해 기본 동시 프로세스 수(1)를 사용합니다.
- 게임 세션 활성화 설정을 건너뛰니다.
- 단일 포트 번호를 지정합니다.
- 게임 세션 리소스 설정을 건너뛰니다.

1. 런타임 구성을 생성하여 플릿의 각 인스턴스에서 서버 프로세스를 실행하는 방법을 Amazon GameLift Servers에 지시합니다. 배포 후 언제든지 플릿의 런타임 구성을 변경할 수 있습니다.
 - a. 빌드에서 실행 파일의 시작 경로를 입력합니다. Windows 인스턴스에서 게임 서버 실행 파일은 C:\game 경로에 빌드됩니다. Linux 인스턴스에서 게임 서버는 /local/game에 빌드됩니다. 예: **C:\game\MyGame\server.exe** 또는 **/local/game/MyGame/server.exe**.
 - b. 게임 실행 파일에 전달할 선택적 시작 파라미터를 입력합니다. 예시: **+sv_port 33435 +start_lobby**.
 - c. 각 인스턴스에서 실행할 동시 프로세스 수를 지정합니다. 게임 서버 실행 파일의 경우 각 프로세스가 하나의 게임 세션을 호스팅할 수 있으므로 동시 프로세스에 따라 인스턴스가 동시에 호스팅할 수 있는 게임 세션 수가 결정됩니다.

인스턴스당 서버 프로세스에 대한 Amazon GameLift Servers [할당량](#)을 검토합니다. 이러한 할당량은 모든 구성의 총 동시 프로세스에 적용됩니다. 한도를 초과하여 플릿을 구성하면 플릿이 활성화되지 않습니다.

2. 게임 세션 활성화 기본값을 사용하거나 게임에 맞게 사용자 지정합니다. 런타임 구성이 인스턴스당 여러 개의 동시 게임 서버 프로세스를 요구하는 경우 이러한 설정은 새 게임 세션을 시작할 수 있는 속도를 결정합니다.
 - a. 새 게임 세션을 준비하는 인스턴스의 게임 서버 수를 제한하도록 최대 동시 게임 세션 활성화를 설정합니다. 이 설정은 시작하는 여러 새 게임 세션이 리소스 집약적이며 실행 중인 다른 게임 세션의 성능에 영향을 미칠 때 유용합니다.

- b. 새 게임 세션이 활성화를 완료하고 플레이어를 호스팅할 준비가 되었음을 보고하는 데 걸리는 최대 시간을 반영하도록 새 활성화 제한 시간을 설정합니다. 이 값을 초과하는 경우 Amazon GameLift Servers가 게임 세션 활성화를 종료합니다.
3. 인바운드 트래픽이 플릿의 서버 프로세스 액세스를 허용하려면 EC2 포트 설정을 엽니다. 이러한 설정은 플릿을 생성하는 데 필요하지 않지만 플레이어가 플릿의 게임 세션에 연결하기 전에 설정해야 합니다.

각 포트 설정에서 게임 클라이언트와 게임 서버 간의 통신에 사용할 데이터 전송 프로토콜 유형을 선택합니다. CIDR 표기법(예: 모든 사람에게 액세스를 허용하는 **0.0.0.0/0**)을 사용하여 포트 범위(nnnnnn[-nnnnn] 형식)와 IP 주소 범위를 제공합니다.

여러 개의 비연속 범위를 설정해야 하는 경우 여러 포트 설정을 생성합니다.

4. 선택적 게임 세션 리소스 설정을 지정합니다. 배포 후 언제든지 이러한 설정을 업데이트할 수 있습니다.
 - a. 플릿의 모든 인스턴스에 대해 게임 조정 보호 정책을 켜거나 끕니다. 스케일 다운 이벤트 중에는 Amazon GameLift Servers가 활성 게임 세션을 호스팅하고 있는 경우, 보호 중인 플릿 인스턴스를 종료하지 않습니다.
 - b. 지정된 기간 동안 한 플레이어가 생성할 수 있는 게임 세션 수를 제한하려면 최대 리소스 생성 제한을 설정합니다.
5. 워크플로를 계속하려면 다음을 선택합니다.

5단계: 검토 및 생성

플릿을 생성하기 전에 설정을 검토합니다. 나중에 일부 설정을 업데이트할 수 있지만([Amazon GameLift Servers 플릿 구성 업데이트](#) 참조) 플릿이 생성된 후에는 다음 설정을 변경할 수 없습니다.

- 컴퓨팅 유형: 관리형 EC2 플릿을 Anywhere 플릿으로 변환할 수 없습니다.
- 게임 서버 바이너리: 게임 서버 빌드에 업데이트를 배포하려면 새 플릿을 생성해야 합니다.
- 인스턴스 역할 및 TLS 인증서 생성을 포함한 추가 옵션
- 플릿 유형(스팟 또는 온디맨드) 및 EC2 인스턴스 유형을 포함한 인스턴스 세부 정보

새 플릿을 배포할 준비가 되면 생성을 선택합니다. Amazon GameLift Servers가 플릿 활성화 프로세스를 즉시 시작하여 고유한 ID를 할당하고 플릿을 NEW 상태로 전환합니다. 플릿 페이지에서 플릿의 진행 상황을 추적합니다. 플릿의 세부 정보 페이지를 확인하고 이벤트 탭으로 이동합니다.

플릿이 ACTIVE 상태에 도달한 후 플릿의 호스팅 용량을 조정할 수 있습니다. Amazon GameLift Servers는 처음에 각 플릿 위치에 단일 인스턴스가 있는 플릿을 배포하며 각 위치에 인스턴스를 추가하여 용량을 조정합니다. 자세한 내용은 [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#) 단원을 참조하십시오.

AWS CLI

[create-fleet](#) 명령을 사용하여 컴퓨팅 유형 EC2의 플릿을 생성합니다. Amazon GameLift Servers는 현재 기본 AWS 리전 에서 플릿 리소스를 생성합니다(또는 다른 AWS 리전에 리전 태그 추가 지정 가능).

최소 관리형 플릿 생성

다음 예제 요청은 게임 클라이언트가 연결할 수 있는 실행 중인 게임 서버가 있는 플릿을 배포하는데 필요한 최소 설정으로 새 플릿을 생성합니다. 신규 플릿의 특징은 다음과 같습니다.

- Amazon GameLift Servers에 업로드되고 READY 상태인 게임 서버 빌드를 지정합니다.
- 선택한 게임 빌드와 일치하는 운영 체제와 함께 c5.large 온디맨드 인스턴스를 사용합니다.
- 플릿의 홈을 us-west-2 AWS 리전 로 설정하고 인스턴스를 해당 리전에만 배포합니다.
- 런타임 구성을 기반으로 플릿의 각 컴퓨팅은 하나의 게임 서버 프로세스를 실행하므로 각 컴퓨팅은 한 번에 하나의 게임 세션만 호스팅할 수 있습니다. 게임 세션 활성화 제한 시간은 기본값인 300초로 설정되며 동시 활성화 횟수에는 제한이 없습니다.
- 플레이어는 33435의 단일 포트 설정을 사용하여 게임 서버에 연결할 수 있습니다.
- 다른 모든 기능은 꺼져 있거나 기본 설정을 사용합니다.

```
aws gamelift create-fleet \
  --name MinimalFleet123 \
  --description "A basic test fleet" \
  --region us-west-2 \
  --ec2-instance-type c5.large \
  --fleet-type ON_DEMAND \
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --runtime-configuration "ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated\
\MultiplayerSampleProjectLauncher_Server.exe, ConcurrentExecutions=10}]" \
  --ec2-inbound-permissions
  "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
```

완전 구성 관리형 플릿 생성

다음 예제 요청은 모든 선택적 기능에 대한 설정을 사용하여 프로덕션 플릿을 생성합니다. 신규 플릿의 특징은 다음과 같습니다.

- Amazon GameLift Servers에 업로드되고 READY 상태인 게임 서버 빌드를 지정합니다.
- 플릿은 선택한 게임 빌드와 일치하는 운영 체제와 함께 c5.large 온디맨드 인스턴스를 사용합니다.
- 플릿의 홈을 us-west-2 AWS 리전 로 설정하고 인스턴스를 홈 리전과 하나의 원격 위치에 배포합니다sa-east-1.
- 런타임 구성에 따라
 - 플릿의 각 컴퓨팅은 동일한 시작 파라미터로 10개의 게임 서버 프로세스를 실행하므로 각 컴퓨팅은 최대 10개의 게임 세션을 동시에 호스팅할 수 있습니다.
 - 각 컴퓨팅에서 2개의 게임 세션만 동시에 활성화할 수 있습니다. 게임 세션을 활성화하려면 300초(5분) 이내에 플레이어를 호스팅하거나 종료할 준비가 되어 있어야 합니다.
- 플레이어는 33435 to 33535 범위의 포트를 사용하여 게임 서버에 연결할 수 있습니다.
- 게임 클라이언트와 서버 간의 암호화된 통신을 위한 TLS 인증서를 생성합니다.
- 이 플릿의 모든 게임 세션에는 게임 세션 보호가 설정되어 있습니다.
- 각 플레이어는 15분 내에 3개의 새 게임 세션을 만들 수 있습니다.
- 이 플릿에 대한 지표는 지표 그룹 AMERfleets에 포함되며, (이 예제에서는) 북미, 중남미의 플릿 그룹에 대한 지표를 집계합니다.

```
aws gamelift create-fleet \
  --name ProdFleet123 \
  --description "A fully configured prod fleet" \
  --ec2-instance-type c5.large \
  --region us-west-2 \
  --locations "Location=sa-east-1" \
  --fleet-type ON_DEMAND \
  --build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
  --certificate-configuration "CertificateType=GENERATED" \
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
  MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=C:\game
  \Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe, Parameters+=sv_port
  33435 +start_lobby, ConcurrentExecutions=10}]" \
  --new-game-session-protection-policy "FullProtection" \
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
  PolicyPeriodInMinutes=15" \
```

```
--ec2-inbound-permissions  
"FromPort=33435,ToPort=33535,IpRange=0.0.0.0/0,Protocol=UDP" \  
--metric-groups "AMERfleets"
```

create-fleet 요청에 성공하면 Amazon GameLift Servers는 사용자가 요청한 구성 설정과 새 플릿 ID를 포함하는 플릿 속성 세트를 반환합니다. 그런 다음 Amazon GameLift Servers는 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](#)

Amazon GameLift Servers 관리형 컨테이너 플릿

Amazon GameLift Servers 관리형 컨테이너 플릿은 컨테이너화된 게임 서버 소프트웨어를 호스팅하기 위한 클라우드 기반 리소스를 제공합니다. 관리형 컨테이너 플릿을 사용하면 멀티플레이어 게임 호스팅에 최적화된 AWS 클라우드 리소스의 유연성, 보안 및 신뢰성을 얻을 수 있습니다. 강력한 호스트 관리 도구를 Amazon GameLift Servers 제공합니다.

i 관리형 컨테이너용 다음 도구로 온보딩 속도를 높입니다.

- [컨테이너 스타터 키트](#)는 통합 및 플릿 설정을 간소화합니다. 게임 서버에 필수 게임 세션 관리 기능을 추가하고 사전 구성된 템플릿을 사용하여 게임 서버를 위한 컨테이너 플릿과 자동 배포 파이프라인을 빌드합니다. 배포 후 Amazon GameLift Servers 콘솔 및 API 도구를 사용하여 플릿 성능을 모니터링하고, 게임 세션을 관리하며, 지표를 분석합니다.
- Unreal Engine 또는 Unity 개발자의 경우 게임 엔진 [Amazon GameLift Servers 플러그인 및 서버 SDK](#)를 사용하여 게임 서버를 통합하고 게임 엔진의 개발 환경 내에서 컨테이너 플릿을 빌드합니다. 플러그인의 안내 워크플로는 관리형 컨테이너를 사용한 클라우드 기반 호스팅으로 빠르고 간단한 솔루션을 만드는 데 도움이 됩니다. 이 기반을 바탕으로 게임에 대한 사용자 지정 호스팅 솔루션을 생성할 수 있습니다.

관리형 컨테이너 플릿은 구성을 기반으로 Amazon GameLift Servers가 소유하고 운영하는 Linux를 실행하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 세트입니다. 이러한 인스턴스는 지원되는 AWS 리전 또는 로컬 영역에 물리적으로 위치합니다. 컨테이너 플릿을 생성할 때 컴퓨팅 성능, 메모리, 스토리지, 네트워킹 기능에 대한 게임 서버의 요구 사항을 충족하는 EC2 인스턴스 유형을 선택합니다.

관리형 컨테이너 플릿의 경우 Linux 기반 컨테이너 이미지를 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 저장하고 컨테이너 아키텍처를 설명하는 컨테이너 그룹 정의를 생성합니다. 플릿을 생성할 때 Amazon GameLift Servers는 최신 버전의 Linux Amazon Machine Image(AMI)로 플릿 인스턴스를 프로비저닝하고 컨테이너 그룹 정의를 사용하여 컨테이너 이미지를 배포합니다. 컨테이너 그룹 정의를 업데이트하거나 컨테이너 이미지를 변경하더라도 컨테이너 플릿의 모든 인스턴스는 동일한 AMI 버전을 사용합니다.

i Note

모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 이를 위해서는 새 플릿을 생성하고 플레이어 트래픽을 해당 플릿으로 마이그레이션해야 합니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#) 단원을 참조하십시오.

컨테이너화된 인스턴스를 배포한 후 컨테이너는 게임 서버 프로세스를 실행하기 시작합니다. 각 게임 서버 프로세스는 Amazon GameLift Servers 서비스에 대한 연결을 설정하고, 게임 세션을 호스팅할 준

비를 보고하며, 상태 통신을 시작합니다. 그러면 Amazon GameLift Servers가 서버 프로세스에 게임 세션을 시작하라는 메시지를 표시할 수 있습니다.

플릿 배포 외에도 Amazon GameLift Servers는 다음과 같은 호스트 관리 작업을 처리하므로 다음 작업을 별도로 수행할 필요가 없습니다.

- 플릿의 모든 컨테이너 상태를 추적하고 오래되거나 비정상 컨테이너를 대체합니다.
- 서버 프로세스와 Amazon GameLift Servers 서비스 간의 통신에 대한 인증을 처리합니다.
- 플레이어 수요에 맞게 플릿 용량을 동적으로 조정하는 오토 스케일링 도구를 제공합니다.
- 플릿의 EC2 인스턴스, 컨테이너, 서버 프로세스에 대한 성능 지표를 보고합니다.

관리형 컨테이너 플릿을 설정하고 유지하는 방법에 대한 다음 주제를 참조하세요.

- [Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers 관리형 컨테이너 플릿 생성](#)
- [Amazon GameLift Servers 컨테이너 플릿 사용자 지정](#)
- [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#)
- [Amazon GameLift Servers 관리형 컨테이너 플릿 업데이트](#)

Amazon GameLift Servers의 컨테이너 작동 방식

Amazon GameLift Servers 컨테이너 플릿은 컨테이너화된 애플리케이션을 배포하고 확장하는 방법에 유연성을 제공하도록 설계되었습니다. Amazon Elastic Container Service(Amazon ECS)를 사용하여 Amazon GameLift Servers 플릿의 작업 배포 및 실행을 관리합니다. 이 주제에서는 Amazon GameLift Servers 관리형 플릿에서 컨테이너를 실행하기 위한 기본 구조 요소, 일반적인 아키텍처, 몇 가지 핵심 개념에 대해 간략하게 설명합니다.

 관리형 컨테이너용 다음 도구로 온보딩 속도를 높입니다.

- [컨테이너 스타터 키트](#)는 통합 및 플릿 설정을 간소화합니다. 게임 서버에 필수 게임 세션 관리 기능을 추가하고 사전 구성된 템플릿을 사용하여 게임 서버를 위한 컨테이너 플릿과 자동 배포 파이프라인을 빌드합니다. 배포 후 Amazon GameLift Servers 콘솔 및 API 도구를 사용하여 플릿 성능을 모니터링하고, 게임 세션을 관리하며, 지표를 분석합니다.
- Unreal Engine 또는 Unity 개발자의 경우 [Amazon GameLift Servers 플러그인](#)을 사용하여 게임 서버를 통합하고 게임 엔진의 개발 환경 내에서 컨테이너 플릿을 빌드합니다. 플러그인

의 안내 워크플로는 관리형 컨테이너를 사용한 클라우드 기반 호스팅으로 빠르고 간단한 솔루션을 만드는 데 도움이 됩니다. 그런 다음 이 기반을 바탕으로 게임에 대한 사용자 지정 호스팅 솔루션을 생성합니다.

컨테이너 플릿 구성 요소

플릿

컨테이너 플릿은 컨테이너화된 게임 서버를 호스팅하기 위한 Amazon EC2 인스턴스 모음입니다. 이러한 인스턴스는 사용자를 대신하여 Amazon GameLift Servers에서 관리합니다. 플릿을 생성할 때 게임 서버 소프트웨어가 포함된 컨테이너 아키텍처가 각 플릿 인스턴스에 배포되는 방식을 구성합니다. 하나 이상의 지리적 위치에 인스턴스가 있는 컨테이너 플릿을 생성할 수 있습니다. Amazon GameLift Servers 조정 도구로 컨테이너 플릿의 용량을 자동으로 조정하여 게임 세션과 플레이어들을 호스팅할 수 있습니다.

Instance

Amazon EC2 인스턴스는 게임 호스팅을 위한 컴퓨팅 용량을 제공하는 가상 서버입니다. Amazon GameLift Servers를 사용하면 다양한 인스턴스 유형 중에서 선택할 수 있습니다. 인스턴스 유형마다 서로 다른 CPU 조합, 메모리, 스토리지 및 네트워크 기능을 제공합니다.

컨테이너 플릿을 생성할 때 Amazon GameLift Servers는 선택한 인스턴스 유형과 플릿 구성을 기반으로 컨테이너를 배포합니다. 배포된 각 플릿 인스턴스는 동일하며 컨테이너화된 게임 서버 소프트웨어를 동일한 방식으로 실행합니다. 플릿의 인스턴스 수에 따라 플릿의 크기와 게임 호스팅 용량이 결정됩니다.

컨테이너 그룹

Amazon GameLift Servers는 컨테이너 그룹의 개념을 사용하여 컨테이너 세트를 설명하고 관리합니다. 컨테이너 그룹은 컨테이너 '작업' 또는 '포드'와 유사합니다. 각 컨테이너 그룹 내에서 컨테이너의 작동 방식을 정의하고, 종속성을 설정하며, 사용 가능한 CPU 및 메모리 리소스를 공유할 수 있습니다.

각 플릿 인스턴스에는 다음과 같은 유형의 컨테이너 그룹이 있을 수 있습니다.

- 게임 서버 컨테이너 그룹은 게임 서버 애플리케이션 및 지원 소프트웨어를 실행하는 컨테이너를 관리합니다. 게임 세션과 플레이어를 호스팅하려면 컨테이너 플릿에 이러한 유형의 컨테이너 그룹 중 하나가 있어야 합니다. 게임 서버 컨테이너 그룹은 플릿 인스턴스 전체에 복제할 수 있습니다. 플릿 인스턴스당 게임 서버 그룹 복제본 수는 소프트웨어의 컴퓨팅 요구 사항과 인스턴스에서 사용할 수 있는 컴퓨팅 리소스에 따라 달라집니다.

- 선택 사항인 인스턴스당 컨테이너 그룹은 각 플릿 인스턴스에서 추가 소프트웨어를 실행할 수 있는 기능을 제공합니다. 모니터링과 같은 백그라운드 서비스 또는 유틸리티 프로그램을 실행하는데 유용합니다. 게임 서버 소프트웨어는 인스턴스당 그룹의 프로세스에 직접 의존하지 않습니다. 인스턴스당 컨테이너 그룹의 사본 하나만 각 플릿 인스턴스에 배포됩니다.

컨테이너 플릿의 각 컨테이너 그룹에는 '필수'로 지정된 컨테이너가 하나씩 있습니다. 필수 컨테이너는 컨테이너 그룹의 수명 주기를 관리합니다. 필수 컨테이너에서 장애가 발생한 경우 전체 컨테이너 그룹을 다시 시작합니다.

컨테이너

컨테이너는 컨테이너 기반 아키텍처의 가장 기본적인 요소입니다. 여기에는 소프트웨어 실행 파일 및 종속 파일이 있는 컨테이너 이미지가 포함됩니다. 컨테이너를 정의하여 소프트웨어를 실행하고 Amazon GameLift Servers와 상호 작용하는 방식을 구성합니다.

Amazon GameLift Servers는 두 가지 유형의 컨테이너를 정의합니다.

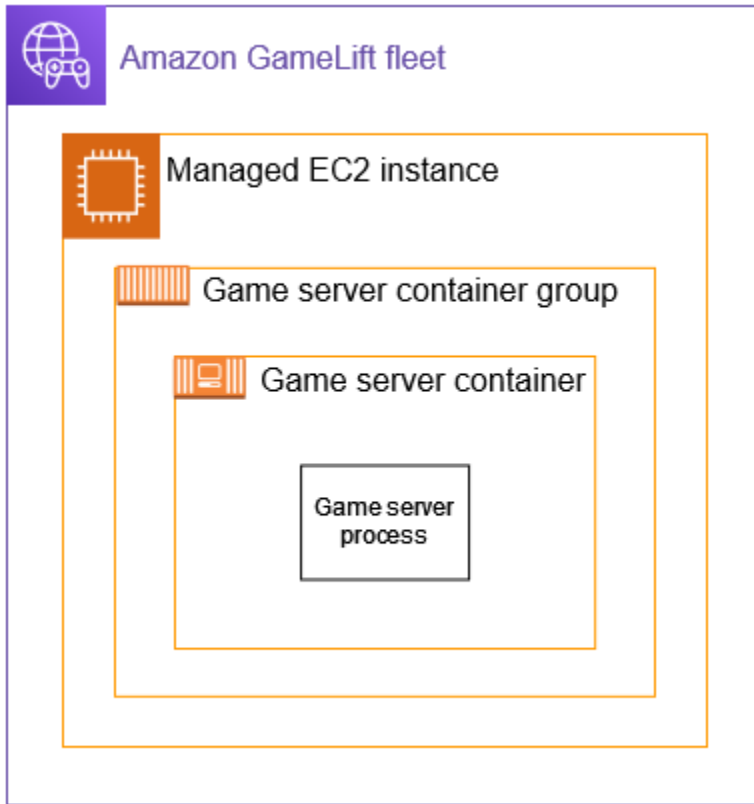
- 게임 서버 컨테이너에는 게임 서버 프로세스를 실행하고 플레이어를 위한 게임 세션을 호스팅하는 데 필요한 모든 것이 포함되어 있습니다. 여기에는 게임 서버 빌드와 종속 소프트웨어가 포함됩니다. 플릿의 게임 서버 컨테이너 그룹에 대해 하나의 게임 서버 컨테이너를 정의합니다. 게임 서버 컨테이너는 컨테이너 그룹에 필수적인 것으로 자동 간주됩니다.
- 지원 컨테이너는 게임 서버를 지원하는 추가 소프트웨어를 실행합니다. 이는 '사이드카' 컨테이너의 개념과 유사합니다. 게임 서버와 함께 지원 소프트웨어를 실행하고 확장할 수 있는 옵션을 제공하지만 별도의 컨테이너로 관리할 수 있습니다. 게임 서버 컨테이너 그룹에서 0개 이상의 지원 컨테이너를 정의할 수 있습니다. 인스턴스당 컨테이너 그룹에서 모든 컨테이너는 지원 컨테이너입니다. 모든 지원 컨테이너는 필수로 지정할 수 있습니다.

컴퓨팅

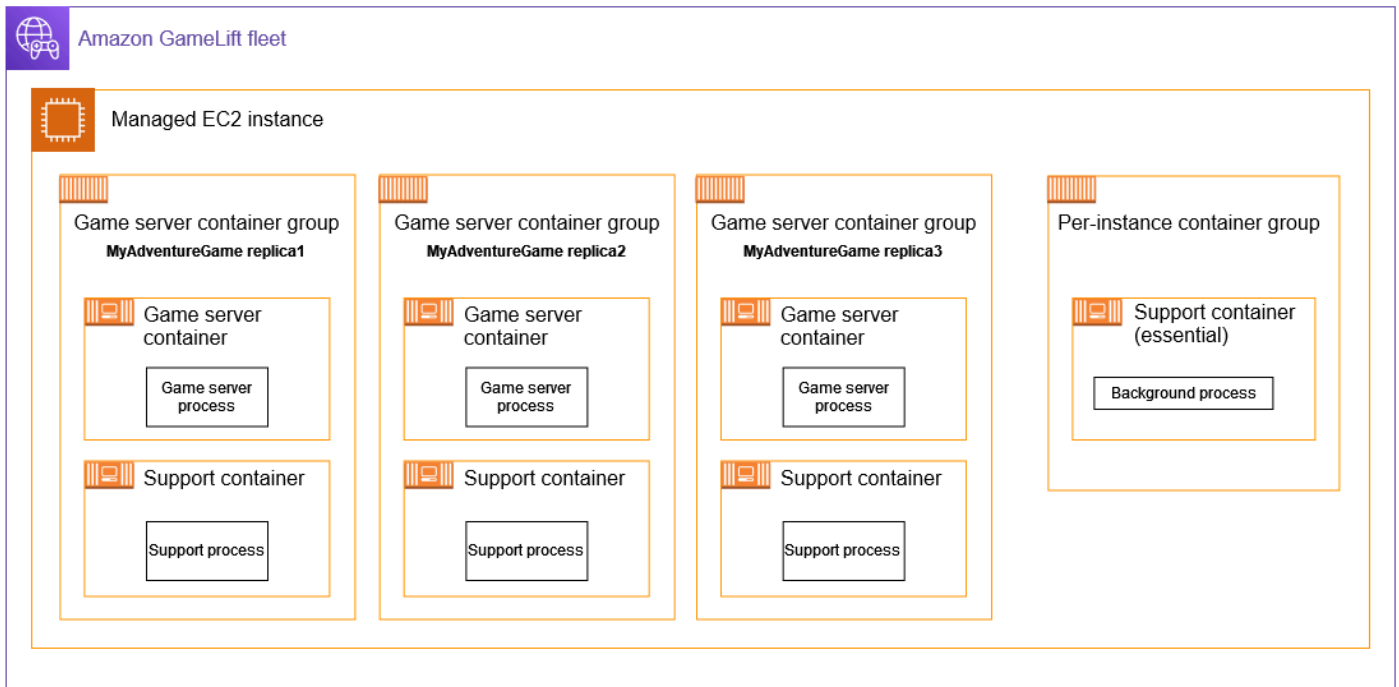
컴퓨팅은 플릿 인스턴스에 있는 게임 서버 컨테이너 그룹의 사본을 나타냅니다.

공통 아키텍처

다음 다이어그램은 가장 간단한 컨테이너 플릿 구조를 보여줍니다. 이 구조에서 플릿의 각 인스턴스는 게임 서버 컨테이너 그룹의 사본 하나를 보관합니다. 컨테이너 그룹에는 하나의 게임 서버 프로세스를 실행하는 단일 게임 서버 컨테이너가 있습니다. 이 예제에서 컨테이너 플릿은 인스턴스당 게임 서버 컨테이너 그룹의 사본 하나를 배치하도록 구성됩니다. 이 아키텍처를 사용하면 각 인스턴스가 하나의 게임 서버 프로세스를 실행합니다.



이 두 번째 다이어그램은 보다 복잡한 컨테이너 플릿 아키텍처를 보여줍니다. 이 구조에서 플릿에는 게임 서버 컨테이너 그룹과 인스턴스당 컨테이너 그룹이 모두 있습니다. 게임 서버 컨테이너 그룹에는 게임 서버 프로세스와 지원 프로세스를 위한 별도의 컨테이너가 있습니다. 플릿은 게임 서버 컨테이너 그룹의 사본 3개를 각 플릿 인스턴스에 배치하도록 구성됩니다. 인스턴스당 컨테이너 그룹은 복제되지 않습니다. 이 예제에서 컨테이너 플릿은 인스턴스당 게임 서버 컨테이너 그룹의 사본 3개를 배치하도록 구성됩니다. 이 아키텍처를 사용하면 각 인스턴스가 3개의 게임 서버 프로세스를 실행합니다.



핵심 기능

이 섹션에서는 Amazon GameLift Servers가 몇 가지 기본 컨테이너 개념을 구현하는 방법에 대해 요약합니다. 컨테이너 플릿 작업 방법에 대한 지침은 이 가이드의 관련 주제를 참조하세요.

활성 플릿 업데이트

관리형 컨테이너는 호스팅 소프트웨어 및 컨테이너 아키텍처의 수명 주기를 관리하는 데 도움이 되는 고급 지원을 제공합니다. 컨테이너 이미지를 포함하여 컨테이너 정의를 업데이트하고 기존 플릿에 변경 사항을 배포할 수 있습니다. 이 기능을 사용하면 개발 중에 컨테이너에 대한 변경 사항을 더 빠르고 쉽게 반복할 수 있습니다. 또한 시간 경과에 따른 소프트웨어 버전 업데이트를 빌드, 배포 및 추적하는 데 도움이 되는 기능도 제공합니다. 이러한 기능에는 다음이 포함됩니다.

- 컨테이너 그룹 정의 업데이트 및 버전 관리를 관리합니다. 컨테이너 이미지 및 구성 설정을 포함하여 컨테이너 그룹 정의의 거의 모든 속성을 업데이트할 수 있습니다. 컨테이너를 업데이트할 때마다 Amazon GameLift Servers는 업데이트에 버전 번호를 자동으로 할당하고 기본적으로 모든 버전을 유지합니다. 특정 버전에 액세스하고 원하는 대로 버전을 삭제할 수 있습니다. 컨테이너 플릿을 생성할 때 배포할 컨테이너 그룹 정의와 버전을 지정할 수 있습니다.
- 기존 컨테이너 플릿을 새 컨테이너 그룹 정의 및 구성 설정으로 업데이트합니다. 플릿 인스턴스에 이미 배포된 플릿에 컨테이너 업데이트를 배포할 수 있습니다. AWS Management Console 또는 AWS SDK 및 CLI를 사용하여 각 플릿 위치에서 업데이트 배포 상태를 추적할 수 있습니다.
- 플릿 업데이트를 활성 플릿에 배포하는 방법을 구성합니다.

- 게임 세션 보호. 게임 세션이 종료(안전한 배포)될 때까지 활성 게임 세션으로 플릿 인스턴스를 보호하도록 선택합니다. 또는 게임 세션 활동(안전하지 않은 배포)에 관계없이 플릿 인스턴스를 교체하도록 선택합니다. 개발 및 테스트 단계에서 안전하지 않은 배포를 사용하여 배포 시간을 줄입니다.
- 최소 건전 백분율. 배포 중에 유지하려는 정상 작업의 비율을 지정합니다. 이 기능을 사용하면 배포 중에 영향을 받는 플릿 인스턴스 수를 결정할 수 있습니다. 값이 낮을수록 배포 속도의 우선 순위가 지정되고 값이 높을수록 배포 전반에 걸쳐 게임 서버 가용성이 높게 유지됩니다.
- 배포 실패 전략. 배포에 실패할 경우 수행할 작업을 결정합니다. 배포 실패는 업데이트된 컨테이너 중 일부가 상태 확인에 실패했으며 손상된 것으로 간주됨을 의미합니다. 모든 플릿 인스턴스를 이전에 배포된 상태로 자동으로 롤백하도록 배포를 설정할 수 있습니다. 또는 디버깅에 사용할 손상된 플릿 인스턴스 중 일부를 유지 관리하도록 선택할 수 있습니다.

활성 플릿을 업데이트하는 기능은 게임 서버 소프트웨어에 업데이트를 배포하려는 경우에 매우 유용합니다. 게임 서버용 새 컨테이너 이미지를 빌드한 후 배포하는 것은 2단계 프로세스입니다. 첫 번째로 새 이미지로 컨테이너 그룹 정의를 업데이트하고, 두 번째로 컨테이너 플릿을 업데이트합니다. Amazon GameLift Servers는 필요에 따라 다른 모든 작업을 처리합니다.

컨테이너 패킹

컨테이너 플릿에 배포할 컨테이너 구조를 개발할 때 일반적인 목표는 사용 가능한 컴퓨팅 성능 사용을 최적화하는 것입니다. 이 목표를 달성하려면 각 플릿 인스턴스에 가능한 한 많은 게임 서버 컨테이너 그룹을 패키징해야 합니다.

Amazon GameLift Servers는 다음 정보를 기반으로 인스턴스당 최대 게임 서버 컨테이너 그룹을 계산하여 이 작업을 수행할 수 있도록 지원합니다.

- 플릿의 인스턴스 유형과 해당 vCPU 및 메모리 리소스입니다.
- 게임 서버 컨테이너 그룹의 모든 컨테이너에 대한 vCPU 및 메모리 요구 사항입니다.

인스턴스별 컨테이너 그룹이 있는 경우 인스턴스당 컨테이너 그룹의 모든 컨테이너에 대한 vCPU 및 메모리 요구 사항입니다.

컨테이너 플릿을 생성할 때 계산된 최대값을 사용하거나 원하는 수를 지정할 수 있습니다. 가장 좋은 방법은 컨테이너화된 게임 서버 소프트웨어를 실험하여 최적의 게임 서버 성능을 위한 리소스 요구 사항을 결정하는 것입니다.

용량 규모 조정

플릿 용량은 플릿이 동시에 호스팅할 수 있는 게임 세션 수를 측정합니다. 플릿이 지원할 수 있는 동시 플레이어 수를 기준으로 용량을 측정할 수도 있습니다. 플릿의 호스팅 용량을 늘리거나 줄이려면 플릿 인스턴스를 추가하거나 제거합니다.

컨테이너 플릿은 각 플릿 인스턴스에서 특정 수의 동시 게임 서버 프로세스를 실행하도록 구성됩니다. ((1) 인스턴스당 게임 서버 컨테이너 그룹 및 (2) 각 컨테이너 그룹에서 실행되는 게임 서버 프로세스 수를 기준으로 이를 계산할 수 있습니다.) 인스턴스당 동시 게임 서버 수는 각 플릿 인스턴스를 추가하거나 제거하는 것이 미치는 영향을 알려줍니다. 예를 들어 컨테이너 플릿이 각 게임 서버 컨테이너 그룹에서 1개의 게임 서버 프로세스를 실행하고 각 플릿 인스턴스에 100개의 게임 서버 컨테이너 그룹이 있는 경우 동시 게임 세션을 호스팅하는 플릿의 용량을 100씩 늘리거나 줄입니다. 각 게임 세션에 10개의 플레이어 슬롯이 있는 경우 플레이어를 호스팅할 플릿의 용량을 1000씩 늘리거나 줄입니다.

컨테이너 플릿을 사용하면 Amazon GameLift Servers에서 제공하는 용량 조정 방법 중 하나를 사용할 수 있습니다. 다음이 포함됩니다.

- 원하는 플릿 인스턴스 수를 설정하여 플릿 용량을 수동으로 설정합니다.
- 사용 가능한 인스턴스의 원하는 버퍼를 대상으로 자동 조정을 설정합니다(대상 추적). 이 방법은 들어오는 플레이어가 게임에 빠르게 참여할 수 있도록 일정량의 유휴 호스팅 리소스를 자동으로 유지합니다. 플레이어 수요가 증가하거나 감소하면 이 버퍼의 크기가 지속적으로 조정됩니다.
- 사용자 지정 조정 규칙을 사용하여 자동 조정을 설정합니다(고급 기능). 이 방법을 사용하면 선택한 플릿 지표에 따라 확장할 수 있습니다.

게임 클라이언트/서버 연결

Amazon GameLift Servers 관리형 플릿을 사용하면 게임 클라이언트가 클라우드 호스팅 게임 서버에 직접 연결됩니다. 게임 클라이언트가 게임에 참여하도록 요청하면 Amazon GameLift Servers는 게임 세션을 찾고 게임 클라이언트에 연결 정보(IP 및 포트)를 제공합니다. 플릿의 특정 포트 범위(인바운드 권한)를 열어 플릿 인스턴스에 대한 외부 액세스를 제어할 수 있습니다. 인바운드 권한은 수신 트래픽에 개방된 포트를 결정합니다. 모든 포트를 빠르게 종료하거나, 몇 개로 제한하거나, 모든 포트를 열 수 있습니다.

관리형 컨테이너 플릿에는 컨테이너에서 실행 중인 프로세스에 대한 액세스를 허용하는 추가 설정이 필요합니다. 컨테이너 정의를 생성할 때 연결을 담당하는 각 프로세스마다 하나씩 포트 세트를 지정합니다. 여기에는 다음이 포함됩니다.

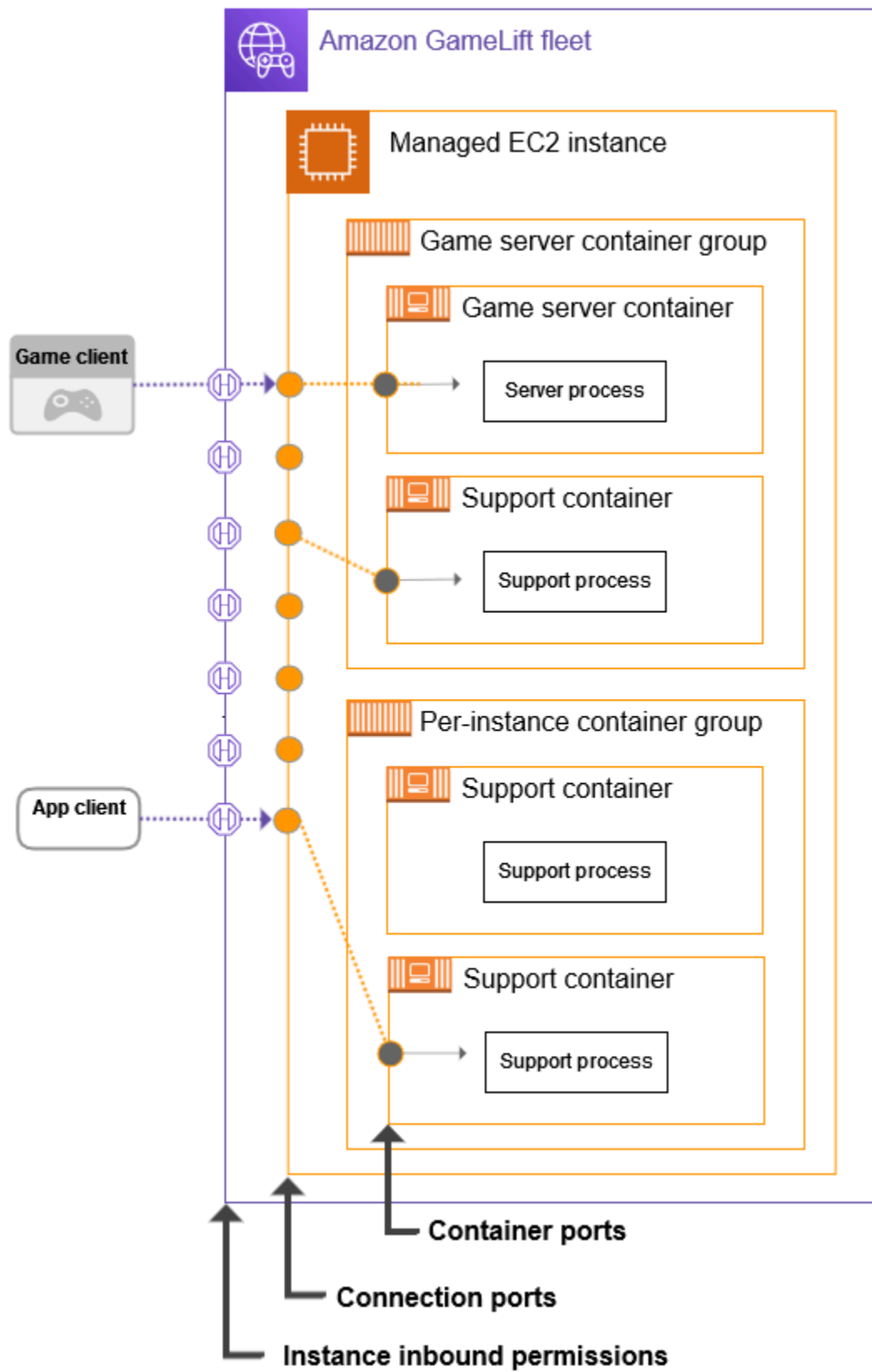
- 게임 서버 컨테이너에서 동시에 실행되는 모든 게임 서버 프로세스입니다. 게임 세션에 참여하려면 모든 게임 서버 프로세스에서 게임 클라이언트가 연결되도록 허용해야 합니다.

- 외부 소스가 연결해야 하는 지원 컨테이너의 모든 프로세스입니다. 예를 들어 테스트 앱에 원격으로 연결할 수 있습니다.

내부용 컨테이너 포트 설정을 설정하면는 이를 Amazon GameLift Servers 사용하여 게임 클라이언트와 다른 애플리케이션이 연결할 수 있는 외부용 인바운드 권한을 계산합니다. Amazon GameLift Servers 또한는 인바운드 권한과 플레이어에게 컨테이너의 게임 세션에 대한 액세스 권한을 부여하는 개별 컨테이너 포트 간의 매핑을 관리합니다. 이 내부 매핑은 컨테이너 포트에 대한 직접 액세스로부터 게임 서버를 보호하여 보안 계층을 제공합니다. 필요에 따라 플릿의 외부 포트 설정을 사용자 지정할 수 있는 옵션이 있습니다. 컨테이너 플릿 포트를 수동으로 설정하는 방법에 대한 자세한 내용은 [네트워크 연결 구성](#) 섹션을 참조하세요.

컨테이너 플릿의 포트 설정은 언제든지 수정할 수 있습니다. 이 변경에는 플릿 업데이트 배포가 필요합니다.

다음 다이어그램은 컨테이너 플릿에서 포트 연결의 역할을 보여줍니다. 그림과 같이 개별 컨테이너에 포트를 설정하고 Amazon GameLift Servers가 이 정보를 사용하여 플릿 인스턴스에서 각 컨테이너 포트에 매핑할 수 있는 충분한 포트를 구성합니다. 외부 인스턴스 인바운드 권한과 내부 연결 포트는 모두 수동으로 설정하지 않는 한 플릿용 Amazon GameLift Servers가 계산합니다.



컨테이너 로깅

관리형 컨테이너 플릿에서는 모든 컨테이너에 대해 표준 출력(및 표준 오류) 스트림이 캡처됩니다. 여기에는 게임 서버의 게임 세션 로그가 포함됩니다. 여러 옵션 중 하나를 사용하여 출력 스트림을 처리하도록 컨테이너 플릿을 구성할 수 있습니다.

- 컨테이너 출력을 Amazon CloudWatch 로그 스트림으로 저장합니다. 각 로그 스트림은 플릿 ID와 컨테이너를 참조합니다. 플릿에 대해 이 로깅 옵션을 선택하면 플릿의 모든 로그 스트림을 구성하는 CloudWatch 로그 그룹을 지정합니다. 그런 다음 CloudWatch 기능을 사용하여 필요에 따라 로그 데이터를 검색하고 분석할 수 있습니다.
- 컨테이너 출력을 Amazon Simple Storage Service(Amazon S3) 스토리지 버킷에 저장합니다. 필요에 따라 콘텐츠를 보거나 공유하거나 다운로드할 수 있습니다.
- 로깅을 해제합니다. 이 시나리오에서는 컨테이너 출력이 저장되지 않습니다.

Amazon GameLift Servers는 관리형 컨테이너 플릿의 로그 데이터를 AWS 계정의 CloudWatch 또는 Amazon S3 서비스로 전송합니다. 데이터를 보려면 AWS 계정에 로그인하고 개별 서비스를 사용하여 AWS Management Console 또는 기타 도구를 사용합니다. 컨테이너 플릿에 대한 서비스 역할을 생성하여 이러한 작업을 수행하도록 Amazon GameLift Servers에 대한 제한된 액세스를 확장합니다.

컨테이너 플릿의 로깅 구성은 언제든지 수정할 수 있습니다. 이 변경에는 플릿 업데이트 배포가 필요합니다.

컨테이너 플릿 및 Amazon GameLift Servers 에이전트

일반적으로 사용되는 컨테이너 아키텍처는 각 컨테이너에서 단일 프로세스를 실행합니다. Amazon GameLift Servers 컨테이너 플릿에서 게임 서버 컨테이너 그룹에는 하나의 게임 서버 프로세스를 실행하는 하나의 게임 서버 컨테이너가 있습니다. 이 아키텍처를 통해 Amazon GameLift Servers는 플릿 인스턴스의 각 게임 서버 컨테이너 그룹에서 단일 게임 서버 프로세스의 수명 주기를 관리합니다.

각 게임 서버 컨테이너 그룹에서 여러 게임 서버 프로세스를 실행하는 컨테이너 아키텍처를 빌드하려면 모든 프로세스의 수명 주기를 관리할 방법이 필요합니다. 여기에는 필요에 따라 프로세스 시작, 종료 및 교체, 원하는 수의 프로세스 동시 실행 관리, 실패 상태 처리와 같은 작업이 포함됩니다.

이러한 작업에 Amazon GameLift Servers 에이전트를 사용하도록 선택할 수 있습니다. 컨테이너 플릿의 경우 에이전트는 실행할 실행 파일(및 수)을 지정하고, 시작 파라미터를 제공하며, 게임 서버 활성화에 대한 규칙을 설정하는 런타임 지침을 구현합니다. 예를 들어, 런타임 지침은 에이전트에게 프로덕션용 게임 서버 프로세스 10개와 테스트용 특수 시작 파라미터가 있는 게임 서버 프로세스 1개를 유지하도록 지시할 수 있습니다.

컨테이너 플릿과 함께 에이전트를 사용하려면 컨테이너 이미지에 에이전트를 추가하고 런타임 지침 세트를 포함합니다. 에이전트에 대한 자세한 내용은 섹션을 참조하세요 [Amazon GameLift Servers에 이진트 사용법](#).

Amazon GameLift Servers 관리형 컨테이너 플릿 생성

Amazon GameLift Servers 관리형 컨테이너 플릿을 생성하여 AWS 클라우드에서 컨테이너화된 게임 서버를 배포하고 호스팅합니다. 컨테이너 플릿을 생성할 때 하나 이상의 컨테이너 이미지(게임 서버 빌드가 포함된 하나 이상) 및 구성 설정을 지정하는 컨테이너 그룹 정의를 지정합니다.

새 관리형 컨테이너 플릿 리소스를 생성하면 플릿 생성의 첫 번째 단계가 즉시 시작됩니다. 관리형 플릿 생성은 Amazon GameLift Servers가 EC2 인스턴스를 프로비저닝하고, 런타임 환경을 설치하며, 컨테이너 그룹을 인스턴스에 배포하고, 게임 서버 프로세스를 시작하면서 여러 단계를 거칩니다. 게임 서버 빌드에 필요한 런타임 환경에 따라 Amazon GameLift Servers는 플릿 생성 시 최신 버전의 Amazon Machine Image(AMI)를 배포합니다(플릿의 향후 모든 인스턴스는 동일한 버전을 사용함). 콘솔에서 또는 AWS Command Line Interface ()를 사용하여 플릿의 상태를 모니터링할 수 있습니다AWS CLI. 플릿이 ACTIVE 상태로 변경되면 게임 세션을 호스팅할 준비가 된 것입니다. 플릿 생성 문제에 도움을 받으려면 [Amazon GameLift Servers 플릿 문제 디버깅](#) 섹션을 참조하세요.

빈 컨테이너 플릿을 생성한 다음 나중에 플릿의 컨테이너 그룹 정의를 추가하거나 업데이트하도록 선택할 수 있습니다. 컨테이너 그룹 정의 없이 플릿을 생성하면 플릿이 활성 상태에 도달하지 않습니다.

Note

모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 이를 위해서는 새 플릿을 생성하고 플레이어 트래픽을 해당 플릿으로 마이그레이션해야 합니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#) 단원을 참조하십시오.

[Amazon GameLift Servers 콘솔](#) 또는 AWS Command Line Interface (AWS CLI)을 사용하여 컨테이너 플릿을 만들 수 있습니다.

Console

[Amazon GameLift Servers 콘솔](#)에서 플릿을 생성할 AWS 리전 를 선택합니다. 컨테이너 그룹 정의는 플릿을 생성하려는 리전과 동일한 리전에 있어야 합니다.

콘솔의 왼쪽 탐색 모음을 열고 관리형 컨테이너: 플릿을 선택합니다. 플릿 페이지에서 컨테이너 플릿 생성을 선택합니다.

1단계: 관리형 컨테이너 플릿 세부 정보 정의

1. 컨테이너 플릿 세부 정보 섹션에 플릿 설명을 입력합니다.
2. 플릿용 IAM 역할을 지정합니다. 이 역할에는 Amazon GameLift Servers가 사용자를 대신하여 컨테이너 플릿을 관리해야 하는 권한이 있습니다. 필요한 서비스 역할을 생성하는 데 도움이 필요하다면 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#) 섹션을 참조하세요.
3. 로그 구성 옵션을 선택합니다. 기본적으로 CloudWatch 옵션이 설정되어 있습니다. 선택한 옵션에 따라 필요한 정보를 제공합니다.
4. 플릿에 컨테이너 그룹을 추가합니다. 이 단계는 선택 사항입니다. 나중에 추가할 계획으로 컨테이너 그룹 없이 플릿을 생성하도록 선택할 수 있습니다. 컨테이너 그룹이 없는 플릿은 플릿 인스턴스를 배포하거나 게임을 호스팅할 수 없지만 플릿 리소스는 생성됩니다.
 - 게임 서버 컨테이너 그룹 정의를 선택합니다. 선택적으로 배포하려는 정의의 버전을 지정합니다. 버전 번호를 지정하지 않을 경우 Amazon GameLift Servers가 자동으로 최신 버전을 사용합니다.
 - 선택적으로 인스턴스당 컨테이너 그룹 정의 및 버전을 추가합니다. 버전 번호를 지정하지 않을 경우 Amazon GameLift Servers가 자동으로 최신 버전을 사용합니다.
5. 추가 세부 정보에서 몇 가지 사용자 지정을 선택적으로 설정할 수 있습니다. 컨테이너 플릿을 생성하는 데 이러한 설정이 필요하지 않습니다.

2단계: 인스턴스 세부 정보 정의

1. 인스턴스 배포에서 인스턴스를 배포할 원격 위치를 하나 이상 선택합니다. 홈 리전이 자동으로 선택됩니다(플릿을 생성하는 리전). 위치를 추가로 선택하면 플릿 인스턴스도 해당 위치에 배포됩니다.

Important

기본적으로 활성화되지 않은 리전을 사용하려면에서 활성화합니다 AWS 계정.

- 2022년 2월 28일 이전에 생성한 리전 중 활성화되지 않은 리전의 플릿은 이 요구 사항의 영향을 받지 않습니다.
- 새 다중 위치 플릿을 생성하거나 기존의 다중 위치 플릿을 업데이트하려면 먼저 사용하려는 리전 또는 로컬 영역을 모두 활성화해야 합니다.

기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 [AWS 리전관리](#)를 참조하세요. AWS 로컬 영역 사용자 가이드의 [로컬 영역 시작하기](#)를 참조하세요.

2. 플릿에 대한 인스턴스 구성을 선택합니다. 콘솔은 필요한 최소 vCPU 및 메모리를 자동으로 계산합니다(각 컨테이너 그룹에 대해 설정한 총 한도 기준). 리소스 요구 사항과 입력한 위치를 기반으로 사용 가능한 인스턴스 유형의 전체 목록을 필터링합니다. 필요에 따라 필터를 추가할 수 있습니다.

인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 [컨테이너 플릿 구성](#) 섹션을 참조하세요. 선택한 인스턴스 유형의 크기는 게임 서버 컨테이너 그룹이 각 플릿 인스턴스에 패키징되는 방식에 영향을 미칩니다. 선택에 따라 인스턴스당 원하는 게임 서버 컨테이너 그룹에 대한 설정을 검토하는 것이 좋습니다.

4단계: 검토 및 생성

- 플릿 구성 설정을 검토합니다.

플릿 상태와 상관없이 플릿의 메타데이터 및 구성을 언제든지 업데이트할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 플릿 구성 업데이트](#) 섹션을 참조하세요. 플릿이 활성 상태가 된 후에 플릿 용량을 업데이트할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#) 섹션을 참조하세요. 원격 위치를 추가하거나 제거할 수도 있습니다.

검토를 마쳤으면 생성을 선택합니다.

요청이 성공하면 콘솔에 새 플릿 리소스에 대한 세부 정보 페이지가 표시됩니다. Amazon GameLift Servers가 플릿 생성 프로세스를 시작하므로 처음에 상태는 NEW입니다. 플릿 페이지에서 새 플릿의 상태를 볼 수 있습니다. 플릿이 ACTIVE 상태가 되면 게임 세션을 호스팅할 준비가 된 것입니다.

AWS CLI

를 사용하여 컨테이너 플릿을 생성하려면 명령줄 창을 AWS CLI 열고 `create-container-fleet` 명령을 사용합니다. 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 [create-container-fleet](#) 섹션을 참조하세요.

아래 표시된 예제 `create-container-fleet` 요청은 다음과 같은 특징을 가진 새로운 컨테이너 플릿을 만듭니다.

- ContainerGroupsConfiguration은 다음 게임 서버 컨테이너 그룹 정의만 지정합니다.
MyAdventureGameContainerGroup. Amazon GameLift Servers가 각 플릿 인스턴스에 배포될 게임 서버 컨테이너 그룹의 수를 계산합니다.
- 플릿은 c5.large 온디맨드 인스턴스를 기본적으로 사용합니다.
- 기본적으로 플릿은 Amazon GameLift Servers에서 계산한 연결 포트 및 인바운드 권한 포트 세트를 엽니다. 컨테이너 그룹을 다음 위치에 배포합니다.

```
aws gamelift create-container-fleet \  
  --fleet-role-arn arn:aws:iam::MyAccount:role/MyContainersRole \  
  --game-server-container-group-definition-name "rn:aws:gamelift:us-  
west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2" \  
  --game-server-container-group-definition-arn arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2
```

`create-fleet` 요청에 성공하면 Amazon GameLift Servers는 사용자가 요청한 구성 설정과 새 컨테이너 플릿 ID를 포함하는 플릿 속성 세트를 반환합니다. Amazon GameLift Servers는 즉시 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-container-fleet](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

이러한 명령을 사용하여 필요에 따라 플릿의 용량 및 기타 구성 설정을 변경할 수 있습니다.

- [update-container-fleet](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Amazon GameLift Servers 컨테이너 플릿에 대한 컨테이너 그룹 정의 생성

컨테이너 그룹 정의는 컨테이너식 게임 서버 애플리케이션을 컨테이너 플릿에 배포하는 방법을 설명합니다. Amazon GameLift Servers에게 플릿에 배포할 컨테이너 이미지와 이를 실행하는 방법을 알려주는 블루프린트입니다. 컨테이너 플릿을 생성할 때 플릿에 배포할 컨테이너 그룹 정의를 지정합니다. 컨테이너 그룹에 대한 자세한 내용은 [컨테이너 플릿 구성 요소](#) 섹션을 참조하세요.

시작하기 전에

컨테이너 그룹 정의 생성을 시작하기 전에 수행할 작업에 대한 팁:

- 컨테이너 이미지를 완성하고 컨테이너 그룹을 생성하려는 AWS 리전 동일한의 Amazon Elastic Container Registry(Amazon ECR) 리포지토리로 푸시합니다. 컨테이너 그룹 정의를 생성할 때 각 이미지의 스냅샷을 Amazon GameLift Servers 캡처하고 컨테이너 플릿에 배포할 때 스냅샷을 사용합니다. [Amazon GameLift Servers용 컨테이너 이미지 빌드](#)을(를) 참조하세요.
- 연결 정의를 JSON 파일로 생성합니다. 컨테이너 그룹 정의에는 하나 이상의 컨테이너 정의를 포함합니다. AWS CLI를 사용하여 컨테이너 그룹 정의를 생성하는 경우 JSON 파일을 사용할 수 있습니다 Amazon GameLift Servers.
- AWS 사용자에게 Amazon ECR 리포지토리에 액세스할 수 있는 IAM 권한이 있는지 확인합니다. [Amazon GameLift Servers의 IAM 권한 예제](#)을(를) 참조하세요.

게임 서버 컨테이너 그룹 정의 생성

게임 서버 컨테이너 그룹은 게임 서버 소프트웨어를 실행합니다. 게임 서버 컨테이너 그룹에는 게임 서버 컨테이너를 실행하는 하나의 게임 서버 컨테이너가 있습니다. 게임 서버를 지원하는 추가 소프트웨어를 실행하기 위한 지원 컨테이너가 하나 이상 있을 수도 있습니다. (이를 '사이드카' 컨테이너라고도 합니다.)

이 주제에서는 Amazon GameLift Servers 콘솔 또는 AWS CLI 도구를 사용하여 간단한 게임 서버 컨테이너 그룹 정의를 생성하는 방법을 설명합니다. 선택적 기능에 대한 자세한 내용은 [Amazon GameLift Servers 컨테이너 플릿 사용자 지정](#) 섹션을 참조하세요.

Note

대부분의 컨테이너 그룹 정의 및 컨테이너 정의 설정은 생성한 후 변경할 수 있습니다. 컨테이너 정의를 변경하면 Amazon GameLift Servers가 업데이트된 컨테이너 이미지의 새 스냅샷을 캡처합니다.

간단한 게임 서버 컨테이너 그룹 정의를 생성하려면:

다음 지침에서는 최소 필수 파라미터를 사용하고 Amazon GameLift Servers 기본값을 사용하여 컨테이너 그룹 정의를 생성하는 방법을 설명합니다.

Console

[Amazon GameLift Servers 콘솔](#)에서 컨테이너 그룹을 생성할 AWS 리전 를 선택합니다.

콘솔의 왼쪽 탐색 모음을 열고 관리형 컨테이너: 그룹 정의를 선택합니다. 컨테이너 그룹 정의 페이지에서 그룹 정의 생성을 선택합니다.

1단계: 컨테이너 그룹 정의 세부 정보 정의

1. 컨테이너 그룹 정의 이름을 입력합니다. 이름은 AWS 계정 및 리전에서 고유해야 합니다.
2. 게임 서버 컨테이너 그룹 유형을 선택합니다.
3. 총 메모리 제한에 컨테이너 그룹의 모든 컨테이너에 사용할 수 있는 최대 메모리 리소스를 입력합니다. 이 값을 계산하는 데 도움이 필요하면 [리소스 제한 설정](#) 섹션을 참조하세요.
4. 총 vCPU 제한에 컨테이너 그룹의 모든 컨테이너에 사용할 수 있는 최대 컴퓨팅 성능을 입력합니다. 이 값을 계산하는 데 도움이 필요하면 [리소스 제한 설정](#) 섹션을 참조하세요.

2단계: 컨테이너 정의 추가

게임 서버 컨테이너 그룹에는 최소한 하나의 게임 서버 컨테이너가 있습니다. 콘솔에서 생성하는 첫 번째 컨테이너 정의는 게임 서버 컨테이너입니다. 이 단계에서는 게임 서버 컨테이너 정의에 필요한 최소 설정을 정의하는 방법을 설명합니다.

1. 컨테이너 정의 이름을 입력합니다. 그룹에 대해 정의된 각 컨테이너에는 고유한 이름 값이 있어야 합니다.
2. 게임 서버 빌드를 사용하여 컨테이너 이미지에 연결합니다. 퍼블릭 또는 프라이빗 리포지토리의 컨테이너 이미지에 대한 Amazon ECR 이미지 URI를 입력합니다. 다음 형식을 사용할 수 있습니다.
 - 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. 게임 서버 빌드에서 사용하는 Amazon GameLift Servers 서버 SDK 버전을 지정합니다. 컨테이너 플릿의 경우 이 값은 5.2.0 이상이어야 합니다.
 4. 내부 컨테이너 포트 범위에서 프로토콜을 설정하고 포트 범위를 정의합니다. 범위 크기는 이 컨테이너에서 실행될 동시 게임 서버 프로세스 수보다 커야 합니다. 게임 서버 컨테이너가 컨테이너당 하나의 서버 프로세스만 실행하는 경우 이 포트 범위에는 몇 개의 포트만 필요합니다. 자세한 내용은 [네트워크 연결 구성](#) 섹션을 참조하세요.
 5. 필요에 따라 컨테이너를 더 추가하여 추가 지원 소프트웨어를 실행합니다. 추가 컨테이너는 자동으로 지원 컨테이너로 지정됩니다. 게임 서버 컨테이너 그룹에는 하나의 게임 서버 컨테이너와 최대 8개의 지원 컨테이너만 있을 수 있습니다. 다음 필수 설정은 반드시 제공해야 합니다.
 - 컨테이너 정의 이름
 - ECR 이미지 URI.
 - 내부 컨테이너 포트(컨테이너에 네트워크 액세스가 필요한 프로세스가 있는 경우에만 포함)

3단계: 종속성 구성

- 컨테이너 그룹 정의에 둘 이상의 컨테이너가 있는 경우 필요에 따라 컨테이너 간에 종속성을 설정할 수 있습니다. 자세한 내용은 [컨테이너 종속성 설정](#) 단원을 참조하십시오.

3단계: 검토 및 생성

1. 컨테이너 그룹 정의 설정을 모두 검토합니다. 편집을 사용하여 그룹에 대한 각 컨테이너 정의를 포함하여 모든 섹션을 변경합니다.
2. 검토를 마쳤으면 생성을 선택합니다.

요청이 성공하면 콘솔에 새 컨테이너 그룹 정의 리소스에 대한 세부 정보 페이지가 표시됩니다. Amazon GameLift Servers가 그룹에 대한 모든 컨테이너 이미지의 스냅샷을 생성하기 시작하므로 처음에 상태는 COPYING입니다. 이 단계가 완료되면 컨테이너 그룹 정의 상태가 READY로 변경됩니다. 컨테이너 그룹 정의가 READY 상태여야 컨테이너 플릿을 생성할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 컨테이너 그룹 정의를 생성할 때 컨테이너 정의 구성을 별도의 JSON 파일로 유지 관리합니다. CLI 명령에서 파일을 참조할 수 있습니다. 스키마 예제는 [컨테이너 정의 JSON 파일 생성](#) 섹션을 참조하세요.

컨테이너 그룹 정의 생성

새 컨테이너 그룹 정의를 생성하려면 `create-container-group-definition` CLI 명령을 사용합니다. 이 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 [create-container-group-definition](#)을 참조하세요.

이 예제에서는 게임 서버 컨테이너 그룹 정의에 대한 요청을 보여줍니다. 이 그룹에 대한 컨테이너 정의가 포함된 JSON 파일을 생성했다고 가정합니다.

```
aws gamelift create-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --game-server-container-definition file://MyAdventureGameContainers.json
```

컨테이너 정의 JSON 파일 생성

컨테이너 그룹 정의를 생성할 때 그룹의 컨테이너도 정의합니다. 컨테이너 정의는 컨테이너 이미지가 저장되는 Amazon ECR 리포지토리와 네트워크 포트에 대한 선택적 구성, CPU 및 메모리 사용량 제한 및 기타 설정을 지정합니다. 컨테이너 그룹의 모든 컨테이너에 대한 구성으로 단일 JSON 파일을 생성하는 것이 좋습니다. 파일을 유지 관리하면 이러한 중요한 구성을 저장, 공유, 버전 추적하는 데 유용합니다. AWS CLI를 사용하여 컨테이너 그룹 정의를 생성하는 경우 명령에서 파일을 참조할 수 있습니다.

컨테이너 정의를 생성하려면

1. 새 .JSON 파일을 생성하고 엽니다. 예제:

```
[~/work/glc]$ vim SimpleServer.json
```

2. 그룹의 각 컨테이너에 대해 별도의 컨테이너 정의를 생성합니다. 다음 예제 콘텐츠를 복사하여 컨테이너에 맞게 수정합니다. 컨테이너 정의의 구문에 대한 자세한 내용은 Amazon GameLift Servers API Reference의 [ContainerDefinitionInput](#)을 참조하세요.

3. AWS CLI 명령에서 참조할 수 있도록 파일을 로컬에 저장합니다.

예: 게임 서버 컨테이너 정의

Example

이 예제에서는 게임 서버 컨테이너 그룹의 필수 컨테이너를 설명합니다. 필수 복제본 컨테이너에는 게임 서버 애플리케이션, Amazon GameLift Servers 에이전트가 포함되며 게임 호스팅을 지원하는 다른 소프트웨어가 포함될 수 있습니다. 정의에는 이름, 이미지 URI 및 포트 구성이 포함되어야 합니다. 이 예제에서는 일부 컨테이너별 리소스 제한도 설정합니다.

```
{
  "ContainerName": "MyAdventureGameServer",
  "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-
containers:myadventuregame-server",
  "PortConfiguration": {
    "ContainerPortRanges": [
      {
        "FromPort": 2000,
        "Protocol": "TCP",
        "ToPort": 2010
      }
    ]
  },
  "ServerSdkVersion": "5.2.0"
}
```

Amazon GameLift Servers Anywhere 플릿

자체 호스팅 리소스와 함께 Amazon GameLift Servers 기능을 활용하려면 Anywhere 플릿을 사용합니다. Anywhere 플릿은 반복적인 개발을 위한 테스트 환경으로, 또는 하이브리드 호스팅 솔루션의 관리형 플릿과 함께 일반적으로 사용됩니다.

Anywhere 플릿은 사용자가 제공하고 관리하는 컴퓨팅 리소스(가상 또는 물리적) 세트로 구성됩니다. 컴퓨팅은 로컬 노트북부터 원격 아웃포스트까지 연결이 가능한 모든 지리적 위치에 상주할 수 있습니다. Anywhere 플릿을 설정할 때 Amazon GameLift Servers를 통해 플릿을 등록하여 플릿에 컴퓨팅을 추가합니다. 각 컴퓨팅은 IP 주소(또는 DNS 이름)로 등록하여 Amazon GameLift Servers와의 연결을 설정할 수 있습니다.

각 컴퓨팅에 게임 서버 소프트웨어를 설치하고 게임 서버 프로세스를 시작하여 Anywhere 플릿에 게임 서버 소프트웨어를 배포합니다. 실행된 각 게임 서버 프로세스는 Amazon GameLift Servers 서비스에

대한 연결을 설정하고 게임 세션을 호스팅할 준비를 보고합니다. 기존의 설정 관리 및 배포 도구를 사용하여 초기 배포 및 호스트 관리 작업을 처리할 수 있습니다. Amazon GameLift Servers와 함께 사용하는 데 필요한 몇 가지 추가 작업은 다음과 같습니다.

- 컴퓨팅을 등록 및 등록 취소하여 플릿에서 추가하거나 제거합니다.
- 모든 컴퓨팅에서 최신 인증 토큰을 유지합니다. 컴퓨팅의 서버 프로세스가 Amazon GameLift Servers 서비스에 연결할 때 사용합니다.

Note

선택적으로 Amazon GameLift Servers 에이전트와 함께 Anywhere 플릿을 배포하여 이러한 키 관리 작업을 자동화할 수 있습니다. [Amazon GameLift Servers 에이전트 사용법](#)(를) 참조하세요.

Anywhere 플릿을 설정하고 유지하는 방법에 대한 다음 주제를 참조하세요.

- [Amazon GameLift Servers Anywhere 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers를 사용한 하이브리드 호스팅을 위한 개발 로드맵](#)
- [Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정](#)
- [Amazon GameLift Servers Anywhere 플릿 생성](#)
- [Amazon GameLift Servers 플릿 구성 업데이트](#)

Anywhere 플릿 생성 워크플로:

Amazon GameLift Servers는 Anywhere 플릿에 대해 플릿 리소스만 설정합니다. 플릿에 컴퓨팅을 설정 및 등록하고 게임 서버 소프트웨어를 설치한 후 게임 서버 프로세스를 시작하여 게임 세션을 호스팅합니다.

1. Amazon GameLift Servers가 플릿의 홈 리전에 플릿 리소스를 생성합니다. 플릿 상태 및 사용자 지정 위치 상태는 신규로 설정됩니다.
2. Amazon GameLift Servers는 플릿 이벤트 로그에 이벤트를 기록하기 시작합니다.
3. 플릿 리소스가 생성된 후 Amazon GameLift Servers는 플릿 상태를 활성으로 설정합니다. 이 시점에서 플릿으로 신규 컴퓨팅을 등록할 수 있습니다.

Amazon GameLift Servers Anywhere 플릿 생성

이 주제에서는 Amazon GameLift Servers Anywhere 플릿을 생성하는 방법을 설명합니다. Anywhere 플릿을 사용하면 Amazon GameLift Servers 게임 세션 관리 주요 기능을 사용하는 동시에 자체 컴퓨팅 리소스로 게임 세션을 호스팅할 수 있습니다. 온프레미스 하드웨어 또는 기타 클라우드 기반 리소스를 위한 Anywhere 플릿을 생성합니다.

일반적으로 모든 플릿은 하이브리드 호스팅 솔루션에서 Amazon GameLift Servers 관리형 플릿과 함께 사용됩니다. 또한 Amazon GameLift Servers로 호스팅하기 위한 게임을 개발할 때 유용한 테스트 환경을 제공합니다. Amazon GameLift Servers Anywhere 플릿을 게임 호스팅 솔루션에 통합하는 시기와 방법에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- [하이브리드 호스팅](#)
- [Amazon GameLift Servers용 호스팅 플릿 배포](#)
- [Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정](#)

Anywhere 플릿은 자체 관리형 플릿으로 설정 시 몇 가지 추가 작업이 필요합니다. Anywhere 플릿이 게임 세션과 플레이어를 호스팅할 수 있도록 준비하려면 다음 작업을 완료해야 합니다.

주제

- [시작하기 전에](#)
- [사용자 지정 위치 생성](#)
- [Anywhere 플릿 생성](#)
- [플릿에 컴퓨팅 추가](#)
- [게임 서버 실행 시작](#)

시작하기 전에

Anywhere 플릿을 생성하기 전에 다음 작업을 수행합니다. 자세한 지침은 [Amazon GameLift Servers Anywhere 호스팅을 위한 개발 로드맵](#) 또는 [Amazon GameLift Servers를 사용한 하이브리드 호스팅을 위한 개발 로드맵](#) 섹션을 참조하세요.

- 게임 서버 코드를 Amazon GameLift Servers 서버 SDK 버전 5.x(또는 그 이상)와 통합합니다. 모든 게임 통합 작업을 완료할 필요는 없으며 게임 서버 빌드에 필요한 작업만 완료하면 됩니다. 일반적인 방법은 로컬 시스템을 Anywhere 플릿으로 설정하고 명령줄 인터페이스를 사용하여 게임 서버 통합을 테스트하는 것입니다([Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#) 참조).

추가 구성 요소(예: Amazon GameLift Servers가 활성화된 게임 클라이언트)를 개발 시 통합할 수 있습니다.

- Anywhere 플릿 컴퓨팅에 설치할 게임 서버 소프트웨어를 패키징합니다. 패키지에는 통합 게임 서버 빌드와 게임 서버를 실행하는 데 필요한 모든 지원 소프트웨어가 포함되어야 합니다.
- Amazon GameLift Servers 에이전트를 Anywhere 플릿과 함께 사용할지 여부를 결정합니다. 에이전트는 Amazon GameLift Servers에 사용할 서버 프로세스 및 컴퓨팅 관리와 관련된 일부 주요 작업을 자동화하는 온컴퓨트 프로세스 관리 도구입니다. 자세한 내용은 [Amazon GameLift Servers 에이전트 사용법](#) 섹션을 참조하세요.

사용자 지정 위치 생성

사용자 지정 위치를 생성하여 컴퓨팅 리소스의 물리적 위치를 나타냅니다. Anywhere 플릿을 생성할 때는 사용자 지정 위치가 하나 이상 이미 정의되어 있어야 합니다. 언제든지 추가 사용자 지정 위치를 생성하고 기존 플릿에 추가할 수 있습니다.

사용자 지정 위치 생성

Amazon GameLift Servers 콘솔 또는 AWS Command Line Interface(AWS CLI)을 사용하여 사용자 지정 위치를 생성합니다.

Console

[Amazon GameLift Servers 콘솔](#)의 탐색 창에서 위치 페이지를 엽니다. 위치 생성을 선택하여 테이프 생성 대화 상자를 엽니다.

1. 대화 상자에 위치 이름을 입력합니다. 식별할 수 있게 위치를 설명하는 이름을 컴퓨팅 리소스 세트에 사용하는 것이 좋습니다. 지리적 위치, 데이터 센터 이름 또는 기타 위치 식별자일 수 있습니다. Amazon GameLift Servers는 사용자 지정 위치의 이름에 custom-를 추가합니다.
2. (선택)태그를 사용자 지정 위치에 추가합니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 용도, 소유자, 환경 등의 기준에 따라 태그를 지정해 AWS 리소스를 유용하게 분류할 수 있습니다. 추가하려는 각 추가 태그에 대해 새 태그 추가를 선택합니다.
3. 생성(Create)을 선택합니다.

AWS CLI

[create-location](#) 명령을 사용하여 사용자 지정 위치를 생성합니다. custom-으로 시작하는 location-name 값을 제공해야 합니다. 식별할 수 있게 위치를 설명하는 이름을 컴퓨팅 리소스 세

트에 사용하는 것이 좋습니다. 지리적 위치, 데이터 센터 이름 또는 기타 위치 식별자일 수 있습니다.

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

Output

```
{  
  "Location": {  
    "LocationName": "custom-location-1",  
    "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-  
location-1"  
  }  
}
```

Anywhere 플릿 생성

보유한 컴퓨팅 리소스 집합에 대한 Anywhere 플릿을 생성합니다. 새 Anywhere 플릿은 비어 있으며 컴퓨팅을 등록하여 플릿에 컴퓨팅을 추가합니다.

생성 과정에서 신규 Anywhere 플릿이 NEW에서 ACTIVE 상태로 빠르게 전환됩니다. 플릿이 ACTIVE 상태로 전환되면 플릿에 컴퓨팅을 추가할 수 있습니다.

Anywhere 플릿 생성

Amazon GameLift Servers 콘솔 또는 AWS Command Line Interface(AWS CLI)을 사용하여 Anywhere 플릿을 생성합니다.

Console

[Amazon GameLift Servers 콘솔](#)의 탐색 창에서 플릿 페이지를 실행합니다. 플릿 생성을 선택하여 플릿 생성 워크플로를 시작합니다.

1단계: 컴퓨팅 유형 선택

Anywhere 옵션을 선택하고 다음을 선택합니다.

2단계: 플릿 세부 정보 지정

이 단계에서는 플릿의 전반적인 주요 설정을 지정합니다.

1. 플릿 세부 정보 섹션을 작성합니다.
 - a. 플릿 이름을 입력합니다. 플릿 목록을 볼 때 플릿 유형을 더 쉽게 식별할 수 있도록 플릿 이름 지정 패턴을 사용하는 것이 좋습니다.
 - b. 플릿에 대한 간략한 설명을 입력합니다.
2. 필요에 따라 선택적으로 추가 세부 정보를 설정합니다. 이러한 플릿 설정은 필요에 따라 나중에 업데이트할 수 있습니다.
 - a. 프로덕션 또는 사전 프로덕션 테스트를 위해 플릿을 생성할 때 이 설정을 사용하여 플릿 컴퓨팅의 시간당 비용 값을 지정합니다. Amazon GameLift Servers는 게임 세션 배치 프로세스 중에 이 정보를 사용하여 비용을 기반으로 호스팅 리소스를 선택할 수 있습니다.
 - b. 해당 플릿 및 기타 플릿의 지표 데이터를 결합하려면 지표 그룹 이름을 지정합니다. 함께 결합하려는 모든 플릿에 동일한 지표 그룹 이름을 사용합니다. 지표 그룹의 지표를 보고 집계된 데이터를 확인합니다.
3. 필요한 태그를 사용자 지정 위치에 추가합니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 용도, 소유자, 환경 등의 기준에 따라 태그를 지정해 AWS 리소스를 유용하게 분류할 수 있습니다. 추가하려는 각 추가 태그에 대해 새 태그 추가를 선택합니다.
4. 워크플로를 계속하려면 다음을 선택합니다.

3단계: 사용자 지정 위치 선택

이 단계에서는 이 플릿에 추가할 컴퓨팅의 물리적 위치를 식별합니다. 지금 하나 이상의 위치를 지정하고 필요에 따라 나중에 위치를 추가하거나 제거할 수 있습니다.

1. 사용자 지정 위치에서 플릿의 컴퓨팅에 대해 하나 이상의 위치를 선택합니다. 목록에는 현재 선택한 AWS 리전에서 정의한 모든 사용자 지정 위치가 포함됩니다. 플릿에 추가할 새 사용자 지정 위치를 정의하려면 위치 생성을 선택합니다.
2. 워크플로를 계속하려면 다음을 선택합니다.

4단계: 검토 및 생성

플릿을 생성하기 전에 설정을 검토합니다.

새 플릿을 배포할 준비가 되면 생성을 선택합니다. Amazon GameLift Servers가 플릿 활성화 프로세스를 즉시 시작하여 고유한 ID를 할당하고 플릿을 NEW 상태로 전환합니다. 플릿 페이지에서 플릿의 상태를 볼 수 있습니다.

AWS CLI

[create-fleet](#) 명령을 사용하여 컴퓨팅 유형 ANYWHERE의 플릿을 생성합니다. 이름과 하나 이상의 사용자 지정 위치를 제공합니다. Amazon GameLift Servers가 현재 기본 AWS 리전에 Anywhere 플릿 리소스를 생성합니다(또는 다른 AWS 리전에 리전 태그 추가 지정 가능).

다음 예제 요청으로 최소 필수 설정을 사용하는 새 플릿을 생성할 수 있습니다. *FleetName*과 *custom-location*를 사용자 정보로 바꿉니다.

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

응답의 예

```
{  
  "FleetAttributes": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "Name": "HardwareAnywhere",  
    "CreationTime": "2023-02-23T17:57:42.293000+00:00",  
    "Status": "ACTIVE",  
    "MetricGroups": [  
      "default"  
    ],  
    "CertificateConfiguration": {  
      "CertificateType": "DISABLED"  
    },  
    "ComputeType": "ANYWHERE"  
  }  
}
```

생성 과정에서 신규 Anywhere 플릿이 ACTIVE 상태로 빠르게 전환됩니다. 플릿이 ACTIVE 상태로 전환되면 플릿에 컴퓨팅을 추가할 수 있습니다.

응답에는 플릿 위치가 포함되지 않습니다. [describe-fleet-attributes](#) 및 [describe-fleet-location-attributes](#)를 호출하여 전체 플릿 세부 정보를 검색할 수 있습니다

플릿에 컴퓨팅 추가

플릿에 컴퓨팅 리소스를 추가하고 게임 세션을 호스팅할 준비를 하려면 다음 작업을 수행합니다.

- 플릿에 컴퓨팅을 등록합니다. 컴퓨팅을 등록해 Amazon GameLift Servers 내 플릿의 물리적 호스팅 리소스를 파악할 수 있습니다.
- 컴퓨팅에 대한 인증 토큰을 요청합니다. 컴퓨팅에서 실행되는 각 게임 서버는 Amazon GameLift Servers 서비스에 연결하기 위해 이 토큰이 필요합니다. 인증 토큰은 임시적이며 정기적으로 새로 고쳐야 합니다.

Note

Amazon GameLift Servers 에이전트를 사용하여 게임 서버 소프트웨어를 배포하는 경우 이 단계를 건너뛸 수 있습니다. 에이전트는 각 컴퓨팅을 자동으로 등록하고 컴퓨팅에 유효한 인증 토큰을 유지합니다. [Amazon GameLift Servers 에이전트 사용법](#)(를) 참조하세요.

AWS CLI를 사용하거나 Amazon GameLift Servers용 AWS SDK를 프로그래밍 방식으로 직접 호출하여 컴퓨팅을 등록하고 인증 토큰을 요청할 수 있습니다. Amazon GameLift Servers 콘솔에서는 이러한 작업을 사용할 수 없습니다.

각 컴퓨팅에 시작 스크립트를 추가하여 이러한 두 작업을 모두 자동화하는 것이 좋습니다. 시작 스크립트는 `register-compute` 및 `get-compute-auth-token` 명령을 모두 자동으로 호출합니다. 또한 작업을 자동화하여 컴퓨팅 수명 기간 동안 인증 토큰을 정기적으로 새로 고치고 종료 시 컴퓨팅 등록을 취소할 수 있습니다.

각 시작 작업은 컴퓨팅에 저장해야 하는 컴퓨팅별 값을 반환합니다. 컴퓨팅에서 게임 서버 프로세스가 시작되면 Amazon GameLift Servers 서비스와의 연결을 초기화할 때 이러한 값을 서버 파라미터로 전달해야 합니다(서버 SDK 참조의 [ServerParameters](#) 참조). 이러한 컴퓨팅별 값(또는 저장된 위치)을 환경 변수로 설정하는 것이 좋습니다. Amazon GameLift Servers 에이전트를 사용하는 경우 이 작업이 처리됩니다. 컴퓨팅별 값은 다음과 같습니다.

- `register-compute`가 `GameLiftServiceSdkEndpoint`에 대한 값을 반환합니다. 이 값을 `webSocketUrl` 서버 파라미터의 값으로 설정합니다.
- `compute-auth-token`이 인증 토큰을 반환합니다. 이 값을 `authToken` 서버 파라미터의 값으로 설정합니다.

AWS CLI

다음 지침은 AWS CLI를 사용하여 각 요청을 수동으로 제출하는 방법을 설명합니다.

컴퓨팅 등록

[register-compute](#)를 호출하여 컴퓨팅을 등록합니다. 컴퓨팅을 추가할 플릿의 ID를 식별합니다. 식별할 수 있는 이름, IP 주소 및 위치 등의 컴퓨팅 정보를 제공합니다. 컴퓨팅의 위치는 플릿과 이미 연결된 사용자 지정 위치여야 합니다. 다른 사용자 지정 위치를 사용하려면 Amazon GameLift Servers 콘솔을 사용하여 플릿을 업데이트하거나 AWS CLI 명령 [create-fleet-locations](#)을 호출하여 플릿에 사용자 지정 위치를 추가합니다.

다음 예에서는 컴퓨팅 및 플릿의 자리 표시자 값을 바꿉니다. Anywhere 플릿을 생성하면 `fleet-id` 값이 반환됩니다. [describe-fleet-attributes](#) 및 [describe-fleet-location-attributes](#)를 호출하여 전체 플릿 세부 정보를 검색할 수 있습니다.

```
aws gamelift register-compute \
  --compute-name HardwareAnywhere \
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --ip-address 10.1.2.3 \
  --location custom-location-1
```

출력 예시

```
{
  "Compute": {
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
    "ComputeName": "HardwareAnywhere",
    "ComputeArn": "arn:aws:gamelift:us-west-2:111122223333:compute/
HardwareAnywhere",
    "IpAddress": "10.1.2.3",
    "ComputeStatus": "Active",
    "Location": "custom-location-1",
    "CreationTime": "2023-02-23T18:09:26.727000+00:00",
    "GameLiftServiceSdkEndpoint": "wss://us-west-2.api.amazongamelift.com"
  }
}
```

인증 토큰 요청

[get-compute-auth-token](#)을 호출하여 유효한 인증 토큰을 요청하고 컴퓨팅을 등록합니다. 플릿 ID와 컴퓨팅 이름을 식별합니다.

다음 예에서는 컴퓨팅 및 플릿의 자리 표시자 값을 바꿉니다. Anywhere 플릿을 생성하면 fleet-id 값이 반환됩니다. [describe-fleet-attributes](#)를 호출하여 전체 플릿 세부 정보를 검색할 수 있습니다. 컴퓨팅 정보를 찾으려면 플릿 ID로 [list-compute](#)를 호출하여 플릿에 등록된 모든 컴퓨팅을 확인합니다.

```
aws gamelift get-compute-auth-token \
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --compute-name HardwareAnywhere
```

출력 예시

```
{
  "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "ComputeName": "HardwareAnywhere",
  "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/
HardwareAnywhere",
  "AuthToken": "0c728041-3e84-4aaa-b927-a0fb202684c0",
  "ExpirationTimestamp": "2023-02-23T18:47:54+00:00"
}
```

게임 서버 실행 시작

Anywhere 플릿을 생성하고 플릿에 하나 이상의 컴퓨팅을 추가하면 게임 서버 실행을 시작할 준비가 완료됩니다.

1단계: 게임 서버 소프트웨어 설치

Anywhere 플릿의 각 컴퓨팅에 게임 서버 빌드와 모든 종속 소프트웨어를 설치합니다. 게임 서버 빌드는 Amazon GameLift Servers 서버 SDK 버전 5.x(또는 그 이상)와 통합되어야 하며 Amazon GameLift Servers 서비스와 통신하는 데 필요한 최소 기능이 있어야 합니다.

2단계: 게임 서버를 실행할 수 있도록 컴퓨팅 준비

각 컴퓨팅이 등록되어 있고 유효한 인증 토큰이 있는지 확인합니다. 스크립트를 사용하여 이러한 작업을 관리하는 경우 게임 서버 프로세스를 시작하기 전에 각 컴퓨팅에서 스크립트가 실행되는지 확인합니다.

Amazon GameLift Servers 에이전트를 게임 서버 소프트웨어와 함께 배포한 경우 에이전트 실행 파일이 시작되는지 확인합니다.

3단계: 게임 서버 프로세스 시작

컴퓨팅에서 게임 서버 실행 파일의 인스턴스를 실행합니다. 게임 서버 빌드가 제대로 통합되면 게임 서버 프로세스는 유효한 서버 파라미터 세트를 사용하여 서버 SDK 작업 `InitSDK()`를 호출합니다. 서버 프로세스가 게임 세션을 호스팅할 준비가 되면 `ProcessReady()`를 호출합니다.

Note

Amazon GameLift Servers 에이전트를 사용하여 게임 서버 소프트웨어가 배포된 경우 이 단계를 건너뛸 수 있습니다. 에이전트는 사용자가 제공한 런타임 지침에 따라 게임 서버 프로세스를 자동으로 시작합니다.

활성화 및 활성 서버 프로세스에 대한 서버 프로세스 지표를 확인하여 진행 상황을 모니터링할 수 있습니다. [플릿에 대한 Amazon GameLift Servers 지표](#)을(를) 참조하세요. 게임 서버 프로세스가 초기화되지 않는 경우 프로세스가 실행 중인 컴퓨팅에 적합한 서버 파라미터 값을 검색하고 있는지 확인합니다.

하이브리드 호스팅 솔루션 구축

하이브리드 호스팅 솔루션은에서 실행되는 Amazon GameLift Servers 관리형 플릿 AWS 클라우드 과 사용자가 직접 제공하고 관리하는 리소스를 포함하여 게임 호스팅 리소스의 여러 소스를 결합하여 게임을 호스팅합니다. 이 주제에서는 하이브리드 솔루션을 구축하기 위한 몇 가지 일반적인 패턴을 설명하고 자체 관리형 게임 호스팅을 Amazon GameLift Servers에서 관리하는 클라우드 기반 게임 호스팅과 성공적으로 결합하는 방법에 대한 팁을 제공합니다.

일반적인 하이브리드 패턴

- **비용 최적화:** 기존 용량을 위한 Anywhere 플릿 및 최대 수요를 위한 관리형 플릿과 같이 각 시나리오에 가장 비용 효율적인 플릿 유형을 사용합니다.

- **지리적 유연성:** 기존 인프라 또는 특정 규정 준수 요구 사항이 있는 트래픽이 많은 AWS 리전 및 Anywhere 플릿에 관리형 플릿을 배포합니다.
- **위험 완화:** 여러 플릿 유형 및 공급자에 로드를 분산하여 단일 호스팅 접근 방식에 대한 종속성을 줄입니다.
- **점진적 마이그레이션:** 서비스 연속성을 유지하면서 온프레미스 또는 기타 호스팅 솔루션에서 AWS 점진적으로 로 전환합니다.

개발 및 프로덕션 분할

개발과 프로덕션 환경에 서로 다른 플릿 유형을 사용합니다.

- **개발:** 비용 효율적인 개발 및 테스트를 위한 Anywhere 플릿
- **프로덕션:** 확장 가능하고 안정적인 프로덕션 호스팅을 위한 관리형 Amazon EC2 또는 컨테이너 플릿

리전 최적화

AWS 리전 특성에 따라 플릿 유형 최적화:

- **트래픽이 많은 AWS 리전:** 가변 수요에 대한 Auto Scaling을 사용하는 관리형 플릿
- **전문:** AWS 리전 규정 준수, 데이터 주권 또는 기존 인프라를 위한 Anywhere 플릿

용량 계층화

용량 계층마다 다른 플릿 유형을 사용합니다.

- **기준 용량:** 예측 가능한 로드를 위한 Anywhere 플릿 또는 예약 인스턴스
- **버스트 용량:** 피크 수요를 위한 오토 스케일링을 사용하는 관리형 플릿
- **오버플로 용량:** 극한 피크에 AWS 리전 대한 스팟 인스턴스 또는 추가

구현 고려 사항

하이브리드 솔루션을 구축할 때 다음 주요 요소를 고려하세요.

게임 세션 대기열 구성

모든 플릿 유형을 포함하도록 게임 세션 대기열을 구성하고 하이브리드 인프라 전체에서 최적의 배치를 보장하기 위해 적절한 우선 순위 및 지연 시간 기본 설정을 지정합니다.

모니터링 및 관찰성

모든 플릿 유형에 걸쳐 포괄적인 모니터링을 구현하여 하이브리드 솔루션 전반의 성능, 용량, 비용에 대한 가시성을 유지합니다.

운영 복잡성

다양한 배포 프로세스, 모니터링 도구, 문제 해결 절차를 포함하여 여러 플릿 유형을 관리할 때 발생하는 운영 복잡성 증가를 고려하세요.

네트워크 연결

특히 온프레미스 또는 다른 클라우드 제공업체에 있을 수 있는 Anywhere 플릿의 경우 다양한 호스팅 환경 간에 안정적인 네트워크 연결을 보장합니다.

하이브리드 호스팅 시작하기

하이브리드 호스팅 솔루션을 구현하려면:

1. 간단한 시작: 단일 플릿 유형으로 시작하고 요구 사항이 더 명확해지면 다른 플릿을 점진적으로 추가합니다.
2. 아키텍처 계획: 비용, 성능, 규정 준수, 운영 복잡성에 대한 특정 요구 사항을 기반으로 하이브리드 아키텍처를 설계합니다.
3. 대기열 구성: 적절한 우선 순위 및 배치 전략을 사용하여 다양한 플릿 유형에 걸쳐 게임 세션 대기열을 설정합니다.
4. 철저한 테스트: 프로덕션으로 이동하기 전에 하이브리드 인프라에서 게임 세션 배치 및 장애 조치 시나리오를 테스트합니다.
5. 모니터링 및 최적화: 모든 플릿 유형에서 성능 및 비용을 지속적으로 모니터링하고 필요에 따라 구성을 조정합니다.

팁

- 관리형 호스팅 리소스와 자체 관리형 호스팅 리소스 모두에서 동일한 게임 클라이언트 및 서버 구성 요소를 사용합니다. 모든 호스팅 리소스에서 통합된 플레이어 환경을 제공합니다.

- 동일한 FlexMatch 매치메이커를 사용하여 모든 호스팅 리소스에 매치를 배치합니다.
- 하이브리드 호스팅 리소스를 전 세계에 배포하는 동안 하이브리드 호스팅 리소스를 중앙에서 함께 관리합니다.
- 플레이어 수요가 변동함에 따라 관리형 및 자체 관리형 리소스 전반에서 게임 세션 로드를 원활하게 관리합니다.
- Amazon GameLift Servers 에이전트를 사용하면 동일한 도구를 사용하여 모든 유형의 호스팅 리소스에서 게임 서버 수명 주기를 관리할 수 있습니다.
- 모든 호스팅 리소스에서 게임 및 플레이어 지표와 로그를 수집합니다. Amazon GameLift Servers 기능 및 기타 AWS 서비스를 활용하여 데이터를 결합하고 일관된 관찰성 솔루션을 개발합니다.

게임 세션 배치 구성

게임 세션 배치는 새 게임 세션을 호스팅하기 위해 사용 가능한 게임 서버를 찾는 프로세스입니다. Amazon GameLift Servers는 게임 세션 대기열을 사용하여 플레이어 지연 시간, 플릿 용량, 비용 최적화와 같은 요소를 고려하여 플릿 전체에 게임 세션을 지능적으로 배치합니다.

잘 구성된 게임 세션 대기열을 사용하면 플레이어가 최상의 게임 서버에 연결되어 최적의 성능을 제공하는 동시에 호스팅 리소스를 효율적으로 사용할 수 있습니다. 대기열은 여러 플릿과 AWS 리전에 걸쳐 글로벌 적용 범위와 중복성을 제공할 수 있습니다.

게임 세션 대기열은 Amazon GameLift Servers가 사용 가능한 게임 서버를 검색하고 새로운 게임 세션을 호스팅하도록 서버를 선택하는 데 사용하는 기본 메커니즘입니다. 대기열은 많은 수의 게임 세션 요청을 처리하고 여러 호스팅 리소스 플릿에서 해당 요청을 위한 배치를 찾는 훨씬 더 효율적인 방법을 제공합니다. 호스팅 솔루션이 두개 이상의 플릿을 사용하고 대용량의 요청을 처리하는 경우 대기열이 필요할 수 있습니다.

게임이 플레이어를 위한 새 게임 세션을 시작하려고 하면 Amazon GameLift Servers 서비스에 배치 요청을 보내 대기열로 유도합니다. 대기열의 구성에 따라 요청이 처리되는 시기와 방법이 결정됩니다. 배치 요청을 처리할 때 Amazon GameLift Servers는 게임 세션을 호스팅할 게임 서버 집합을 검색합니다. Amazon GameLift Servers가 사용 가능한 게임 서버를 찾고 게임 세션을 시작하라는 메시지를 표시하면 배치가 성공합니다.

주제

- [대기열 특성](#)
- [Amazon GameLift Servers 게임 세션 대기열에 대한 모범 사례](#)
- [게임 세션 대기열 생성](#)

- [게임 세션 배치의 이벤트 알림 설정](#)

대기열 특성

Amazon GameLift Servers 게임 세션 대기열은 AWS 클라우드 리소스입니다. Amazon GameLift Servers가 지원하는 AWS 리전 에서 대기열을 생성할 수 있습니다([Amazon GameLift Servers 서비스 위치](#) 참조). 게임 세션 배치 요청은 해당 위치로 전송되어 처리됩니다.

대기열을 사용한 게임 세션 배치 자동화는 게임 개발자와 플레이어 모두에게 상당한 이점을 제공합니다. 다음이 포함됩니다.

- 대기열로 '최상'의 배치를 제공할 수 있습니다. 게임 세션 배치 요청을 처리할 때 대기열은 Amazon GameLift Servers FleetIQ 알고리즘을 사용하여 비용, 위치, 플레이어 지연 시간 등 정의된 기본 설정 세트를 기반으로 배치의 우선 순위를 지정합니다.
- 대기열은 게임 호스팅 비용을 줄이는 데 도움이 되는 스팟 플릿을 지원합니다. AWS 스팟 플릿으로 대기열을 구성할 수 있습니다. 이 플릿은 종종 온디맨드 플릿뿐만 아니라 호스팅 비용도 크게 절감됩니다. 저렴한 비용이 배치의 주요 기준 중 하나이므로 대기열은 항상 비용 차이를 활용할 수 있습니다.
- 수요가 많을 때 대기열은 새 게임을 더 빨리 배치할 수 있습니다. 여러 플릿으로 대기열을 구성하면 게임 세션 배치를 위한 보다 유연한 옵션을 제공할 수 있습니다. 그러나 추가 플릿은 수요가 증가할 때 필요에 따라 백업 용량도 제공합니다. 배치 요청의 경우 Amazon GameLift Servers가 게임 세션을 가장 선호하는 위치에 배치할 수 없는 경우 다른 위치를 평가하기 위해 자동으로 이동합니다.
- 게임 가용성을 보다 신속하게 복원할 수 있습니다. 중단 사태가 발생할 수 있습니다. 다중 플릿 대기열을 사용하면 감속이나 정전으로 인해 플레이어의 게임 액세스에 영향이 미치지 않습니다. 서로 다른 AWS 리전 및 가용 영역에 용량이 있는 플릿으로 대기열을 구성하면 플레이어가 언제든지 참여할 게임 세션을 찾을 수 있도록 할 수 있습니다.
- 게임 세션 배치 및 대기열 성능 지표를 받습니다. Amazon GameLift Servers는 배치 성패 여부에 대한 통계, 대기열의 요청 수, 대기열에서 요청이 기다리는 평균 시간 같은 대기열 지표를 출력합니다. Amazon GameLift Servers 콘솔 또는 CloudWatch에서 이러한 지표를 볼 수 있습니다.

기본 스타터 대기열을 생성하여 시작하려면 [게임 세션 대기열 생성](#) 섹션을 참조하세요.

주제

- [Amazon GameLift Servers 게임 세션 대기열에 대한 모범 사례](#)
- [게임 세션 대기열 생성](#)
- [게임 세션 배치의 이벤트 알림 설정](#)

Amazon GameLift Servers 게임 세션 대기열에 대한 모범 사례

게임 세션 대기열은 Amazon GameLift Servers가 새 게임 세션을 배치할 수 있는 플릿 위치의 목록을 포함합니다. 각 플릿은 여러 지리적 위치에 배포된 호스팅 리소스를 포함할 수 있습니다. 배치를 선택할 때 대기열이 플릿에 대해 설정한 우선 순위 집합을 기반으로 플릿과 플릿 위치를 선택합니다.

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 축소된 상태로 유지하고 사용량이 증가하면 오토 스케일링을 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.

게임 세션 대기열 생성

대기열은 여러 플릿과 위치에 새 게임 세션을 배치하는 데 사용됩니다. 게임은 배치 요청을 대기열에 제출하여 새 게임 세션을 시작합니다. 대기열은 요청을 처리하는 방법에 대한 지침으로 구성됩니다. [게임 세션 만들기](#) 섹션에서 게임 세션 배치 요청의 시작에 대한 자세한 내용을 알아봅니다.

게임 세션 대기열 생성

이 지침은 최소한의 구성 설정과 기본 설정으로 간단한 작업 대기열을 생성하는 방법을 보여 줍니다. 대기열 구성을 사용자 지정하는 방법에는 다양한 옵션이 있습니다. 이러한 옵션을 사용하면 게임의 요구 사항에 따라 최대한 배치할 수 있습니다. 게임에서 대기열을 사용자 지정하는 방법에 대한 자세한 내용은 [게임 세션 대기열 사용자 지정](#) 섹션을 참조하세요. 언제든지 대부분의 대기열 구성 설정을 업데이트할 수 있습니다.

Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 게임 세션 대기열을 생성할 수 있습니다.

Console

[Amazon GameLift Servers 콘솔](#)에서 작업할 AWS 리전을 선택합니다. 콘솔의 왼쪽 탐색 모음을 열고 대기열을 선택합니다.

1. 대기열 페이지에서 대기열 생성을 선택하여 워크플로를 시작합니다.
2. 대기열 설정에서 다음 설정을 입력합니다.
 - a. 대기열 이름을 입력합니다. 이 이름은 대기열을 AWS 리전 생성하는에 고유해야 합니다.
 - b. 기본 제한 시간 설정을 600초(또는 10분)로 유지합니다. 이 값은 Amazon GameLift Servers가 중지하기 전에 새 게임 세션을 배치하려는 시간을 제어합니다. Amazon GameLift Servers는 요청 시간이 초과될 때까지 사용 가능한 리소스를 검색합니다. 언제든지 대기열의 제한 시간 설정을 업데이트할 수 있습니다.
 - c. 플레이어 지연 시간 정책 섹션을 건너뛵니다. 대기열은 플레이어 지연 시간 데이터가 포함된 배치 요청을 수신하는 경우에만 지연 시간 정책을 사용합니다. 언제든지 대기열에 지연 시간 정책을 추가할 수 있습니다. 지연 시간 정책 생성에 대한 자세한 내용은 [플레이어 대기 정책 생성](#) 섹션을 참조하세요.
3. 모든 위치의 기본 설정을 사용하려면 게임 세션 배치 위치 섹션을 건너뛵니다. 이 설정을 사용하면 대기열이 배치할 수 있는 위치의 허용 목록을 생성할 수 있습니다(필터 구성이라고도 함). 위치 및 필터 구성별 우선순위 지정에 대한 자세한 내용은 [위치별 배치 우선순위 지정](#) 섹션을 참조하세요.
4. 대상 순서에서 하나 이상의 플릿을 대기열에 추가합니다. 플릿 ID 또는 ARN을 사용하거나 플릿 별칭을 사용하여 플릿을 식별할 수 있습니다. 여러 플릿을 추가할 때는 모두 유사한 게임 빌드를 실행하고 이 대기열을 사용하는 모든 게임 클라이언트와 호환되어야 한다는 점에 유의하세요. 또한 대기열에 있는 모든 플릿에는 동일한 인증서 구성이 있어야 합니다.
 - a. 플릿 또는 별칭이 생성된 리전을 선택합니다. 다중 위치 플릿의 경우 '홈' 리전입니다.
 - b. 대상 유형에서 플릿 또는 별칭을 선택합니다.

- c. 리전 및 유형 선택은 기존 플릿 또는 별칭의 드롭다운 목록을 채웁니다. 대기열 대상으로 지정할 하나를 선택합니다.
- d. 대기열에 다른 플릿 또는 별칭을 지정하려면 대상 추가를 선택하고 이전 단계를 반복합니다.
- e. 대상 목록을 추가한 후 드래그 앤 드롭 기능을 사용하여 대상의 순서를 변경합니다.

Amazon GameLift Servers는 대상별 배치의 우선순위를 지정할 때 이 순서를 사용합니다.

5. 게임 세션 배치 우선순위 섹션을 건너뛰어 기본 우선순위 순서를 유지합니다. 이 설정을 사용하면 Amazon GameLift Servers가 새 게임 세션 배치에 사용할 수 있는 호스팅 리소스를 어디에서 찾을지 선택하는 방법을 사용자 지정할 수 있습니다. 배치의 우선순위 지정에 대한 자세한 내용은 [게임 세션 배치 우선순위](#) 섹션을 참조하세요. 대기열의 배치 우선순위는 언제든지 업데이트할 수 있습니다.
6. 위치 순서에서 기본값을 유지합니다. 이 설정은 플릿 위치를 기준으로 우선순위를 지정할 때 사용됩니다. 사용할 위치 순서를 제공합니다. 기본 우선순위 설정을 사용하는 경우 선호 대상이 여러 위치가 있는 플릿일 때 위치가 타이브레이커로 사용됩니다.
7. 선택적 이벤트 알림 설정 섹션을 건너뛵니다. 대량의 배치 요청을 처리하는 대기열에는 이벤트 알림이 필요합니다. 개발 또는 테스트 목적과 같이 처리량이 적은 대기열의 경우 [DescribeGameSessionPlacement](#)로 폴링하여 배치 요청의 상태를 추적할 수 있습니다. 자세한 내용은 [게임 세션 배치의 이벤트 알림 설정](#) 섹션을 참조하세요. 언제든지 대기열의 이벤트 알림 설정을 업데이트할 수 있습니다.
8. 생성을 선택하여 사용자 지정을 최소화하면서 새 대기열을 생성합니다.

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 위치에만 배치를 허용하도록 필터링합니다.
- 비용 기준 우선순위 순서 및 지정된 순서의 위치.

```
aws gamelift create-game-session-queue \
```

```
--name "sample-test-queue" \
--timeout-in-seconds 300 \
--destinations DestinationArn="arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-772266ba-8c82-4a6e-b620-a74a62a93ff8" DestinationArn="arn:aws:gamelift:us-
east-1:111122223333:fleet/fleet-33f28fb6-aa8b-4867-85b4-ceb217bf5994" \
--filter-configuration "AllowedLocations=us-east-1, ca-central-1, us-east-2, us-
west-2" \
--priority-configuration PriorityOrder="COST","LOCATION",LocationOrder="us-
east-1","us-east-2","ca-central-1","us-west-2" \
--notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

Note

플릿 또는 별칭 ID로 [describe-fleet-attributes](#) 또는 [describe-alias](#)를 호출하여 플릿 및 별칭 ARN 값을 가져올 수 있습니다.

create-game-session-queue 요청이 성공하면 Amazon GameLift Servers가 새 대기열 구성과 함께 [GameSessionQueue](#) 객체를 반환합니다. 이제 [StartGameSessionPlacement](#)를 사용하여 대기열에 요청을 제출할 수 있습니다.

Example플레이어 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 GameLift Servers가 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 GameLift Servers에서 대기열을 사용하여 Amazon Simple Notification Service(Amazon SNS) 토픽에 이벤트 알림을 게시하도록 합니다.
- 자동으로 게시된 Amazon EventBridge 이벤트 및 이벤트 관리 도구 모음을 사용합니다.

Amazon GameLift Servers에서 내보낸 게임 세션 배치 이벤트 목록에 대해서는 [게임 세션 배치 이벤트](#) 섹션을 참조하세요.

Important

대용량 배치 시스템의 경우 FIFO 주제가 아닌 표준(비 FIFO) Amazon SNS 주제를 사용하는 것이 좋습니다. FIFO 주제는 표준 주제보다 게시 제한이 적기 때문에 높은 로드 시 스로틀링 예외가 발생할 수 있습니다. FIFO 주제에 스로틀링이 발생하면 대기열 배치 알림이 손실될 수 있습니다.

SNS 주제 설정

Amazon GameLift Servers가 게임 세션 대기열에서 생성된 모든 이벤트를 주제에 게시하도록 하려면 알림 대상 필드를 주제로 설정합니다.

Amazon GameLift Servers 이벤트 알림에 대한 SNS 주제를 설정하려면

1. 에 로그인 AWS Management Console 하고 <https://console.aws.amazon.com/sns/v3/home> 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.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sns:Publish",
```



```

    "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/your_queue_name"
      }
    }
  }
]
}

```

- c. (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 암호화](#)를 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 GameLift Servers는 모든 게임 세션 배치 이벤트를 EventBridge에 자동으로 게시합니다. EventBridge를 사용하면 이벤트를 처리 대상으로 라우팅하도록 규칙을 설정할 수 있습니다. 예를 들어 게임 세션에 연결하기 전에 작업을 처리하는 PlacementFulfilled AWS Lambda 함수로 이벤트를 라우팅하는 규칙을 설정할 수 있습니다. Eventbridge에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 [Amazon EventBridge란 무엇입니까?](#)를 참조하세요.

다음은 Amazon GameLift Servers 대기열에 사용할 수 있는 EventBridge 규칙의 몇 가지 예입니다.

모든 Amazon GameLift Servers 대기열의 이벤트와 일치합니다.

```

{
  "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"
    ]
}

```

를 사용한 참조 구현 AWS CDK

Amazon SNS AWS Lambda 및 Amazon DynamoDB를 사용한 이벤트 기반 게임 세션 배치의 전체 참조 구현은 Amazon GameLift Toolkit의 [이벤트 기반 게임 세션 배치 지침](#)을 참조하세요. 이 지침에는 게임 세션 배치 이벤트를 처리하는 AWS Lambda 함수가 있는 전체 이벤트 기반 게임 세션 배치 시스템을 배포하기 위한 AWS CDK 템플릿, 게임 세션 배치 상태를 추적하기 위한 Amazon DynamoDB 테이블, Amazon GameLift Servers 대기열 구성 모범 사례가 포함되어 있습니다.

게임 호스팅 솔루션에 맞게 사용자 지정

기본 게임 호스팅 솔루션을 마련한 상태에서 다음 주제를 사용하여 플레이어 경험을 개선하고, 비용을 최적화하며, 고급 기능을 추가하도록 사용자 지정하고 개선합니다. 이 섹션에서는 주로 영향을 미치는 구성 요소에 따라 구성된 다양한 사용자 지정 옵션을 다룹니다. 게임의 요구 사항과 플레이어 기반에 가장 적합한 사용자 지정을 선택합니다.

주제

- [게임 서버 빌드 사용자 지정](#)
 - [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#)
 - [게임 서버가 Amazon GameLift Servers 플릿 데이터에 액세스하도록 허용](#)
 - [Amazon GameLift Servers의 VPC 피어링 설정](#)
- [플레이어 세션 및 매치메이킹 사용자 지정](#)
 - [플레이어 ID 생성](#)
 - [Amazon GameLift Servers에 FlexMatch 매치메이킹 추가](#)
- [게임 세션 배치 사용자 지정](#)
 - [게임 세션 대기열 사용자 지정](#)
 - [게임 세션 배치 우선순위](#)
 - [스팟 인스턴스를 위한 대기열 빌드](#)
- [리소스 사용자 지정 호스팅](#)
 - [관리형 플릿의 컴퓨팅 리소스 선택](#)

- [Amazon GameLift Servers 컨테이너 플릿 사용자 지정](#)
- [스팟 플릿으로 게임 호스팅 비용 절감](#)
- [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)
- [Amazon GameLift Servers 에이전트 사용법](#)
- [별칭을 사용하여 Amazon GameLift Servers 플릿 지정 추상화](#)

게임 서버 빌드 사용자 지정

게임 서버 빌드 사용자 지정은 다른 AWS 서비스를 활용하는 등 게임 서버의 기능을 향상시킬 수 있는 기회를 제공합니다. 이러한 사용자 지정은 게임 서버 기능을 기본 호스팅 이상으로 확장하여 고급 기능 및 통합을 지원합니다.

Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결

Amazon GameLift Servers 플릿에 배포할 게임 서버 빌드를 생성할 때는 게임 빌드의 애플리케이션이 소유한 다른 AWS 리소스와 직접 안전하게 통신하기를 원할 수 있습니다. Amazon GameLift Servers는 게임 호스팅 플릿을 관리하므로 Amazon GameLift Servers에 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 GameLift Servers는 액세스를 설정하기 위해 다음과 같은 방법을 지원합니다.

- [IAM 역할을 사용하여 AWS 리소스에 액세스](#)
- [VPC 피어링을 사용하여 AWS 리소스에 액세스](#)

IAM 역할을 사용하여 AWS 리소스에 액세스

IAM 역할을 사용하여 리소스에 액세스할 수 있는 사용자를 지정하고 해당 액세스에 제한을 설정합니다. 신뢰할 수 있는 당사자는 역할을 “수입”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 역할의 권한을 사용할 Amazon GameLift Servers 권한을 부여하는 신뢰 정책을 사용하여 IAM 역할을 생성합니다.

IAM 역할을 생성하는 방법에 대한 지침은 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#) 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 GameLift Servers와 통신하는 데 사용하는 서버 SDK 버전을 기반으로 애플리케이션을 수정하는 방법을 결정합니다.

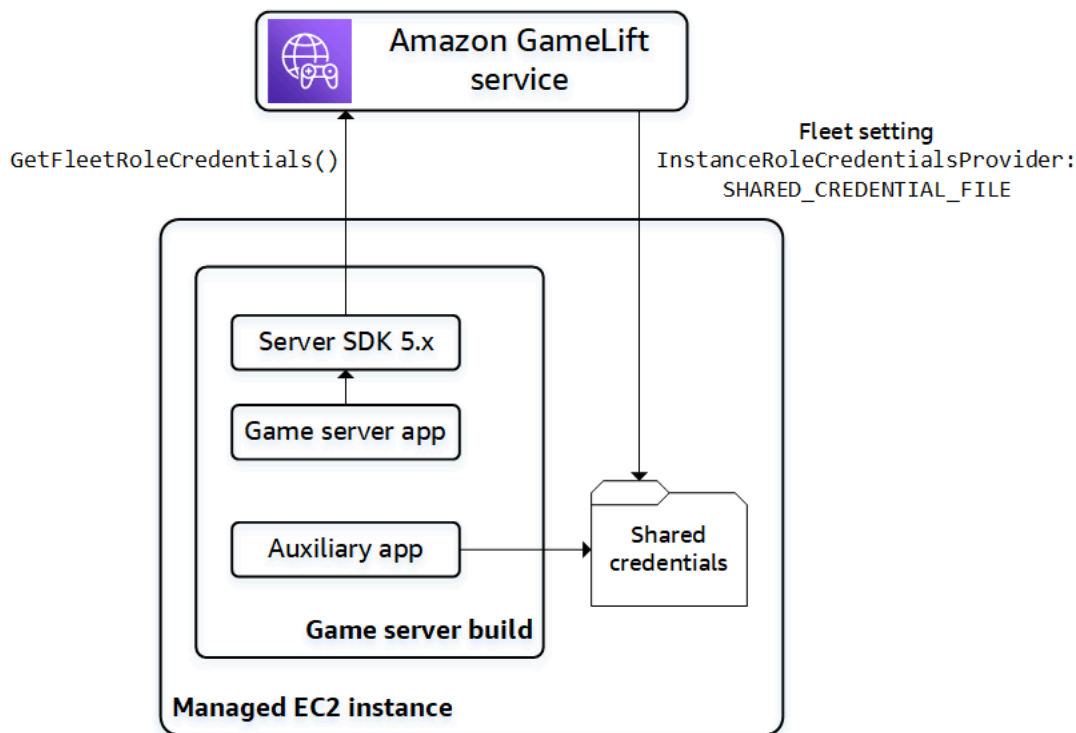
	게임 서버 애플리케이션	기타 애플리케이션
Server SDK 버전 5.x 사용	게임 서버 코드에서 Server SDK 메서드 <code>GetFleetRoleCredentials()</code> 를 호출합니다.	코드를 애플리케이션에 추가하여 플릿 인스턴스의 공유 파일에서 자격 증명을 가져옵니다.

	게임 서버 애플리케이션	기타 애플리케이션
Server SDK 버전 4 또는 이전 버전 사용	역할 ARN을 AssumeRole 사용하여 AWS Security Token Service (AWS STS)를 호출합니다.	역할 ARN을 AssumeRole 사용하여 AWS Security Token Service (AWS STS)를 호출합니다.

Note

컨테이너 플릿의 경우 자격 FleetRoleArn 증명이 각 컨테이너에 주입됩니다. 애플리케이션은 기본 AWS 자격 증명 공급자를 사용하여 이러한 자격 증명에 액세스할 수 있습니다. 동일한 자격 증명을 반환 `GetFleetRoleCredentials()` 하는를 계속 호출할 수 있습니다. 이러한 플릿 역할 자격 증명은 컨테이너 내에서만 액세스할 수 있습니다.

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 Servers 서버 SDK 5.x와 이미 통합되어 있어야 하는 게임 서버 코드에서 `GetFleetRoleCredentials` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#))를 호출하여 임시 자격 증명 세트를 검색합니

다. 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 GameLift Servers는 각 플릿 인스턴스에 대한 자격 증명 프로필을 생성합니다. 자격 증명은 인스턴스의 모든 애플리케이션에서 사용할 수 있습니다. Amazon GameLift Servers는 임시 자격 증명을 지속적으로 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 Servers 플릿 인스턴스에 대한 지표를 수집하도록 CloudWatch 에이전트를 설정합니다.

Amazon CloudWatch 에이전트를 사용하여 Amazon GameLift Servers 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)

애플리케이션에 코드를 추가하여 IAM 역할을 수입하고 AWS 리소스와 상호 작용하기 위한 자격 증명을 가져옵니다. Sever SDK 4 또는 이전 버전을 사용하는 Amazon GameLift Servers 플릿 인스턴스에서 실행되는 모든 애플리케이션이 IAM 역할을 수입할 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 애플리케이션에 AWS 리소스에 대한 액세스 권한을 부여하는 임시 자격 증명 세트를 반환합니다. 자세한 내용은 IAM 사용 설명서의 [AWS 리소스에서 임시 자격 증명 사용](#)을 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.

컨테이너 플릿의 경우:

- [FleetRoleArn](#) -이 파라미터를 IAM 역할의 ARN으로 설정합니다.

다른 플릿 유형의 경우:

- [InstanceRoleArn](#) - 이 파라미터를 IAM 역할의 ARN으로 설정합니다.
- [InstanceRoleCredentialsProvider](#) - Amazon GameLift Servers가 각 플릿 인스턴스에 대한 공유 자격 증명 파일을 생성하도록 유도하려면 이 파라미터를 SHARED_CREDENTIAL_FILE로 설정합니다.

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

VPC 피어링을 사용하여 AWS 리소스에 액세스

Amazon Virtual Private Cloud(VPC) 피어링을 사용하여 Amazon GameLift Servers 인스턴스에서 실행되는 애플리케이션과 다른 AWS 리소스 간에 통신할 수 있습니다. VPC는를 통해 관리되는 리소스 세트를 포함하는 사용자가 정의하는 가상 프라이빗 네트워크입니다 AWS 계정. 각 Amazon GameLift Servers 플릿에는 고유의 VPC가 있습니다. VPC 피어링을 사용하면 플릿의 VPC와 다른 AWS 리소스의 VPC 간에 직접 네트워크 연결을 설정할 수 있습니다.

Amazon GameLift Servers는 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 GameLift Servers의 VPC 피어링 설정](#) 섹션을 참조하세요.

Note

컨테이너 플릿에는 VPC 피어링이 지원되지 않습니다.

게임 서버가 Amazon GameLift Servers 플릿 데이터에 액세스하도록 허용

사용자 지정 게임 빌드 또는 Amazon GameLift Servers Realtime 스크립트에 Amazon GameLift Servers 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

gamelift-metadata.json 파일에는 [Amazon GameLift Servers 플릿 리소스의 속성이](#) 포함됩니다.

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",
  "name": "ReallyFunGameTestFleet08",
  "fleetType": "ON_DEMAND",
  "instanceRoleArn": "arn:aws:iam::123456789012:role/S3AccessForGameLift",
  "instanceType": "c5.large",
  "serverLaunchPath": "/local/game/reallyfungame.exe"
```

}

Amazon GameLift Servers의 VPC 피어링 설정

이 주제에서는 Amazon GameLift Servers 호스팅 게임 서버와 다른 비 Amazon GameLift Servers 리소스 사이의 VPC 피어링 연결을 설정하는 방법을 안내합니다. Amazon Virtual Private Cloud(VPC) 피어링 연결을 사용하면 게임 서버가 웹 서비스 또는 리포지토리와 같은 다른 AWS 리소스와 직접 비공개로 통신할 수 있습니다. 에서 실행 AWS 되고 액세스 권한이 있는 AWS 계정에서 관리하는 모든 리소스와 VPC 피어링을 설정할 수 있습니다.

Note

VPC 피어링은 고급 기능입니다. 게임 서버가 다른 AWS 리소스와 직접 비공개로 통신할 수 있도록 하는 기본 옵션에 대한 자세한 내용은 섹션을 참조하세요 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#).

Amazon VPC 및 VPC 피어링에 대해 잘 알고 있는 경우 Amazon GameLift Servers 게임 서버와의 피어링 설정은 약간 다릅니다. 게임 서버(Amazon GameLift Servers 서비스로 제어됨)를 포함하는 VPC에 액세스할 권한이 없으므로 VPC 피어링을 직접 요청할 수 없습니다. 그 대신 비 Amazon GameLift Servers 리소스를 포함하는 VPC를 사전 승인하여 Amazon GameLift Servers 서비스의 피어링 요청을 수락합니다. 그런 다음 방금 승인한 VPC 피어링을 요청하도록 Amazon GameLift Servers를 트리거합니다. Amazon GameLift Servers는 피어링 연결 생성, 라우팅 테이블 설정 및 연결 구성 작업을 처리합니다.

기존 플릿용 VPC 피어링을 설정하려면

1. AWS 계정 ID(들) 및 자격 증명을 가져옵니다.

다음 AWS 계정에 대한 ID 및 로그인 자격 증명が必要です. 에 로그인 [AWS Management Console](#)하고 AWS 계정 설정을 확인하여 계정 IDs를 찾을 수 있습니다. 자격 증명을 가져오려면 IAM 콘솔로 이동합니다.

- AWS Amazon GameLift Servers 게임 서버를 관리하는 데 사용하는 계정입니다.
- AWS 비Amazon GameLift Servers 리소스를 관리하는 데 사용하는 계정입니다.

Amazon GameLift Servers 리소스와 비 Amazon GameLift Servers 리소스에 대해 동일한 계정을 사용할 경우 한 계정에 대해서만 ID와 자격 증명が必要です.

2. 각 VPC에 대한 ID를 가져옵니다.

피어링할 두 VPC에 대해 다음 정보를 가져옵니다.

- Amazon GameLift Servers 게임 서버용 VPC이며, 이는 Amazon GameLift Servers 플릿 ID입니다. EC2 인스턴스 플릿의 Amazon GameLift Servers에 게임 서버가 배포됩니다. 플릿은 해당 VPC에 자동으로 배치되며 Amazon GameLift Servers 서비스에 의해 관리됩니다. VPC에 직접 액세스할 권한이 없어 플릿 ID로 확인됩니다.
- 비 Amazon GameLift Servers AWS 리소스용 VPC -에서 실행 AWS 되고 액세스 권한이 있는 AWS 계정에서 관리하는 모든 리소스와 VPC 피어링을 설정할 수 있습니다. 이러한 리소스에 대한 VPC를 아직 생성하지 않은 경우에는 [Amazon VPC 시작하기](#)를 참조하세요. VPC를 생성한 경우 Amazon VPC용 [AWS Management Console](#)에 로그인한 후 VPC를 확인하여 VPC ID를 찾을 수 있습니다.

Note

피어링을 설정할 때는 두 VPC가 동일한 리전에 있어야 합니다. Amazon GameLift Servers 플릿 게임 서버용 VPC는 플릿과 동일한 리전에 있습니다.

3. VPC 피어링을 승인합니다.

이 단계에서는 비 Amazon GameLift Servers 리소스에 대해 VPC가 있는 게임 서버와 함께 VPC를 피어링하기 위해 Amazon GameLift Servers에서 향후 요청을 사전 승인합니다. 이 작업은 VPC에 대한 보안 그룹을 업데이트합니다.

VPC 피어링을 승인하려면 서비스 API [CreateVpcPeeringAuthorization\(\)](#)을 호출하거나 AWS CLI 명령어를 사용합니다 `create-vpc-peering-authorization`. 비 Amazon GameLift Servers 리소스를 관리하는 계정을 사용하여 호출합니다. 다음 정보를 식별합니다.

- 피어 VPC ID - 비 Amazon GameLift Servers 리소스를 포함하는 VPC를 위한 것입니다.
- Amazon GameLift Servers AWS 계정 ID - Amazon GameLift Servers 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 GameLift Servers가 피어링 연결을 설정하도록 요청할 수 있습니다.

VPC 피어링을 요청하려면 서비스 API [CreateVpcPeeringConnection\(\)](#)을 호출하거나 AWS CLI 명령을 사용합니다. create-vpc-peering-connection. Amazon GameLift Servers 게임 서버를 관리하는 계정을 사용하여 호출합니다. 다음 정보를 사용하여 피어링할 두 VPC를 식별합니다.

- 피어 VPC ID 및 AWS 계정 ID -Amazon GameLift Servers 비 리소스의 VPC와 이를 관리하는 데 사용하는 계정입니다. VPC ID는 유효한 피어링 승인의 ID와 일치해야 합니다.
- 플릿 ID - Amazon GameLift Servers 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 Servers 플릿을 생성하는 동시에 VPC 피어링 연결을 요청할 수 있습니다.

1. AWS 계정 ID(들) 및 자격 증명을 가져옵니다.

다음 두 AWS 계정에 대한 ID와 로그인 자격 증명이 필요합니다. 에 로그인 [AWS Management Console](#)하고 AWS 계정 설정을 확인하여 계정 IDs를 찾을 수 있습니다. 자격 증명을 가져오려면 IAM 콘솔로 이동합니다.

- AWS Amazon GameLift Servers 게임 서버를 관리하는 데 사용하는 계정입니다.
- AWS 비Amazon GameLift Servers 리소스를 관리하는 데 사용하는 계정입니다.

Amazon GameLift Servers 리소스와 비 Amazon GameLift Servers 리소스에 대해 동일한 계정을 사용할 경우 한 계정에 대해서만 ID와 자격 증명이 필요합니다.

2. 비 Amazon GameLift Servers AWS 리소스의 VPC ID를 가져옵니다.

이러한 리소스에 대한 VPC를 아직 생성하지 않았다면 지금 생성합니다([Amazon VPC 시작하기](#) 참조). 새 플릿을 생성할 위치와 동일한 리전에 새 VPC를 생성해야 합니다. 비Amazon GameLift Servers 리소스가에서 사용하는 것과 다른 AWS 계정 또는 사용자/사용자 그룹에서 관리Amazon GameLift Servers되는 경우 다음 단계에서 권한 부여를 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 Servers 리소스와의 VPC 피어링을 승인합니다.

Amazon GameLift Servers가 새 플릿과 해당 VPC를 생성할 때 비 Amazon GameLift Servers 리소스용 VPC와 피어링하기 위한 요청도 전송합니다. 이 요청을 사전에 승인해야 합니다. 이 단계에서는 VPC에 대한 보안 그룹을 업데이트합니다.

비Amazon GameLift Servers 리소스를 관리하는 계정 자격 증명을 사용하여 서비스 API [CreateVpcPeeringAuthorization\(\)](#)을 호출하거나 AWS CLI 명령을 사용합니다create-vpc-peering-authorization. 다음 정보를 식별합니다.

- 피어 VPC ID - 비 Amazon GameLift Servers 리소스를 포함하는 VPC의 ID입니다.
- Amazon GameLift Servers AWS 계정 ID - Amazon GameLift Servers플릿을 관리하는 데 사용하는 계정의 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 사용하여 새 플릿을 생성하는](#) 지침을 따릅니다. 다음 추가 파라미터를 포함합니다.

- `peer-vpc-aws-account-id` - 비 Amazon GameLift Servers 리소스와 함께 VPC를 관리할 때 사용하는 계정용 ID입니다.
- `peer-vpc-id` - 비 Amazon GameLift Servers 계정이 있는 VPC ID입니다.

VPC 피어링 파라미터로 [create-fleet](#)을 성공적으로 호출하여 새로운 플릿 및 VPC 피어링 요청을 생성합니다. 플릿의 상태가 `New`로 설정되고 플릿 활성화 프로세스가 시작됩니다. 피어링 연결 요청의 상태가 `initiating-request`로 설정됩니다. [describe-vpc-peering-connections](#)를 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 `Activating`에서 `Active`로 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. 미리 설정된 VPC는 비Amazon GameLift Servers AWS 계정 ID와 VPC ID의 조합으로 고유하게 식별됩니다.

```
$ 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 Servers 게임 서버에 대한 VPC 피어링 연결을 설정하는 데 문제가 있는 경우, 다음과 같은 일반적인 근본 원인을 확인하십시오.

- 요청된 연결에 대한 승인을 찾을 수 없습니다.
 - 비 Amazon GameLift Servers 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 Servers 플릿에 대한 VPC의 CIDR 블록은 자동으로 할당되며 변경할 수 없으므로 비 Amazon GameLift Servers 리소스에 대해 VPC의 CIDR 블록을 변경해야 합니다. 이 문제를 해결하려면:
 - DescribeVpcPeeringConnections()을 호출하여 해당 Amazon GameLift Servers 플릿에 대한 이 CIDR 블록을 조회합니다.
 - Amazon VPC 콘솔로 이동하여 비 Amazon GameLift Servers 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.

플레이어 세션 및 매치메이킹 사용자 지정

플레이어 세션 및 매치메이킹 사용자 지정을 통해 균형 잡히고 매력적인 멀티플레이어 경험을 제공하는 데 도움이 되는 미묘한 매치메이킹 시스템을 포함하여 정교한 플레이어 관리 워크플로를 개발할 수 있습니다.

플레이어 ID 생성

Amazon GameLift Servers는 플레이어 세션을 사용하여 게임 세션에 연결된 플레이어를 나타냅니다. Amazon GameLift Servers는 플레이어가 Amazon GameLift Servers와 통합된 게임 클라이언트를 사용하여 게임 세션에 연결할 때마다 플레이어 세션을 생성합니다. 플레이어가 게임을 나가면 플레이어 세션이 종료됩니다. Amazon GameLift Servers는 플레이어 세션을 재사용하지 않습니다.

Important

FlexMatch 매치메이킹을 사용할 때 기존 활성 매치메이킹 요청에 이미 포함된 플레이어 ID가 포함된 새 매치메이킹 요청을 생성하면 기존 요청이 자동으로 취소됩니다. 그러나 취소된 요청에 대해서는 MatchmakingCancelled 이벤트가 전송되지 않습니다. 기존 매치메이킹 요청 상태를 모니터링하려면 [DescribeMatchmaking](#)을 사용하여 자주 발생하지 않는 간격(30~60초)으로 요청 상태를 폴링합니다. 취소된 요청은 Cancelled due to duplicate player 이 유인 CANCELLED 상태로 표시됩니다.

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 [Amazon GameLift Servers 콘솔의 게임 및 플레이어 세션](#) 섹션을 참조하세요.

Amazon GameLift Servers에 FlexMatch 매치메이킹 추가

Amazon GameLift Servers FlexMatch를 사용하여 Amazon GameLift Servers 호스팅하는 게임에 플레이어 매치메이킹 기능을 추가합니다. FlexMatch를 사용자 지정 게임 서버 또는 Amazon GameLift Servers Realtime과 함께 사용할 수 있습니다.

FlexMatch는 매치메이킹 서비스를 사용자 지정 가능한 규칙 엔진과 연결합니다. 게임에 적합한 플레이어 속성과 게임 모드를 기반으로 플레이어를 함께 매칭하는 방법을 설계합니다. FlexMatch는 게임을 찾고 있는 플레이어를 평가하고, 하나 이상의 팀과 매치를 구성하며, 매치를 호스팅하기 위해 게임 세션을 시작하는 기본 사항을 관리합니다.

전체 FlexMatch 서비스를 사용하려면 호스팅 리소스를 대기열로 설정해야 합니다. Amazon GameLift Servers는 대기열을 사용하여 여러 리전 및 컴퓨팅 유형에 걸쳐 게임에 가장 적합한 호스팅 위치를 찾습니다. 특히 Amazon GameLift Servers 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 Servers FlexMatch 개발자 가이드](#)의 다음 주제를 참조하세요.

- [Amazon GameLift Servers FlexMatch 작동 방식](#)
- [FlexMatch 통합 단계](#)

게임 세션 배치 사용자 지정

게임 세션 배치 사용자 지정을 통해 배치 시스템을 미세 조정하여 플레이어 기반에 최상의 게임 플레이 경험을 제공할 수 있습니다. 플레이어 호환성 및 기본 설정뿐만 아니라 비용 효율성, 지리적 배포, 서비스 중단에 대한 복원력과 같은 운영 고려 사항에 맞게 배치를 최적화할 수 있습니다.

게임 세션 대기열 사용자 지정

이 주제에서는 게임 세션 배치에 대해 가능한 최상의 결정을 내릴 수 있도록 게임 세션 대기열을 사용자 지정하는 방법을 설명합니다. 게임 세션 대기열 및 작동 방식에 대한 자세한 내용은 [게임 세션 배치 구성](#) 섹션을 참조하세요.

이러한 Amazon GameLift Servers 기능에는 대기열이 필요합니다.

- [FlexMatch를 통한 매치메이킹](#)

- [스팟 인스턴스를 위한 대기열 빌드](#)

주제

- [대기열의 범위 정의](#)
- [다중 위치 대기열 빌드](#)
- [대기열 지표 평가](#)

대기열의 범위 정의

게임의 플레이어 모집단에는 함께 플레이해서는 안 되는 플레이어 그룹이 있을 수 있습니다. 예를 들어 게임을 두 언어로 게시하는 경우 각 언어에는 고유한 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.

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 GameLift Servers가 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 [대기열에 대한 Amazon GameLift Servers 지표](#) 단원을 참조하십시오.

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 [스팟 플릿으로 게임 호스팅 비용 절감](#)의 지침을 따릅니다.

다중 위치 대기열을 생성하는 한 가지 방법은 대기열에 [다중 위치 플릿](#)을 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 배치 위치에 AWS 리전 대한 us-east-1, us-east-2, 및 ca-central-1가 있습니다 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-1에서 대기열을 만듭니다.
2. 새 대기열을 생성하고 대기열 대상으로 다중 위치 플릿을 추가합니다. 대상 순서에 따라 Amazon GameLift Servers가 게임 세션을 배치하는 방식이 결정됩니다. 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 GameLift Servers는 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				

Game session placement priority

The values that GameLift uses to prioritize game session placement. The default order is latency, cost, destination, and location.



Latency

Prioritize locations with the lowest average player latency.



Cost

Prioritize destinations with the lowest current hosting cost.



Destination

Prioritize based on the defined destination order.



Location

Prioritize based on the defined location order.

▼ Location order

An ordered list of locations that the queue can use for game session placement.

Location

ca-central-1



Remove

us-east-1



Remove

us-east-2



Remove

us-west-2



Remove

Add locations

대기열 지표 평가

자체 대기열이 잘 수행되고 있는지 지표를 통해 평가하세요. [Amazon GameLift Servers 콘솔](#) 또는 Amazon CloudWatch에서 대기열과 관련된 지표를 볼 수 있습니다. 대기열 지표의 목록 및 설명은 [대기열에 대한 Amazon GameLift Servers 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 GameLift Servers가 이상적인 배치를 찾지 못하는 일반적인 이유를 식별하는 오류 지표도 있습니다. 지표에 대한 자세한 정보는 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 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.
 - FirstChoiceOutOfCapacity값이 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.

게임 세션 배치 우선순위

Amazon GameLift Servers는 알고리즘을 사용하여 대기열의 대상에 우선순위를 지정하는 방법과 새 게임 세션을 배치할 위치를 결정합니다. 알고리즘은 정렬된 기준 세트를 기반으로 합니다. 기본 우선 순위 순서를 사용하거나 순서를 사용자 지정할 수 있습니다. 대기열의 우선 순위는 언제든지 편집할 수 있습니다.

기본 우선 순위 순서

1. 지연 시간 - 게임 세션 배치 요청에 플레이어의 위치별 지연 시간 데이터가 포함된 경우 Amazon GameLift Servers는 각 위치의 평균 플레이어 지연 시간을 계산하고 평균이 가장 낮은 플릿 위치에 게임 세션을 배치하도록 시도합니다.

2. 비용 - 요청에 지연 시간 데이터가 포함되지 않거나 여러 플릿의 지연 시간이 동일한 경우 Amazon GameLift Servers는 각 플릿의 호스팅 비용을 평가합니다. 플릿의 호스팅 비용은 플릿 유형(스팟 또는 온디맨드), 인스턴스 유형, 위치에 따라 달라집니다.
3. 대상 - 여러 플릿이 지연 시간과 비용이 동일한 경우 Amazon GameLift Servers는 대기열 구성에 나열된 대로 대상 순서에 따라 플릿의 우선 순위를 지정합니다.
4. 위치 - 다중 위치 플릿이 있는 대기열의 경우 다른 모든 기준이 동일한 경우 Amazon GameLift Servers는 알파벳 순서에 따라 플릿의 위치의 우선 순위를 지정합니다.

대기열이 게임 세션 배치의 우선 순위를 지정하는 방법에 대한 사용자 지정

대기열이 배치 기준의 우선 순위를 지정하는 방법을 사용자 지정하도록 선택할 수 있습니다. 대기열은 수신하는 모든 게임 세션 배치 요청에 사용자 지정 우선 순위를 적용합니다.

Note

사용자 지정 우선 순위 구성을 생성할 때 네 가지 기준을 모두 포함하지 않는 경우 Amazon GameLift Servers는 누락된 기준을 기본 순서대로 자동으로 추가합니다.

대기열의 우선 순위 구성을 사용자 지정하려면

[Amazon GameLift Servers 콘솔](#) 또는 AWS Command Line Interface (AWS CLI)를 사용하여 사용자 지정 우선 순위 구성을 생성합니다.

Console

[Amazon GameLift Servers 콘솔](#)에서 새 대기열을 생성하거나 기존 대기열을 업데이트할 때 대기열의 우선 순위를 사용자 지정할 수 있습니다. 작업할 AWS 리전을 선택합니다.

콘솔의 왼쪽 탐색 모음을 열고 대기열을 선택합니다. 대기열 페이지에서 기존 대기열을 선택하고 편집을 선택합니다.

1. 게임 세션 배치 우선순위 섹션으로 이동합니다. 각 우선순위 기준을 끌어서 놓아 원하는 순서를 생성합니다.
2. 위치 순서 섹션으로 이동합니다. 우선순위를 지정할 위치를 추가합니다. 이 목록은 대기열에 여러 위치가 있는 플릿이 있는 경우에 유용합니다. 최소한 한 위치를 지정해야 합니다. 여기에서 지정하는 위치의 우선순위가 먼저 지정되고 대기열의 대상에 있는 다른 모든 위치의 우선순위가 지정됩니다.

3. 변경 사항 저장을 선택합니다.

AWS CLI

[update-game-session-queue](#) 명령을 `--priority-configuration` 옵션과 함께 사용하여 대기열의 우선 순위 순서를 사용자 지정합니다.는 현재 기본 AWS 리전에서 대기열을 Amazon GameLift Servers 업데이트하거나 `--region` 태그를 추가하여 다른 AWS 리전을 지정할 수 있습니다.

다음 예제 요청은 지정된 대기열에 대한 우선순위 구성을 추가하거나 업데이트합니다.

```
aws gamelift update-game-session-queue \
  --name "example-queue-with-priority"
  --priority-configuration
  PriorityOrder="COST", "LOCATION", "DESTINATION", LocationOrder="us-east-1", "us-east-2", "ca-central-1", "us-west-2" \
```

플레이어 지연 시간을 기준으로 배치 우선 순위 지정

플레이어에게 최상의 플레이어 경험을 제공하고 지연 시간을 최소화하려면 게임 세션 배치 시스템을 설정할 때 다음 단계를 수행하세요.

- 게임 세션을 배치할 위치를 선택할 때 지연 시간을 우선순위로 지정하도록 대기열을 설정합니다. 지연 시간은 기본적으로 우선순위 목록의 맨 위에 있습니다. 대기열의 우선순위 구성을 사용자 지정하고 지연 시간을 우선순위에 넣을 위치를 선택할 수도 있습니다.
- 대기열에 대한 플레이어 지연 시간 정책을 설정합니다. 지연 시간 정책을 사용하면 게임 세션 배치에서 허용할 지연 시간에 대한 하드 제한을 설정할 수 있습니다. Amazon GameLift Servers가 제한을 초과하지 않고 게임 세션을 배치할 수 없는 경우 배치 요청이 시간 초과되어 실패합니다. 단일 지연 시간 정책을 설정하거나 시간 경과에 따라 지연 시간 제한을 점진적으로 완화하는 일련의 정책을 생성할 수 있습니다. 일련의 정책을 사용하면 초기 지연 시간 제한을 매우 낮게 지정하고 짧은 지연 시간 후에도 지연 시간이 더 긴 플레이어를 수용할 수 있습니다. 지연 시간 정책 생성에 대한 자세한 내용은 [플레이어 대기 정책 생성](#) 섹션을 참조하세요.
- 게임 세션 배치를 요청할 때([StartGameSessionPlacement](#) 참조) 각 플레이어의 지연 시간 데이터를 포함합니다. 플레이어 지연 시간 데이터에는 게임 세션이 배치될 수 있는 모든 가능한 위치에 대한 값이 포함됩니다. 예를 들어 AWS 리전 us-east-2 및 ca-central-1에 게임 세션을 배치하는 대기열의 경우 지연 시간 데이터는 다음과 같을 수 있습니다.

```
"PlayerLatencies": [
```



```
{ "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "us-east-2" },
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "ca-central-1" },
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "us-east-2" },
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "ca-central-1" }
]
```

정확한 지연 시간 측정값을 얻으려면 Amazon GameLift Servers의 UDP 핑 비콘을 사용합니다. 이러한 엔드포인트를 사용하면 플레이어 디바이스와 각 잠재적 호스팅 위치 간의 실제 UDP 네트워크 지연 시간을 측정할 수 있으므로 ICMP Ping을 사용하는 것보다 더 정확한 배치 결정을 내릴 수 있습니다. UDP 핑 비콘을 사용하여 지연 시간을 측정하는 방법에 대한 자세한 내용은 [UDP 핑 비콘](#) 섹션을 참조하세요.

위치별 배치 우선순위 지정

우선순위가 지정된 지리적 위치 목록을 기반으로 게임 세션을 배치하도록 대기열을 구성할 수 있습니다. 위치는 대기열이 새 게임 세션을 배치할 위치를 선택하는 방법을 결정하는 기준 중 하나입니다. 기본적으로 위치는 지연 시간, 비용, 대상 이후 네 번째로 우선순위가 지정됩니다.

게임 세션 배치의 경우 대상과 위치의 의미는 약간 다릅니다.

- 대상은 특정 플릿을 가리키며 배포된 모든 플릿의 호스팅 리소스를 포함합니다. 대상에 따라 우선순위를 지정할 때 Amazon GameLift Servers는 플릿의 모든 위치에 배치할 수 있습니다. 다중 위치 관리형 플릿 및 Anywhere 플릿에는 하나 이상의 위치에 배포되는 호스팅 리소스가 있을 수 있습니다.
- 위치는 플릿의 호스팅 리소스가 배포되는 특정 지리적 위치를 나타냅니다. 플릿에는 AWS 리전로컬 영역 또는 사용자 지정 위치(Anywhere 플릿의 경우)를 포함할 수 있는 여러 위치가 있을 수 있습니다. 단일 위치 관리형 플릿은 하나의 위치를 가지며 항상 AWS 리전입니다. 다중 위치 관리형 플릿은 홈 리전을 보유하며 원격 위치를 보유할 수 있습니다. Anywhere 플릿에는 하나 이상의 사용자 지정 위치가 있습니다.

위치별로 배치의 우선순위를 지정할 때 Amazon GameLift Servers는 우선순위 위치가 포함된 대기열 대상을 찾고 사용 가능한 호스팅 리소스를 검색합니다. 우선순위 위치가 있는 대상이 여러 개 있는 경우 Amazon GameLift Servers는 다음 우선순위 기준(비용, 지연 시간, 대상)으로 이동합니다.

대기열 위치의 우선순위 지정 방식에 영향을 줄 수 있는 몇 가지 방법이 있습니다.

- 대기열이 모든 게임 세션 배치 요청을 처리하는 방법을 구성합니다.
- 대기열에 우선순위 구성을 추가합니다. 대기열의 우선순위 구성에는 정렬된 위치 목록이 포함됩니다. 우선순위를 지정할 위치를 하나 이상 지정할 수 있습니다. 이 목록은 어떤 위치도 제외하지 않으며 Amazon GameLift Servers에게 사용 가능한 호스팅 리소스를 어디에서 먼저 찾아야 하는지 알려줄 뿐입니다. 정렬된 위치 목록의 일반적인 용도는 대부분의 트래픽을 하나 이상의 특정 지리적 위치로 유도하고 추가 위치를 백업 용량으로 사용하려는 경우입니다. [UpdateGameSessionQueue](#)를 호출하여 우선순위 구성을 추가합니다.
- 대기열에 필터 구성을 추가합니다. 필터 구성은 대기열의 허용 목록입니다. 사용 가능한 호스팅 리소스를 찾을 때 목록에 없는 모든 위치를 무시하도록 Amazon GameLift Servers에 지시합니다. 필터 구성에는 두 가지 일반적인 용도가 있습니다. 첫 번째로 여러 위치를 가진 플릿의 경우 필터를 사용하여 플릿 위치 중 일부를 제외할 수 있습니다. 두 번째로 특정 위치에 대한 배치를 일시적으로 허용하지 않을 수 있습니다. 예를 들어 위치에 일시적인 문제가 발생할 수 있습니다. 언제든지 대기열의 필터 구성을 업데이트할 수 있으므로 필요에 따라 위치를 쉽게 추가하고 제거할 수 있습니다. [UpdateGameSessionQueue](#)를 호출하여 필터 구성을 추가합니다.
- 개별 배치 요청에 대한 특별 지침을 사용합니다.
- 게임 세션 배치 요청에 우선순위 재정의 목록을 포함합니다. [StartGameSessionPlacement](#) 요청과 함께 대체 우선순위 위치 목록을 제공할 수 있습니다. 이 목록은 단일 요청에 한해서만 대기열에 구성된 위치 우선순위 지정을 효과적으로 대체합니다. 다른 요청에는 영향을 주지 않습니다. 이 재정의 기능에는 몇 가지 요구 사항이 있습니다.
 - 우선순위가 LOCATION인 구성이 첫 번째 우선순위인 대기열에서만 재정의 목록을 사용합니다.
 - 동일한 배치 요청에 플레이어 지연 시간 데이터를 포함하지 마세요. 지연 시간 데이터를 포함하면 Amazon GameLift Servers가 해결할 수 없는 위치의 우선순위를 지정할 때 충돌이 발생합니다.
 - Amazon GameLift Servers가 우선순위 재정의 목록에서 사용 가능한 리소스를 찾을 수 없는 경우 진행할 방법을 결정합니다. 대기열의 다른 위치로 대체할지 또는 재정의 목록으로 배치를 제한할지 선택합니다. 기본적으로 Amazon GameLift Servers는 대기열의 다른 위치에 배치하려고 대체합니다.
 - 재정의 목록에 위치를 추가하는 등 필요에 따라 대기열의 필터 구성을 업데이트합니다. 재정의 목록은 필터 목록을 무효화하지 않습니다.

플레이어 대기 정책 생성

배치 요청에 플레이어 지연 시간 데이터가 포함된 경우 Amazon GameLift Servers는 모든 플레이어의 평균 지연 시간이 가장 낮은 위치에서 게임 세션을 찾습니다. 평균 플레이어 지연 시간을 기준으로 게

임 세션을 배치하면 Amazon GameLift Servers에서 지연 시간이 긴 게임에 대부분의 플레이어를 배치하지 못합니다. 하지만 Amazon GameLift Servers는 여전히 플레이어에게 극심한 지연 시간을 안겨줍니다. 이러한 플레이어를 수용하려면 플레이어 지연 시간 정책을 생성해야 합니다.

플레이어 지연 시간 정책을 통해 Amazon GameLift Servers는 요청 대상 플레이어가 최대값을 초과하는 지연 시간을 경험할 수 있는 위치에 요청된 게임 세션을 배치하지 못하게 됩니다. 플레이어 지연 시간 정책은 Amazon GameLift Servers가 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 Servers FlexMatch](#)를 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.

Create queue

Queue settings

Name

The name must be unique and have 1-128 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Timeout

Specify how long GameLift tries to place a game session before stopping.

 seconds

Must be 10-600 seconds.

 We recommend setting player latency policies, unless you're using GameLift FlexMatch.



Player latency policies - *optional*

Add policies to help place players into games with lower latency. Use multiple policies to reduce latency requirements per policy so that each player eventually finds a match.

0 seconds left to allocate

100%

Period start

Seconds

Period end

Seconds

Max player latency

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Add policy

스팟 인스턴스를 위한 대기열 빌드

스팟 플릿을 사용하여 호스팅 비용을 잠재적으로 크게 절감할 수 있습니다. 스팟 플릿 및 사용 방법에 대한 자세한 내용은 [온디맨드 인스턴스 및 스팟 인스턴스 비교](#) 섹션을 참조하세요.

게임 호스팅 솔루션에 스팟 플릿이 포함된 경우 게임 세션 배치 대기열을 사용해야 합니다. Amazon GameLift Servers는 대기열을 사용하여 여러 게임 호스팅 리소스를 검색하고 새 게임 세션을 호스팅하는 데 사용할 수 있는 최상의 리소스를 선택합니다. 스팟 플릿을 사용하면 호스팅 비용을 최소화하고 스팟 중단 가능성을 방지하는 데 대기열이 특히 중요합니다. 이 주제는 중단, 속도 저하, 중단 발생 시에도 플레이어가 게임을 계속 호스팅할 수 있도록 복원력이 뛰어난 대기열을 설정하는 데 도움이 됩니다. 호스팅 비용을 포함한 여러 요인을 기반으로 대기열이 사용 가능한 호스팅 리소스의 우선순위를 지정하는 방법을 사용자 지정할 수 있습니다.

FlexMatch를 매치메이킹에 사용하고 있나요? 매치에 대한 게임 세션 배치를 위해 스팟 플릿과 함께 대기열을 사용할 수 있습니다

스팟 플릿의 구현 작업

스팟 플릿을 사용하도록 게임 호스팅 솔루션을 생성하거나 업데이트할 때 다음 작업을 완료합니다. 스팟 가용성 및 복원력을 최적화하는 대기열을 구축하는 방법에 대한 자세한 지침은 [스팟 플릿으로 게임 호스팅 비용 절감](#) 섹션을 참조하세요.

1. 게임 세션 대기열에 대한 플릿 대상 세트를 선택하고 생성합니다.

먼저 대기열에서 게임 세션을 배치할 위치를 결정합니다. 대기열은 여러 플릿을 검색하여 최상의 배치를 찾을 수 있습니다. 각 플릿에는 하나의 인스턴스 유형을 보유하지만, 여러 지리적 위치를 보유할 수 있습니다. 위치 및 인스턴스 유형 모두에서 다양성을 제공하는 플릿이 있는 대기열은 성공적으로 배치될 가능성이 더 높습니다. 효과적이고 복원력이 뛰어난 스팟 최적화 대기열을 설계하려면 다음 모범 사례를 참조하세요.

2. 스팟 최적화 게임 세션 대기열을 생성합니다.

대기열을 생성하고 스팟 플릿에 맞게 구성합니다. 새 대기열의 생성 및 구성에 대한 자세한 내용은 [게임 세션 대기열 생성](#) 섹션을 참조하세요. Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 대기열을 생성하거나 편집할 수 있습니다.

- 1단계에서 플릿 대상을 추가합니다.
- 대상 순서의 우선순위를 적절하게 지정합니다. 기본적으로 Amazon GameLift Servers는 대상보다 비용을 기준으로 우선순위를 지정하므로 대상 순서는 대상 간의 최저 비용이 동일할 때만 사용됩니다.
- 플레이어 지연 시간보다 게임 호스팅 비용을 우선순위에 지정하려면 사용자 지정 배치 우선순위를 제공합니다. [게임 세션 배치 우선순위](#)(를) 참조하세요.

3. 새 대기열을 사용하도록 솔루션의 다른 구성 요소를 업데이트합니다.

솔루션이 스팟 최적화 대기열을 사용하여 새 게임 세션을 시작하면 대기열은 중단 가능성이 높은 플릿에 게임 세션을 자동으로 배치하지 않습니다. 대신에, 플레이어 지연 시간, 호스팅 비용, 대상 순서를 포함하여 정의된 우선순위와 일치하는 리소스를 모든 실행 가능한 플릿에서 검색합니다.

- FlexMatch를 사용하지 않는 경우 - 게임 세션 요청에서 새 스팟 최적화 대기열을 지정하도록 백엔드 서비스를 업데이트합니다. 백엔드 서비스는 게임 클라이언트 (StartGameSessionPlacement() 사용)를 대신하여 Amazon GameLift Servers에 API 요청을 하며, 각 요청은 대기열 이름을 지정해야 합니다. 게임 클라이언트에서 게임 세션 배치를 구현하는 방법은 [게임 세션 만들기](#) 섹션을 참조하세요.
- FlexMatch를 사용 중인 경우 - 게임 세션 요청을 새 스팟 최적화 대기열로 전송하도록 매치메이킹 구성을 업데이트합니다. 매치메이킹 시스템이 플레이어 매치를 구성할 때 게임 세션 배치 요청을 지정된 대기열로 보내 매치를 위한 새 게임 세션을 시작합니다. FlexMatch 모드가 '관리형'으로 설정된 매치메이킹 구성만 배치 대기열을 지정할 수 있습니다. AWS CLI 또는 Amazon GameLift Servers 콘솔을 사용하여 매치메이킹 구성을 업데이트할 수 있습니다([매치메이킹 구성 편집](#) 참조).

4. 스팟 플릿 및 대기열의 성능을 검토합니다.

Amazon GameLift Servers 콘솔 또는 Amazon CloudWatch를 사용하여 Amazon GameLift Servers 지표를 보고 성능을 검토할 수 있습니다. Amazon GameLift Servers 지표에 대한 자세한 정보는 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#) 단원을 참조하십시오. 주요 지표는 다음과 같습니다.

- 중단율 - InstanceInterruptions 및 GameSessionInterruptions 지표를 사용하여 인스턴스 및 게임 세션에 대한 스팟 관련 중단의 횟수와 빈도를 추적합니다. TERMINATED에서 회수된 게임 세션은 상태이고 상태 이유는 INTERRUPTED입니다.
- 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.

스팟 플릿의 대기열에 대한 모범 사례

스팟 인스턴스에 대한 플릿 및 대기열을 생성할 때 다음 모범 사례를 사용합니다.

- 대기열의 지리적 범위를 확장합니다. 플레이어가 단일 클러스터링되어 있더라도 스팟 플릿에 인접 위치를 AWS 리전 추가합니다. 이 접근 방식을 통해 리전 속도 저하, 중단, 스팟 중단 중에 용량을 유지하는 대기열의 기능을 개선합니다. 다중 위치 플릿은 스팟 인스턴스와 온디맨드 인스턴스 모두에서 작동합니다.
- 대기열의 인스턴스 유형 범위를 다양화합니다. Amazon GameLift Servers는 인스턴스 유형을 기반으로 스팟 실행 가능성을 평가하므로 다양한 인스턴스 유형의 스팟 플릿을 사용하면 여러 스팟 플릿을 동시에 사용할 수 없을 가능성이 줄일 수 있습니다. 각 위치마다 인스턴스 유형이 다른 스팟 플릿을 두 개 이상 포함합니다.

Note

요금은 플릿 수가 아닌 사용하는 인스턴스를 기준으로 합니다. 10개의 인스턴스를 가진 5개의 플릿을 실행하는 것은 유사한 비용의 인스턴스 50개를 가진 하나의 플릿을 실행하는 것과 같습니다. 요금은 인스턴스 유형, 크기, 위치에 따라 다릅니다.

스팟 인스턴스 유형 그룹화에 대한 팁:

- m6g.medium, m6g.large, m6g.xlarge와 같은 동일한 패밀리의 인스턴스 유형을 사용합니다. 인스턴스 유형이 클수록 비용이 많이 들지만 한 번에 더 많은 게임 세션을 호스팅할 수도 있습니다.
- 널리 사용 가능한 인스턴스 유형을 선택합니다. 일반적으로 이전 세대 패밀리(예: C5, M5, R5)와 일반적인 크기(예: .large, .xlarge, .2xlarge)의 가용성이 향상됩니다.
- Amazon GameLift Servers 콘솔에서 30~90일 요금 내역을 확인합니다. 일관된 가용성 패턴을 가진 인스턴스 유형을 찾습니다.
- 플릿 생성 도구인 Amazon GameLift Servers 콘솔을 사용하여 인스턴스 유형의 위치 범위를 탐색합니다.
- 백업 용량을 위한 온디맨드 플릿을 추가합니다. 게임 호스팅은 스팟 플릿을 사용할 수 없을 때마다 온디맨드 플릿으로 전환할 수 있습니다. 플레이어 지연 시간을 낮게 유지하려면 각 위치에 온디맨드 플릿을 하나 이상 배치합니다. 백업 온디맨드 플릿에 오토 스케일링을 추가하여 필요할 때까지 스케일 다운 상태를 유지할 수 있습니다.
- 모든 플릿 대상에 별칭을 할당합니다. 각 대기열의 대상에 대해 별칭을 생성합니다. 별칭을 사용하면 플릿을 교체해야 할 때마다 더 쉽고 효율적으로 사용할 수 있습니다.
- 대기열 우선순위 지정 전략을 적용합니다. 대기열이 게임 세션을 배치할 위치의 우선순위를 지정하는 방법을 사용자 지정할 수 있습니다(자세한 내용은 [게임 세션 배치 우선순위](#) 참조). 스팟 최적화 대

기열의 경우, 비용을 우선순위로 지정하면 가능할 때마다 언제든지 저렴한 비용의 스팟 플릿이 사용 되도록 보장합니다.

대상 순서를 지정하여 특정 플릿의 우선순위를 지정할 수도 있습니다. 예를 들어 일부 사용자는 정기적으로 사용할 기본 플릿 세트와 보조 플릿 세트를 백업으로 지정합니다. 이 시나리오에서는 기본 플릿을 먼저 나열하도록 대기열의 대상 순서를 설정합니다. 그런 다음 대상과 비용 순으로 대기열의 우선순위 순서를 구성합니다.

리소스 사용자 지정 호스팅

이 섹션에서는 특정 성능, 비용, 운영 요구 사항을 충족하도록 Amazon GameLift Servers 인프라를 구성하고 관리하기 위한 고급 옵션을 제공합니다. 특히 이 섹션의 주제에서는 게임과 플레이어에 가장 적합하도록 Amazon GameLift Servers 관리형 호스팅 리소스를 사용자 지정하는 방법을 설명합니다.

고려해야 할 몇 가지 결정 사항은 다음과 같습니다.

- 플레이어용 호스팅 리소스는 어디에서 배포하나요? 게임 플레이 지연 시간은 플릿의 지리적 위치를 선택하는 주요 요인이지만 위치에 따라 리소스 유형 가용성 및 비용을 포함한 다른 요인들에 의해 달라집니다.
- 게임을 가장 잘 지원할 수 있는 EC2 인스턴스 유형은 무엇인가요? 사용할 수 있는 인스턴스 유형 중에서 선택하여 컴퓨팅 아키텍처, 메모리, 스토리지 및 네트워킹 용량의 최상의 조합을 사용해야 합니다.
- 필요한 인스턴스 유형의 크기: 게임 서버 소프트웨어의 리소스 요구 사항(메모리 및 CPU) 및 기타 요인에 따라 인스턴스 유형 크기를 선택해야 합니다.
- 플릿에서 사용할 인스턴스 유형(온디맨드 또는 스팟): 더 낮은 스팟 요금을 활용할 수 있는지, Amazon GameLift Servers가 게임 세션에 대한 스팟 중단 가능성을 충분히 완화하는지 고려합니다.
- 각 플릿 인스턴스에서 게임 서버 소프트웨어가 실행되는 방법: 런타임 구성으로 Amazon GameLift Servers가 실행할 서버 소프트웨어와 방법을 지정할 수 있습니다.
- 컨테이너 플릿의 경우 기본 구성이 게임에 적합하나요? Amazon GameLift Servers는 컨테이너 플릿 구성을 최적화하도록 많은 작업을 수행하지만 대부분의 구성 설정은 사용자 지정할 수 있습니다.

관리형 플릿의 컴퓨팅 리소스 선택

관리형 EC2 및 관리형 컨테이너를 포함한 AWS 클라우드관리형 호스팅의 경우 서비스는 게임 서버를 Amazon GameLift Servers의 컴퓨팅 리소스 플릿에 배포합니다. 관리형 플릿을 생성할 때 게임에 가장

적합하도록 호스팅 리소스를 구성하고자 합니다. 이 주제에서는 게임 호스팅 플릿을 선택하고 구성할 때의 주요 결정 사항에 대해 설명합니다.

Note

Anywhere 플릿과 Amazon GameLift Servers 관리형 플릿을 모두 사용하여 하이브리드 솔루션을 구축하는 경우 다음 주제를 사용하여 자체 관리형 리소스를 보완하도록 관리형 플릿을 설계합니다. [Amazon GameLift Servers용 호스팅 플릿 배포](#)을(를) 참조하세요.

주제

- [지리적 위치](#)
- [운영 체제](#)
- [인스턴스 유형](#)
- [온디맨드 인스턴스 및 스팟 인스턴스 비교](#)
- [Service Quotas](#)

지리적 위치

게임 서버를 배포할 위치를 고려합니다. 일반적으로 최상의 플레이어 경험을 제공하기 위해 게임 서버를 플레이어와 최대한 가깝게 배치해야 합니다. Amazon GameLift Servers 관리형 호스팅의 경우 지원되는 AWS 리전 및 로컬 영역에 게임 서버를 배치하도록 선택할 수 있습니다. 하이브리드 솔루션을 구축하는 경우 관리형 플릿 배포가 자체 관리형 Amazon GameLift Servers Anywhere 플릿의 위치를 보완할 수 있는 방법을 고려하세요.

대부분의 개발 및 테스트의 경우 단일 위치에 배포하는 것이 좋습니다. 출시 및 그 이후를 준비할 때 많은 이유로 인해 여러 지리적 위치에 배포해야 합니다. 여기에는 광범위한 플레이어 그룹을 지원하고, 전반적인 게임 호스팅 복원력과 신뢰성을 개선하는 것이 포함됩니다. 또한 여러 위치는 게임 세션 배치 속도를 높이고 지연 시간 및 비용을 위해 배치를 최적화할 때 더 많은 선택을 가능하게 하여 플레이어 경험을 향상시킬 수 있습니다.

Amazon GameLift Servers에서 지원하는 위치 목록과 모든 플릿 유형의 위치에 대한 자세한 내용은 [Amazon GameLift Servers 서비스 위치](#) 섹션을 참조하세요.

다중 위치 플릿

단일 관리형 플릿은 여러 위치에 리소스를 배포할 수 있습니다. 다중 위치 플릿에서 각 개별 위치의 용량을 수동으로 설정할 수 있습니다.

다중 위치 플릿 사용의 이점:

- 간소화된 플릿 배포 및 관리 - 게임 서버 소프트웨어 및 플릿 구성을 제공하고 Amazon GameLift Servers가 여러 위치(한 번 빌드, 어디에나 배포)의 플릿 인스턴스에 배포합니다. 프로덕션 플릿에서는 각각 다른 리전에 있는 여러 플릿을 관리할 필요 없이 플릿의 모든 위치를 보고 관리할 수 있습니다.
- 로컬 영역 가용성 - 로컬 영역을 사용하려면 AWS 리전 홈 위치와 로컬 영역을 원격 위치로 사용하여 다중 위치 플릿을 생성해야 합니다. 로컬 영역은 필요한 영역과 고객에게 훨씬 더 짧은 지연 시간을 제공할 AWS 리전 수 있는의 확장입니다. 다중 위치 플릿에 로컬 영역을 추가할 수 있으며, 로컬 영역의 상위 AWS 리전을 포함할 필요는 없습니다.
- 게임 세션 대기열과의 호환성 - 하나 이상의 다중 위치 플릿을 사용하여 게임 세션 배치 대기열을 구축할 수 있습니다. 이 접근 방식은 새 게임 세션을 호스팅할 위치의 우선순위를 지정하고 선택할 때 대기열에 유연성을 제공합니다.
- 효율적인 리소스 사용률 - Auto Scaling이 켜져 있으면 Amazon GameLift Servers는 플릿의 모든 위치에서 용량 조정을 더 잘 최적화할 수 있습니다.

다중 위치 플릿 사용에 대한 팁:

- AWS 리전 또는 플릿당 위치 수에 대한 할당량을 확인합니다. [Amazon GameLift Servers 서비스 할당량](#)을 참조하세요.
- 모든 위치에서 모든 인스턴스 유형을 사용할 수 있는 것은 아닙니다. 선택한 위치에 따라 인스턴스 유형 옵션이 제한될 수 있습니다. Amazon GameLift Servers 콘솔은 위치와 인스턴스 유형의 적절한 균형을 찾는 데 도움이 되는 유용한 도구를 제공합니다.
- [UDP 핑 비콘](#)을 사용하여 모든 플릿 위치에 대한 플레이어 지연 시간 데이터를 수집하는 것이 좋습니다. Amazon GameLift Servers는 이 데이터를 사용하여 짧은 지연 시간으로 게임 세션을 최적화하고 플레이어가 허용할 수 없을 정도의 높은 지연 시간으로 세션에 참여하지 못하도록 할 수 있습니다. 이러한 특수 엔드포인트는 기존 ICMP 핑 대신 UDP 메시지를 수락하여 정확한 지연 시간 측정값을 제공하므로 최적의 플릿 위치를 선택하는 데 도움이 됩니다.

운영 체제

관리형 플릿의 모든 인스턴스는 게임 서버 소프트웨어를 위한 완전한 런타임 환경을 제공하는 Amazon Machine Image(AMI)와 함께 배포됩니다. 관리형 EC2 플릿의 경우 빌드를 Amazon GameLift Servers에 업로드할 때 게임 서버 빌드의 운영 체제를 지정합니다. 관리형 컨테이너 플릿의 경우 컨테이너 그룹 정의에서 운영 체제를 지정합니다. 최신 AMI 버전에 대한 자세한 내용은 [Amazon GameLift Servers AMI 버전](#) 섹션을 참조하세요.

AMI 버전은 정기적으로 업데이트됩니다. 새 플릿을 생성할 때 Amazon GameLift Servers는 게임 빌드에 대해 선택한 AMI의 사용 가능한 최신 버전을 할당합니다. 해당 플릿에 배포된 모든 인스턴스는 동일한 버전을 사용합니다. 최신 보안 및 소프트웨어 업데이트로 AMI 버전을 최신 상태로 유지하려면 플릿을 정기적으로 교체해야 합니다. 모범 사례에 따라, 게임 서버의 런타임 환경을 유지하기 위해 30일마다 관리형 플릿을 교체하는 것이 좋습니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#)를 참조하세요.

인스턴스 유형

관리형 플릿의 인스턴스 유형은 모든 플릿 인스턴스에 배포되는 하드웨어의 종류를 결정하며, 인스턴스 유형은 일반적으로 다양한 크기로 제공됩니다. 모든 Amazon GameLift Servers 관리형 플릿은 Amazon EC2 인스턴스를 사용하며 컴퓨팅 성능, 메모리, 스토리지, 네트워킹 기능의 다양한 조합을 제공하는 다양한 인스턴스 유형을 지원합니다. 인스턴스 유형의 가용성은 선택한 위치에 따라 다릅니다.

Amazon GameLift Servers 콘솔은 게임 빌드와 개발 위치에 대한 적절한 인스턴스 유형을 찾는 데 도움이 되는 유용한 도구를 제공합니다. 관리형 컨테이너 플릿의 경우 콘솔은 게임의 CPU 성능 및 메모리 요구 사항에 대한 지침도 제공합니다.

게임에 사용할 수 있는 인스턴스 유형을 선택할 때는 다음 사항을 고려합니다.

- 게임 서버의 컴퓨팅 아키텍처: x64 또는 Arm(AWS Graviton).

Note

Graviton Arm 인스턴스의 경우 Linux AMI에서 서버 빌드를 사용해야 합니다. 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.

- 게임 서버 빌드의 컴퓨팅, 메모리 및 스토리지 요구 사항입니다.
- 인스턴스 유형의 크기입니다. 게임 서버 소프트웨어 실행 파일의 요구 사항을 충족하는 것 외에도 인스턴스 유형 크기가 클수록 각 인스턴스에서 여러 게임 서버 프로세스 및/또는 컨테이너를 실행할 수 있습니다. 고려해야 할 요소에는 비용이 포함됩니다(몇 개의 큰 인스턴스 또는 여러 개의 작은 인스턴스를 실행하는 것이 더 저렴함). 또한 플릿 조정 이벤트 중에 또는 비정상 인스턴스를 종료할 때 인스턴스를 추가하거나 제거하여 게임 세션 용량에 어떤 영향을 미칠 수 있는지 고려합니다. 각 인스턴스가 여러 게임 서버 프로세스를 동시에 실행하는 경우 인스턴스를 추가하거나 제거하면 게임 호스팅 용량에 상당한 영향을 미칠 수 있습니다.

인스턴스 유형에 대한 자세한 내용은 [Amazon EC2 인스턴스 유형](#)을 참조하십시오.

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 지불한다는 걸 의미입니다. 장기 약정을 적용하지 않습니다.

스팟 인스턴스

스팟 인스턴스는 미사용 AWS 컴퓨팅 용량을 활용하여 온디맨드 인스턴스에 대한 비용 효율적인 대안을 제공할 수 있습니다. 스팟 인스턴스 요금은 각 위치의 각 인스턴스 유형에 대한 공급 및 수요에 따라 달라집니다. 용량이 다시 필요할 때마다 2분 알림으로 스팟 인스턴스를 회수할 AWS 수 있으며 회수된 인스턴스에서 적극적으로 실행되는 게임 세션이 중단됩니다.

Amazon GameLift Servers는 게임 세션에 대한 스팟 중단 가능성을 완화하는 데 도움이 되는 몇 가지 도구를 제공합니다. 스팟 실행 가능성 알고리즘은 인스턴스 유형 기록 데이터를 추적하여 중단 위험이 중요 지점에 도달하는 시기를 예측하고 해당 유형의 스팟 인스턴스에 새 게임 세션을 배치하는 것을 방지합니다. 중단이 발생할 경우 게임 서버는 알림을 사용하여 플레이어의 게임 세션을 정상적으로 종료할 수 있습니다.

스팟 플릿을 사용한 게임 호스팅은 게임 세션 배치에 대기열을 사용해야 합니다. 대기열은 스팟 플릿 실행 가능성, 비용, 기타 요인을 기반으로 게임 세션 배치의 우선순위를 지정할 수 있습니다. 게임 서버 호스팅에서 스팟을 활용하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- [스팟 플릿으로 게임 호스팅 비용 절감](#)
- [스팟 인스턴스를 위한 대기열 빌드](#)

Service Quotas

다음 도구를 사용하여 Amazon GameLift Servers에 대한 기본 서비스 할당량과 AWS 계정 에 대한 현재 할당량 상태를 볼 수 있습니다.

- Amazon GameLift Servers에 대한 일반 서비스 할당량 정보는 AWS 일반 참조의 [Amazon GameLift Servers 엔드포인트 및 할당량](#)을 참조하세요.

- 계정에 대한 위치별 사용 가능한 인스턴스 유형 목록을 보려면 Amazon GameLift Servers 콘솔의 [Service Quotas](#) 페이지를 엽니다. 또한 이 페이지에는 각 위치의 각 인스턴스 유형에 대한 계정의 현재 사용량도 표시됩니다.
- 리전별 인스턴스 유형에 대한 계정의 현재 할당량 목록을 보려면 AWS Command Line Interface (AWS CLI) 명령을 실행합니다 [describe-ec2-instance-limits](#). 이 명령은 기본 리전(또는 지정한 다른 리전)에 있는 활성 인스턴스 수를 반환합니다.

게임을 시작할 준비를 할 때 [Amazon GameLift Servers 콘솔](#)에서 시작 설문지를 작성합니다. Amazon GameLift Servers 팀은 출시 설문지를 사용하여 게임에 대한 올바른 할당량 및 한도를 결정합니다.

Amazon GameLift Servers 컨테이너 플릿 사용자 지정

이 섹션의 주제에서는 Amazon GameLift Servers 관리형 컨테이너의 몇 가지 선택적 기능에 대해 설명합니다. 이러한 기능 중 일부 또는 전부를 사용하도록 설정할 수 있습니다.

주제

- [리소스 제한 설정](#)
- [컨테이너 플릿 메모리 할당 이해](#)
- [NVMe 드라이브 액세스 구성](#)
- [필수 컨테이너 지정](#)
- [네트워크 연결 구성](#)
- [컨테이너에 대한 상태 확인 설정](#)
- [컨테이너 종속성 설정](#)
- [컨테이너 플릿 구성](#)

리소스 제한 설정

각 컨테이너 그룹에 대해 컨테이너 그룹이 소프트웨어를 실행하는 데 필요한 메모리 및 컴퓨팅 성능을 확인할 수 있습니다. Amazon GameLift Servers는 이 정보를 사용하여 컨테이너 그룹 전반의 리소스를 관리합니다. 또한 이 정보를 사용하여 플릿 이미지가 보유할 수 있는 게임 서버 컨테이너 그룹 수를 계산합니다. 개별 컨테이너에 대한 제한을 설정할 수도 있습니다.

컨테이너 그룹의 메모리 및 컴퓨팅 성능에 대한 최대 제한을 설정할 수 있습니다. 기본적으로 이러한 리소스는 그룹의 모든 컨테이너에서 공유됩니다. 개별 컨테이너에 대한 제한을 설정하여 리소스 관리를 추가로 사용자 지정할 수 있습니다.

개별 컨테이너에 대한 선택적 제한 설정

컨테이너별 리소스 제한을 설정하면 개별 컨테이너가 그룹의 리소스를 사용할 수 있는 방법을 더 잘 제어할 수 있습니다. 컨테이너별 제한을 설정하지 않으면 그룹의 모든 컨테이너가 그룹 리소스를 공유합니다. 공유를 통해 필요한 곳에 리소스를 사용할 수 있는 유연성이 향상됩니다. 또한 프로세스가 서로 경쟁하여 컨테이너 장애가 발생할 가능성도 높입니다.

컨테이너에 대해 다음 ContainerDefinition 속성 중 하나를 설정합니다.

- **MemoryHardLimitMebibytes** - 컨테이너의 최대 메모리 제한을 설정합니다. 컨테이너가 이 제한을 초과하면 다시 시작됩니다.
- **Vcpu 제한** - 컨테이너의 독점적인 사용을 위해 최소량의 vCPU 리소스를 예약합니다. 컨테이너는 항상 예약 용량을 사용할 수 있습니다. 추가 리소스를 사용할 수 있는 경우 언제든지 이 최소값을 초과할 수 있습니다. (1024 CPU 단위는 vCPU 1개에 해당합니다.)

컨테이너 그룹에 대한 총 리소스 제한 설정

개별 컨테이너에 대한 제한을 설정하는 경우 컨테이너 그룹에 필요한 메모리 및 vCPU 리소스의 양을 수정해야 할 수 있습니다. 목표는 게임 서버 성능을 최적화하기 위해 충분한 리소스를 할당하는 것입니다. Amazon GameLift Servers는 이러한 제한을 사용하여 플릿 인스턴스에 게임 서버 컨테이너 그룹을 패키징하는 방법을 계산합니다. 컨테이너 플릿의 인스턴스 유형을 선택할 때 사용할 수도 있습니다.

컨테이너 그룹에 필요한 총 메모리 및 vCPU를 계산합니다. 다음을 고려하세요.

- 컨테이너 그룹의 모든 컨테이너에서 실행되는 모든 프로세스는 무엇인가요? 이러한 프로세스에 필요한 리소스를 추가합니다. 컨테이너별 제한을 기록해 둡니다.
- 각 컨테이너 그룹에서 실행할 동시 게임 서버 프로세스는 몇 개인가요? 게임 서버 컨테이너 이미지에서 이를 결정합니다.

컨테이너 그룹 요구 사항 추정치에 따라 다음 ContainerGroupDefinition 속성을 설정합니다.

- **TotalMemoryLimitMebibytes** - 컨테이너 그룹의 최대 메모리 제한을 설정합니다. 그룹의 모든 컨테이너는 할당된 메모리를 공유합니다. 개별 컨테이너 제한을 설정하는 경우 총 메모리 제한은 가장 높은 컨테이너별 메모리 제한보다 크거나 같아야 합니다.
- **TotalVcpuLimit** - 컨테이너 그룹의 최대 vCPU 제한을 설정합니다. 그룹의 모든 컨테이너는 할당된 CPU 리소스를 공유합니다. 개별 컨테이너 제한을 설정하는 경우 총 CPU 제한은 모든 컨테이너별 CPU 제한 합계보다 크거나 같아야 합니다. 컨테이너 CPU 제한 합계의 두 배로 이 값을 설정하는 것이 좋습니다.

예제 시나리오

다음 세 개의 컨테이너로 게임 서버 컨테이너 그룹을 정의한다고 가정해 보겠습니다.

- 컨테이너 A는 게임 서버 컨테이너입니다. 하나의 게임 서버에 대한 리소스 요구 사항은 512MiB 및 1024CPU로 추정됩니다. 컨테이너가 1개의 서버 프로세스를 실행하도록 할 계획입니다. 이 컨테이너는 가장 중요한 소프트웨어를 실행하므로 메모리 제한이나 vCPU 예약 제한을 설정하지 않습니다.
- 컨테이너 B는 1024MiB 및 1536CPU로 예상되는 리소스 요구 사항이 있는 지원 컨테이너입니다. 메모리 제한을 2048MiB로 설정하고 CPU 예약 제한을 1024CPU로 설정합니다.
- 컨테이너 C는 다른 지원 컨테이너입니다. 메모리 제한을 512MiB로 설정하고 CPU 예약 제한을 512CPU로 설정합니다.

이 정보를 사용하여 컨테이너 그룹에 대해 다음과 같은 총 제한을 설정합니다.

- 총 메모리 제한: 7680MiB. 이 값은 가장 높은 메모리 제한(1024MiB)을 초과합니다.
- 총 CPU 한도: 13312CPU. 이 값은 CPU 제한(1024+512CPU)의 합계를 초과합니다.

컨테이너 플릿 메모리 할당 이해

가 플릿 인스턴스에 컨테이너 그룹을 Amazon GameLift Servers 배포하는 경우 모든 인스턴스 메모리를 컨테이너에 사용할 수 있는 것은 아닙니다.는 운영 체제, Amazon ECS 에이전트 및 기타 지원 서비스에 대해 인스턴스 메모리의 일부를 Amazon GameLift Servers 예약합니다. 예약 메모리의 양은 인스턴스 유형의 총 메모리에 따라 달라집니다. 이 오버헤드를 이해하면 사용 가능한 리소스를 완전히 활용하도록 컨테이너 그룹 정의를 구성할 수 있습니다.

메모리 오버헤드 공식

Amazon GameLift Servers는 다음 단계를 사용하여 컨테이너 그룹에 사용할 수 있는 메모리를 계산합니다.

1. 메모리 버퍼 백분율을 결정합니다.는 다음 계층을 기반으로 인스턴스 총 메모리의 백분율을 Amazon GameLift Servers 예약합니다.

인스턴스 메모리(MiB)	예약 비율
5,000 미만	8%
5,000~9,999	6%
10,000~89,999	5%
90,000~199,999	4%

인스턴스 메모리(MiB)	예약 비율
200,000 이상	3%

2. 사용 가능한 메모리를 계산합니다. 총 인스턴스 메모리에서 예약된 메모리를 뺍니다.

$$\text{AvailableMemory} = \text{InstanceMemory} - \text{round}(\text{InstanceMemory} \times \text{BufferPercentage})$$

3. 인스턴스당 컨테이너 그룹 메모리를 뺍니다. 플릿이 인스턴스당 컨테이너 그룹을 사용하는 경우 사용 가능한 메모리 `TotalMemoryLimitMebibytes`에서 해당 그룹을 뺍니다. 인스턴스당 하나의 컨테이너 그룹이 각 플릿 인스턴스에서 실행됩니다.

$$\text{AvailableMemory} = \text{AvailableMemory} - \text{PerInstanceCGD.TotalMemoryLimitMebibytes}$$

4. 로그 라우터 오버헤드에 대한 계정입니다. 플릿에 로깅이 활성화된 경우는 로그 라우터에 대해 게임 서버 컨테이너 그룹당 50MiB를 추가로 Amazon GameLift Servers 예약합니다.
5. 최대 게임 서버 컨테이너 그룹을 계산합니다. 메모리별로 인스턴스에 맞는 게임 서버 컨테이너 그룹의 최대 수는 다음과 같습니다.

$$\text{MaxGroupsByMemory} = \text{floor}(\text{AvailableMemory} / (\text{GameServerCGD.TotalMemoryLimitMebibytes} + \text{LogRouterMemory}))$$

여기서 `LogRouterMemory`는 로깅이 활성화된 경우 50MiB이고 로깅이 비활성화된 경우 0입니다.

Note

메모리는 인스턴스에 맞는 게임 서버 컨테이너 그룹 수를 결정하는 한 가지 요소일 뿐입니다. vCPU 용량과 사용 가능한 연결 포트 Amazon GameLift Servers도 고려하고 세 가지 계산 모두의 최소값을 사용합니다.

메모리 계산 예

로깅이 활성화된 c5.xlarge 인스턴스(총 메모리 8,192MiB)를 사용하는 플릿을 생각해 보십시오.

- 인스턴스 메모리는 8,192MiB이며 5,000~9,999 티어(6% 버퍼)에 속합니다.
- 예약 메모리 = $\text{round}(8,192 \times 0.06) = 492\text{MiB}$
- 사용 가능한 메모리 = $8,192 - 492 = 7,700\text{MiB}$

4. TotalMemoryLimitMebibytes 512의 인스턴스당 컨테이너 그룹을 사용하는 경우: 사용 가능한 메모리 = $7,700 - 512 = 7,188\text{MiB}$
5. 각 게임 서버 컨테이너 그룹의 값이 TotalMemoryLimitMebibytes 1,024인 경우:
 $\text{MaxGroupsByMemory} = \text{floor}(7,188 / (1,024 + 50)) = \text{floor}(7,188 / 1,074) = 6$

인스턴스 유형별 사용 가능한 메모리

다음 표에는 일반적으로 사용되는 인스턴스 유형에 대한 총 메모리와 사용 가능한 메모리(Amazon GameLift Servers버퍼 이후)가 나와 있습니다. 컨테이너 그룹 정의를 구성할 때 이러한 값을 시작점으로 사용합니다. 사용 가능한 메모리 열에는 인스턴스당 컨테이너 그룹 또는 로그 라우터 오버헤드를 빼기 전에 인스턴스의 모든 컨테이너 그룹에 사용할 수 있는 메모리가 표시됩니다.

인스턴스 유형	총 메모리(MiB)	버퍼 백분율	사용 가능한 메모리(MiB)
c5.large	4,096	8%	3,768
c5.xlarge	8,192	6%	7,700
c5.2xlarge	16,384	5%	15,565
c5.4xlarge	32,768	5%	31,130
c5.9xlarge	73,728	5%	70,042
c5.12xlarge	98,304	4%	94,372
c5.18xlarge	147,456	4%	141,558
c5.24xlarge	196,608	4%	188,744
m5.large	8,192	6%	7,700
m5.xlarge	16,384	5%	15,565
m5.2xlarge	32,768	5%	31,130
m5.4xlarge	65,536	5%	62,259
m5.8xlarge	131,072	4%	125,829

인스턴스 유형	총 메모리(MiB)	버퍼 백분율	사용 가능한 메모리(MiB)
m5.12xlarge	196,608	4%	188,744
r5.large	16,384	5%	15,565
r5.xlarge	32,768	5%	31,130
r5.2xlarge	65,536	5%	62,259
r5.4xlarge	131,072	4%	125,829
c6i.large	4,096	8%	3,768
c6i.xlarge	8,192	6%	7,700
c6i.2xlarge	16,384	5%	15,565
c6i.4xlarge	32,768	5%	31,130
c6i.8xlarge	65,536	5%	62,259
c7i.large	4,096	8%	3,768
c7i.xlarge	8,192	6%	7,700
c7i.2xlarge	16,384	5%	15,565
c7i.4xlarge	32,768	5%	31,130
c7i.8xlarge	65,536	5%	62,259
m7i.large	8,192	6%	7,700
m7i.xlarge	16,384	5%	15,565
m7i.2xlarge	32,768	5%	31,130
m7i.4xlarge	65,536	5%	62,259
m7i.8xlarge	131,072	4%	125,829

인스턴스 유형	총 메모리(MiB)	버퍼 백분율	사용 가능한 메모리(MiB)
m7i.12xlarge	196,608	4%	188,744
r7i.large	16,384	5%	15,565
r7i.xlarge	32,768	5%	31,130
r7i.2xlarge	65,536	5%	62,259
r7i.4xlarge	131,072	4%	125,829

여기에 나열되지 않은 인스턴스 유형의 경우 위에서 설명한 공식을 사용하여 사용 가능한 메모리를 계산할 수 있습니다. [Amazon EC2 인스턴스 유형 설명서](#)에서 선택한 인스턴스 유형의 총 메모리를 확인합니다.

NVMe 드라이브 액세스 구성

d 유형 인스턴스에서 NVMe 드라이브는 호스트 시작 중에 /data 디렉터리에 자동으로 마운트됩니다. 컨테이너가 SSD 스토리지에 액세스할 수 있도록 하려면 MountPoints다음 ContainerGroupDefinition 속성을 설정합니다.

- InstancePath - 호스트 인스턴스에서 자동 탑재된 NVMe 드라이브를 참조/data하려면 로 설정합니다.
- AccessLevel - 컨테이너의 요구 사항에 적합한 액세스 수준을 선택합니다(예: READ_ONLY 또는 READ_WRITE).
- ContainerPath - (선택 사항) 컨테이너 내에 인스턴스 경로를 탑재할 경로를 지정합니다. 지정하지 않으면 기본적으로 인스턴스 경로로 설정됩니다.

탑재 지점에 대한 자세한 내용은 Amazon GameLift Servers API 참조의 [ContainerMountPoint](#)를 참조하세요.

필수 컨테이너 지정

인스턴스당 컨테이너 그룹의 경우 각 컨테이너를 필수 또는 비필수로 지정합니다. 인스턴스당 컨테이너 그룹에는 하나 이상의 필수 지원 컨테이너가 있어야 합니다. 필수 컨테이너는 컨테이너 그룹의 중요

한 작업을 수행합니다. 필수 컨테이너는 항상 실행 중일 것으로 예상됩니다. 실패하면 전체 컨테이너 그룹이 다시 시작됩니다.

각 컨테이너에 대해 ContainerDefinition 속성 Essential을 true 또는 false로 설정합니다.

네트워크 연결 구성

외부 트래픽이 컨테이너 플릿의 모든 컨테이너에 연결되도록 네트워크 액세스를 사용자 지정할 수 있습니다. 예를 들어 게임 서버 프로세스를 실행하는 컨테이너에 네트워크 연결을 반드시 설정해야 게임 클라이언트가 게임에 참여하고 플레이할 수 있습니다. 게임 클라이언트는 포트와 IP 주소를 사용하여 게임 서버에 연결합니다.

컨테이너 플릿에서는 클라이언트와 서버 간의 연결이 직접적이지 않습니다. 내부적으로 컨테이너의 프로세스는 컨테이너 포트에서 수신 대기합니다. 외부적으로 수신 트래픽은 연결 포트를 사용하여 플릿 인스턴스에 연결됩니다. Amazon GameLift Servers는 내부 컨테이너 포트와 외부 연결 포트 간의 매핑을 유지하여 수신 트래픽이 인스턴스의 올바른 프로세스로 라우팅되도록 합니다.

Amazon GameLift Servers는 네트워크 연결을 위한 추가 제어 계층을 제공합니다. 각 컨테이너 플릿에는 각 외부 연결 포트에 대한 액세스를 제어할 수 있는 인바운드 권한 설정이 있습니다. 예를 들어 모든 연결 포트에 대한 권한을 제거하여 플릿의 컨테이너에 대한 모든 액세스를 종료할 수 있습니다.

플릿의 인바운드 권한, 연결 포트 및 컨테이너 포트를 업데이트할 수 있습니다.

Warning

사용자 지정 InstanceConnectionPortRange 또는 InstanceInboundPermissions 속성을 제공하는 경우 Amazon GameLift Servers는 더 이상 플릿의 두 값 중 어느 것도 관리하지 않습니다. 정의되지 않은 동작을 방지하려면 두 필드를 모두 설정해야 합니다.

컨테이너 포트 범위 설정

각 컨테이너 정의의 일부로 컨테이너 포트 범위를 구성합니다. 이는 컨테이너 그룹 정의에 필요한 파라미터입니다. 외부 액세스가 필요한 동시에 실행되는 모든 프로세스를 수용할 수 있도록 충분한 포트를 구성해야 합니다. 일부 컨테이너에는 포트가 필요하지 않습니다.

게임 서버를 실행하는 게임 서버 컨테이너에는 동시에 실행되는 모든 게임 서버 프로세스에 대한 포트가 필요합니다. 게임 서버 프로세스는 할당된 포트를 수신 대기하고 이를 Amazon GameLift Servers에 보고합니다.

연결 포트 범위 설정

연결 포트 세트를 사용하여 컨테이너 플릿을 구성합니다. 연결 포트는 컨테이너를 실행하는 플릿 인스턴스에 대한 외부 액세스를 제공합니다. Amazon GameLift Servers는 연결 포트를 할당하고 필요에 따라 컨테이너 포트에 매핑합니다.

기본적으로 Amazon GameLift Servers는 모든 컨테이너 그룹에 필요한 포트 수를 계산하고 이를 수용하도록 포트 범위를 설정합니다. 컨테이너 그룹 정의에 업데이트를 배포할 때 업데이트되는 Amazon GameLift Servers 계산값을 사용하는 것이 가장 좋습니다. 연결 포트 범위를 사용자 지정해야 하는 경우 다음 지침을 사용합니다.

컨테이너 플릿을 생성할 때 연결 포트 범위를 정의합니다([ContainerFleet:InstanceConnectionPortRange](#) 참조). 범위에 플릿의 두 컨테이너 그룹에 있는 모든

컨테이너에 정의된 모든 컨테이너 포트에 매핑할 수 있는 충분한 포트가 있는지 확인합니다. 필요한 최소 연결 포트를 계산하려면 다음 공식을 사용합니다.

[Total number of container ports defined for containers in the game server container group] * [Number of game server container groups per instance] + [Total number of container ports defined for containers in the per-instance container group]

가장 좋은 방법은 최소 연결 포트 수를 두 배로 늘리는 것입니다.

Note

연결 포트 수는 인스턴스당 게임 서버 컨테이너 그룹 수를 잠재적으로 제한할 수 있습니다. 플릿에 인스턴스당 하나의 게임 서버 컨테이너 그룹에 대한 연결 포트만 있는 경우 인스턴스에 여러 게임 서버 컨테이너 그룹에 대한 컴퓨팅 파워가 충분하더라도 Amazon GameLift Servers는 하나의 게임 서버 컨테이너 그룹만 배포합니다.

인바운드 권한 설정

인바운드 권한은 수신 트래픽에 대해 열어둘 연결 포트를 지정하여 컨테이너 플릿에 대한 외부 액세스를 제어합니다. 이 설정을 사용하여 필요에 따라 플릿의 네트워크 액세스를 켜거나 끌 수 있습니다.

기본적으로 Amazon GameLift Servers는 모든 컨테이너 그룹에 필요한 포트 수를 계산하고 이를 수용하도록 포트 범위를 설정합니다. 컨테이너 그룹 정의에 업데이트를 배포할 때 업데이트되는

Amazon GameLift Servers 계산값을 사용하는 것이 가장 좋습니다. 연결 포트 범위를 사용자 지정해야 하는 경우 다음 지침을 사용합니다.

컨테이너 플릿을 생성할 때 인바운드 권한 세트를 정의합니다(

[ContainerFleet:InstanceInboundPermissions](#) 참조). 인바운드 권한 포트는 플릿의 연결 포트 범위와 일치해야 합니다.

Note

컨테이너 포트는 InstanceConnectionPortRange에서 무작위로 선택되므로 세션 연결을 보장하려면 InstanceConnectionPortRange의 모든 포트가 InstanceInboundPermissions에 적용되어야 합니다.

예제 시나리오

이 예제에서는 세 네트워크 연결 속성을 모두 설정하는 방법을 보여줍니다.

- 플릿의 게임 서버 컨테이너 그룹에는 게임 서버 프로세스 1개를 실행하는 컨테이너 1개가 있습니다.

게임 서버 컨테이너 그룹 정의에서 이 컨테이너의 PortConfiguration 파라미터를 다음과 같이 설정합니다.

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP"} ]
}
```

- 또한 플릿에는 컨테이너가 1개인 인스턴스당 컨테이너 그룹도 있습니다. 여기에는 네트워크 액세스가 필요한 프로세스가 1개 있습니다. 인스턴스당 컨테이너 정의에서 이 컨테이너의 PortConfiguration 파라미터를 다음과 같이 설정합니다.

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ] }
```

- 플릿 인스턴스당 20개의 게임 서버 컨테이너 그룹으로 플릿이 구성됩니다. 이 정보를 고려할 때 공식을 사용하여 필요한 연결 포트 수를 계산할 수 있습니다.
 - 최소: 포트 포트 21개[게임 서버 컨테이너 포트 1개 * 인스턴스당 게임 서버 컨테이너 그룹 20개 + 인스턴스당 컨테이너 포트 1개]
 - 모범 사례: 포트 42개[최소 포트 * 2]

컨테이너 플릿을 생성할 때 InstanceConnectionPortRange 파라미터를 다음과 같이 설정합니다.

```
"InstanceConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- 사용 가능한 모든 연결 포트에 대한 액세스를 허용하려고 합니다. 컨테이너 플릿을 생성할 때 InstanceInboundPermissions 파라미터를 다음과 같이 설정합니다.

```
"InstanceInboundPermissions": [
  { "FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":
    "TCP" } ]
```

컨테이너에 대한 상태 확인 설정

컨테이너는 터미널 장애가 발생하여 실행을 중지하면 자동으로 다시 시작됩니다. 컨테이너가 필수로 간주되는 경우 전체 컨테이너 그룹을 다시 시작하라는 메시지가 표시됩니다.

모든 게임 서버 컨테이너는 자동으로 필수로 처리됩니다. 지원 컨테이너는 필수로 지정할 수 있지만 상태를 보고하는 메커니즘이 있어야 합니다. 비필수 지원 컨테이너에 대한 상태 확인도 설정할 수 있습니다.

추가 사용자 지정 기준을 정의하여 컨테이너 상태를 측정하고 상태 확인을 사용하여 해당 기준을 테스트할 수 있습니다. 컨테이너 상태 확인을 설정하려면 Docker 컨테이너 이미지 또는 컨테이너 정의에서 정의할 수 있습니다. 컨테이너 정의에서 상태 확인을 설정하면 컨테이너 이미지의 모든 설정이 재정의됩니다.

컨테이너 상태 확인에 대해 다음 SupportContainerDefinition 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 확인 실패가 계산을 시작하기 전에 초기 지연을 지정합니다. 이 지연은 컨테이너가 프로세스를 부트스트랩할 시간을 제공합니다.
- **Interval** - 상태 확인 명령을 실행할 빈도를 결정합니다. 컨테이너 장애를 얼마나 빨리 감지하고 해결하려고 하나요?
- **Timeout** - 상태 확인 명령을 다시 시도하기 전에 성공 또는 실패를 기다릴 시간을 결정합니다. 상태 확인 명령을 완료하는 데 시간이 얼마나 소요되나요?
- **Retries** - 실패를 등록하기 전에 상태 확인 명령을 몇 번 재시도해야 하나요?

컨테이너 종속성 설정

각 컨테이너 그룹 내에서 컨테이너 상태에 따라 컨테이너 간에 종속성을 설정할 수 있습니다. 종속성은 종속 컨테이너가 다른 컨테이너의 상태에 따라 시작하거나 종료할 수 있는 경우에 영향을 줍니다.

종속성의 주요 사용 사례는 컨테이너 그룹에 대한 시작 및 종료 시퀀스를 생성하는 것입니다.

예를 들어 컨테이너 B와 C가 시작되기 전에 컨테이너 A를 먼저 시작하고 성공적으로 완료하도록 할 수 있습니다. 이렇게 하려면 먼저 컨테이너 A가 성공적으로 완료되어야 하는 조건으로 컨테이너 A에서 컨테이너 B에 대한 종속성을 생성합니다. 그런 다음 동일한 조건으로 컨테이너 A에서 컨테이너 C에 대한 종속성을 생성합니다. 시작 시퀀스는 종료의 역순으로 실행됩니다.

컨테이너 플릿 구성

컨테이너 플릿을 생성할 때 다음 결정 사항을 고려하세요. 대부분의 이러한 지점은 컨테이너 아키텍처 및 구성에 따라 달라집니다.

플릿을 배포할 위치 결정

일반적으로 플레이어 근처에 지리적으로 플릿을 배포하여 지연 시간을 최소화하려고 합니다. 가 Amazon GameLift Servers 지원하에 컨테이너 플릿 AWS 리전 을 배포할 수 있습니다. 동일한 게임 서버를 추가 지리적 위치에 배포하려는 경우 AWS 리전 및 로컬 영역을 포함한 원격 위치를 플릿에 추가할 수 있습니다. 다중 위치 플릿의 경우 각 플릿 위치에서 독립적으로 용량을 조정할 수 있습니다. 지원되는 플릿 위치에 대한 자세한 내용은 [Amazon GameLift Servers 서비스 위치](#) 섹션을 참조하세요.

[UDP 핑 비콘](#)을 사용하여 다양한 지리적 위치에서 네트워크 지연 시간 데이터를 수집하여 플레이어 디바이스와 잠재적 플릿 위치 간의 지연 시간을 예측할 수 있습니다. 이러한 특수 엔드포인트는 기존 ICMP 핑 대신 UDP 메시지를 수락하여 정확한 지연 시간 측정값을 제공하므로 최적의 플릿 위치를 선택하는 데 도움이 됩니다.

플릿의 인스턴스 유형 및 크기 선택

Amazon GameLift Servers는 다양한 Amazon EC2 인스턴스 유형을 지원하며, 모두 컨테이너 플릿과 함께 사용할 수 있습니다. 인스턴스 유형 가용성 및 가격은 위치에 따라 다릅니다. Amazon GameLift Servers 콘솔(리소스, 인스턴스 및 서비스 할당량 아래)에서 위치별로 필터링된 지원되는 인스턴스 유형 목록을 볼 수 있습니다.

인스턴스 유형을 선택할 때는 먼저 인스턴스 패밀리를 고려하세요. 인스턴스 패밀리는 CPU, 메모리, 스토리지, 네트워킹 용량의 다양한 조합을 제공합니다. [EC2 인스턴스 패밀리](#)에서 자세한 정보를 확인하세요. 각 패밀리에는 선택할 수 있는 다양한 인스턴스 크기가 있습니다. 인스턴스 크기를 선택할 때 다음 문제를 고려하세요.

- 워크로드를 지원할 수 있는 최소 인스턴스 크기는 무엇인가요? 이 정보를 사용하여 너무 작은 인스턴스 유형을 제거합니다.
- 컨테이너 아키텍처에 적합한 인스턴스 유형 크기는 무엇인가요? 낭비되는 공간을 최소화하면서 게임 서버 컨테이너 그룹의 여러 사본을 수용할 수 있는 크기를 선택하는 것이 가장 좋습니다.
- 게임에 적합한 세분화된 규모 조정은 무엇인가요? 플릿 용량 규모 조정에는 인스턴스 추가 또는 제거가 포함되며, 각 인스턴스는 특정 수의 게임 세션을 호스팅하는 기능을 나타냅니다. 각 인스턴스에 추가하거나 제거할 용량의 양을 고려합니다. 플레이어 수요가 분당 수천 개씩 다를 경우 수백 또는 수천 개의 게임 세션을 호스팅할 수 있는 매우 큰 인스턴스를 사용하는 것이 합리적일 수 있습니다. 반면, 더 작은 인스턴스 유형으로 더 세분화된 규모 조정 제어를 선호할 수 있습니다.
- 크기에 따라 비용을 절감할 수 있나요? 가용성으로 인해 특정 인스턴스 유형의 비용이 위치에 따라 다를 수 있습니다.

기타 선택적 플릿 설정 구성

컨테이너 플릿을 구성할 때 다음과 같은 선택적 기능을 사용할 수 있습니다.

- 다른 AWS 리소스에 액세스하도록 게임 서버를 설정합니다. [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결\(를\)](#) 참조하세요.
- 스케일 다운 이벤트 중에 활성 플레이어가 있는 게임 세션이 조기에 종료되지 않도록 보호합니다.
- 제한된 시간 내에 한 개인이 플릿에서 생성할 수 있는 게임 세션 수를 제한합니다.

스팟 플릿으로 게임 호스팅 비용 절감

Amazon GameLift Servers 관리형 호스팅을 사용하여 멀티플레이어 게임 서버를 호스팅할 때 스팟 인스턴스는 온디맨드 인스턴스에 대한 비용 효율적인 대안을 제공할 수 있습니다. 스팟 요금 모델은 온디맨드와 동일한 하드웨어 및 성능을 제공하지만 잠재적으로 상당한 비용 절감(최대 70~90%)을 제공합니다. 그러나 제한 사항이 있습니다.에 용량이 다시 AWS 필요한 경우 2분 중단 알림으로 이러한 인스턴스를 회수할 수 있습니다.

Amazon GameLift Servers는 게임 서버 호스팅 중단 위험을 완화합니다. Amazon GameLift Servers는 스팟 인스턴스 유형의 중단 가능성을 예측하고 위험한 인스턴스에 게임 세션을 배치하지 않습니다. 드물게 중단이 발생하더라도 알림을 통해 플레이어의 게임 세션을 정상적으로 종료할 수 있습니다.

Amazon GameLift Servers가 스팟 플릿과 함께 작동하는 방법

게임 호스팅을 위해 스팟 플릿을 설정할 때 Amazon GameLift Servers는 게임 호스팅 실행 가능성을 위해 스팟 플릿 인스턴스 유형 및 위치를 지속적으로 평가합니다.

- 스팟 실행 가능성 알고리즘은 위치별로 스팟 인스턴스 유형의 최근 가용성 패턴과 과거 중단율을 분석합니다.
- 이 분석을 기반으로 Amazon GameLift Servers는 게임 세션 중단의 허용 불가능한 가능성이 있는 스팟 인스턴스 유형 및 위치를 식별합니다. 다음과 같은 작업을 수행합니다.
 - 인스턴스 유형과 위치의 조합을 일시적으로 실행할 수 없는 것으로 표시합니다.
 - 새 게임 세션을 배치할 때 실행 불가능한 스팟 플릿 위치를 고려하지 않습니다. 따라서 게임 세션은 중단 없는 게임 서버 호스팅 가능성이 높은 스팟 플릿 위치에만 배치됩니다.
 - AWS가 회수하지 않더라도 기존 인스턴스의 스팟 플릿 위치를 비우므로 게임 호스팅에 사용할 수 없는 인스턴스에 대해서는 비용을 지불하지 않습니다. 게임 세션 보호가 켜져 있으면 활성 게임 세션이 완료된 후에만 인스턴스가 종료됩니다.
- Amazon GameLift Servers는 게임 호스팅 실행 가능성을 위해 스팟 플릿 인스턴스 유형 및 위치를 지속적으로 재평가합니다. 이전에 실행 불가능한 인스턴스 유형이 업데이트된 기록 데이터를 기반으로 다시 실행 가능하게 되면 스팟 플릿을 다시 확장할 수 있으며 Amazon GameLift Servers가 이를 사용하여 게임 세션 배치를 재개합니다.

설계 고려 사항

스팟 플릿을 사용하도록 솔루션을 설계할 때는 다음 문제를 고려하세요.

- 게임 세션 길이 평가 - 게임 세션의 평균 길이는 스팟이 게임에 얼마나 잘 작동하는지에 영향을 미칠 수 있습니다. 게임 세션이 짧아질수록 더 빠른 반환이 가능해져, 최신 기록 데이터를 기반으로 실행 가능한 인스턴스 유형에서 게임 세션을 안정적으로 운영할 수 있습니다. 더 긴 게임 세션은 최근 실행 가능성 데이터를 평가하지 않고 인스턴스 유형에서 계속 실행되므로 시간이 지남에 따라 중단 위험이 커집니다.
- 인스턴스 유형 가용성 평가 - 모든 플릿 위치가 모든 인스턴스 유형을 스팟으로 제공하는 것은 아닙니다. 스팟 플릿의 인스턴스 유형을 선택할 때 Amazon GameLift Servers 콘솔 플릿 생성 도구를 사용하여 필요한 위치에서 스팟 인스턴스 유형을 찾으도록 도와줍니다. 이 도구를 사용하면 플릿 위치를 선택한 다음 해당 위치의 인스턴스 유형 가용성을 볼 수 있습니다.
- 다중 위치 스팟 플릿 생성 - 여러 위치로 스팟 플릿을 생성할 수 있습니다. 단일 다중 위치 스팟 플릿은 동일한 인스턴스 유형의 인스턴스를 여러 AWS 리전 또는 로컬 영역에 배포합니다. 스팟 실행 가능성 알고리즘은 인스턴스 유형과 위치 모두를 기반으로 실행 가능성을 평가합니다. 스팟 플릿 위치가 실행 불가능한 것으로 평가되는 경우 플릿의 다른 위치에 영향을 주지 않으며 게임 세션을 호스팅하는 데에도 여전히 사용할 수 있습니다.
- 스팟 플릿 다양성을 사용하여 대기열 생성 - 게임 호스팅에 스팟 플릿을 사용하는 경우 게임 세션 배치 대기열을 설정해야 합니다. 각 새 게임 세션 요청에 대해 대기열은 사용 가능한 게임 호스팅 리소

스를 찾고 가능한 최상의 옵션을 선택합니다. 스팟 플릿을 사용할 때에는 위치 및 인스턴스 유형 모두에서 서로 다른 여러 플릿을 검색할 수 있는 대기열이 필요하며, 적어도 하나 이상의 온디맨드 플릿을 백업 용량으로 포함해야 합니다. 다양한 배치 옵션을 제공하는 잘 설계된 다중 플릿 대기열은 중단, 속도 저하 및 장애에 대한 복원력이 뛰어납니다. 스팟용 대기열 설계에 대한 추가 지침은 [스팟 인스턴스를 위한 대기열 빌드](#) 섹션을 참조하세요.

- 정상적으로 중단 처리 - 스팟 중단 발생 시 플레이어에 미치는 영향을 최소화하도록 게임 서버를 설정합니다. 가 스팟 인스턴스를 AWS 회수하면는 서버 SDK 콜백 함수를 사용하여 영향을 받는 모든 서버 프로세스에 종료 알림을 Amazon GameLift Servers 전달합니다 `onProcessTerminate()`. 게임 세션을 정상적으로 종료하려면 게임에서 이 콜백을 구현해야 합니다. 자세한 내용은 [서버 프로세스 종료 알림에 응답](#) 단원을 참조하십시오.

Note

AWS 는 인스턴스를 회수하기 전에 알림을 제공하기 위해 모든 노력을 기울이지만 경고가 도착하기 전에가 스팟 인스턴스를 AWS 회수할 수 있습니다. 또한 게임 서버를 준비하여 예기치 않은 중단을 처리해야 합니다.

- 스팟 중단 중에 서비스를 유지하도록 백업 플릿에 대한 오토 스케일링을 구성합니다. 대상 추적 오토 스케일링은 용량 버퍼를 유지하고 수요에 따라 자동으로 확장합니다. 오토 스케일링을 사용하면 백업 플릿(스팟 또는 온디맨드)이 더 많은 게임 세션 요청을 수신하기 시작할 때마다 용량이 증가하기 시작합니다.

스팟 플릿을 사용할 수 없게 될 때 손실된 용량을 빠르게 교체하려면 사용자 지정 조정 메커니즘은 사용 가능한 대기열 및 플릿 지표를 사용하여 백업 플릿의 신속한 스케일 아웃을 시작할 수 있습니다. `FirstChoiceOutOfCapacity`, `FirstChoiceNotViable`, `PercentAvailableGameSessions`과 같은 지표를 사용하여 스팟 플릿을 실행할 수 없게 되는 시점을 감지합니다. 최근 `PlacementsStarted` 지표 데이터를 분석하여 대체 용량 요구 사항을 추정합니다. 백업 플릿을 조정하여 즉각적인 수요를 처리하면 일반적인 오토 스케일링이 인계될 수 있습니다.

- FlexMatch와의 통합 - 솔루션이 FlexMatch 매치메이커를 사용하는 경우 스팟 플릿에 대한 특별한 요구 사항은 없습니다. 스팟 플릿과 함께 대기열을 사용하도록 매치메이커를 구성할 수 있습니다. Amazon GameLift Servers는 새 게임 세션을 배치할 때와 기존 게임 세션에서 빈 플레이어 슬롯을 채우는 경우를 포함하여 스팟 및 온디맨드 플릿에서 매치 배치의 우선 순위가 자동으로 지정됩니다.

관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화

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 GameLift Servers는 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 GameLift Servers는 각 프로세스의 시작을 시차를 두고 진행합니다. 서버 프로세스의 수명은 제한되어 있습니다. 작업이 끝나면 Amazon GameLift Servers는 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 GameLift Servers가 런타임 구성 변경을 채택하는 방식은 다음과 같습니다.

1. 인스턴스는 Amazon GameLift Servers에 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 GameLift Servers를 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 생성](#)한 다음 Amazon GameLift Servers에 빌드를 업로드합니다. 빌드의 모든 게임 서버는 동일한 플랫폼에서 실행되어야 하며 Amazon GameLift Servers용 서버 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.

- Amazon GameLift Servers가 프로세스를 효율적으로 재활용할 수 있도록 서버 프로세스 종료 시나리오를 처리합니다. 예제:
 - 서버 API `ProcessEnding()`을 호출하는 게임 서버 코드에 종료 절차를 추가합니다.
 - 게임 서버 코드에 콜백 함수 `OnProcessTerminate()`를 구현하여 Amazon GameLift Servers의 종료 요청을 처리합니다.
- Amazon GameLift Servers가 비정상 서버 프로세스가 종료하고 다시 시작하는지 확인합니다. 게임 서버 코드에 `OnHealthCheck()` 콜백 함수를 구현하여 Amazon GameLift Servers에 상태를 다시 보고합니다. Amazon GameLift Servers는 3개의 연속된 보고서에서 비정상적으로 보고된 서버 프로세스를 자동으로 종료합니다. `OnHealthCheck()`를 구현하지 않으면, 프로세스가 통신에 응답하지 않는 한 Amazon GameLift Servers는 서버 프로세스를 정상이라고 가정합니다.

인스턴스당 프로세스 수 선택

인스턴스에서 실행할 동시 프로세스 수를 결정할 때 다음과 같이 유념해야 합니다.

- Amazon GameLift Servers는 각 인스턴스에 지정할 수 있는 [최대 동시 프로세스 수](#)를 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 GameLift Servers는 구성된 전체 프로세스 수보다 많은 동시 프로세스를 실행하지 않습니다. 즉, 이전 런타임 구성에서 새 구성으로의 전환이 점진적으로 이루어질 수 있습니다.

Amazon GameLift Servers 에이전트 사용법

Amazon GameLift Servers 에이전트는 Amazon GameLift Servers 플릿에서 게임 서버 프로세스의 실행을 감독합니다. 에이전트는 플릿의 각 컴퓨팅에 배포되며 컴퓨팅에 대한 자동화된 프로세스 관리, 호스팅 관리 및 로깅을 제공합니다. 에이전트를 사용하려면 게임 서버 빌드가 Amazon GameLift Servers 용 서버 SDK 버전 5.x 이상과 통합되어야 합니다.

Amazon GameLift Servers 에이전트는 관리형 EC2 플릿이 아닌 Amazon GameLift Servers 플릿과 함께 외부에서 사용할 수 있습니다. (관리형 EC2 플릿은 에이전트의 작업을 자동으로 처리합니다.) 에이전트를 사용 여부에 상관없이 Anywhere 플릿을 포함한 Amazon GameLift Servers 플릿을 실행하도록 선택할 수 있습니다. 에이전트가 없으면 필요한 작업을 완료하기 위한 대체 솔루션을 제공해야 합니다.

컴퓨팅에 배포 시 게임 서버 프로세스가 시작되기 전에 Amazon GameLift Servers 에이전트를 시작해야 합니다. 실행 시점에 에이전트가 다음 작업을 완료합니다.

- [RegisterCompute](#) API를 사용하여 Amazon GameLift Servers Anywhere 플릿에 컴퓨팅을 등록합니다.
- [GetComputeAuthToken](#) API를 호출하여 권한 부여 토큰을 가져와 컴퓨팅에서 실행 중인 서버 프로세스에서 사용할 수 있도록 저장합니다.
- 컴퓨팅에 대한 WebSocket URL 환경 변수를 설정하고 Amazon GameLift Servers 서비스에 대한 WebSocket 연결을 설정합니다.
- Amazon GameLift Servers 서비스에서 최신 버전의 플릿 런타임 구성을 위한 요청을 전송합니다.
- 런타임 구성 지침에 따라 서버 프로세스를 시작하고 중지합니다.

Amazon GameLift Servers 에이전트에 대한 소스 코드 및 빌드 지침은 [Amazon GameLift Servers 에이전트](#) GitHub에서 확인할 수 있습니다.

에이전트 정보

Amazon GameLift Servers 에이전트는 플릿에 대해 다음 작업을 처리하도록 설계되었습니다.

프로세스 관리

- 런타임 지침에 정의된 대로 새 서버 프로세스를 시작합니다. 에이전트는 에이전트와 함께 배포된 사용자 지정 런타임 구성을 사용할 수 있습니다. 또는 플릿 정의의 일부로 RuntimeConfiguration을 제공할 수 있습니다. 이 접근 방식은 언제든지 플릿의 런타임 구성을 수정할 수 있다는 이점이 있습니다. 에이전트는 Amazon GameLift Servers 서비스에서 업데이트된 런타임 구성을 주기적으로 요청합니다.

- 서버 프로세스 활성화를 모니터링하고 제시간에 활성화되지 않으면 프로세스를 종료합니다.
- 하트비트를 Amazon GameLift Servers로 전송합니다. 에이전트가 하트비트를 전송하지 못하면 컴퓨팅이 오래된 것으로 표시될 수 있습니다.
- 서버 프로세스가 종료되면 Amazon GameLift Servers에 보고합니다. Amazon GameLift Servers는 이 정보를 사용하여 게임 세션 배치의 게임 서버 가용성을 모니터링합니다.
- 다음을 포함하여 서버 프로세스에 대한 플릿 이벤트를 내보냅니다.
 - `SERVER_PROCESS_INVALID_PATH`: 게임 서버 프로세스 시작 파라미터가 잘못 구성되었습니다.
 - `SERVER_PROCESS_TERMINATED_UNHEALTHY`: 게임 서버 프로세스가 활성화 후 3분 이내에 유효한 상태 확인을 보고하지 않아 종료되었습니다.
 - `SERVER_PROCESS_FORCE_TERMINATED`: `OnProcessTerminate()`가 전송된 후 30초 이내에 게임 서버 프로세스가 완전히 종료되지 않았습니다.
 - `SERVER_PROCESS_CRASHED`: 게임 서버 프로세스가 어떠한 이유로 충돌했습니다.

컴퓨팅 관리

- Amazon GameLift Servers 서비스에서 메시지를 수신하여 컴퓨팅을 종료합니다.
- Amazon GameLift Servers로 컴퓨팅을 종료하라는 메시지를 표시합니다.

로깅

- AWS 계정의 Amazon S3 버킷에 로그를 업로드합니다.

별칭을 사용하여 Amazon GameLift Servers 플릿 지정 추상화

Amazon GameLift Servers 별칭은 호스팅 대상을 추상화하는 데 사용됩니다. Amazon GameLift Servers는 호스팅 대상을 통해 플레이어를 위한 새 게임 세션을 호스팅할 때 사용 가능한 리소스를 검색하는 위치를 알 수 있습니다. 별칭은 다음 시나리오에서 유용합니다.

- 게임이 게임 세션 배치에 다중 플릿 대기열을 사용하지 않는 경우 Amazon GameLift Servers 플릿 ID를 지정하여 새 게임 세션을 요청합니다. 게임이 진행되는 동안 플릿을 여러 번 교체하여 서버 빌드를 업데이트하거나, 호스팅 하드웨어 및 운영 체제를 업데이트하거나, 성능 문제를 해결합니다. 별칭을 사용하여 플릿 ID를 추상화하면 기존 플릿에서 새 플릿으로 플레이어 트래픽을 원활하게 전환할 수 있습니다.

- 게임 클라이언트가 요청할 때 새 게임 세션을 생성하는 것 이외의 다른 작업을 수행하려는 경우, 예를 들어, 오래된 클라이언트를 이용하는 플레이어를 웹사이트 업그레이드로 연결해야할 수 있습니다.

별칭은 라우팅 전략을 지정해야 하며 두 가지의 유형이 있습니다. 단순 라우팅 전략은 플레이어 트래픽을 지정된 플릿 ID로 라우팅하며, 이를 업데이트하여 트래픽을 리디렉션할 수 있습니다. 터미널 라우팅 전략은 새 게임 세션을 생성하는 대신 클라이언트에 메시지를 다시 전달합니다. 별칭의 라우팅 전략은 언제든지 변경할 수 있습니다.

게임 세션 배치에 대기열을 사용하는 경우 플릿을 교체할 때 트래픽을 리디렉션하는 데 별칭이 필요하지 않습니다. 대기열을 사용하면 새 플릿을 추가하고 이전 플릿을 제거할 수 있습니다. 새 게임 세션 요청은 새 플릿을 사용하여 자동으로 이행되므로 플레이어는 이 작업을 볼 수 없습니다. 기존 게임 세션에는 영향을 주지 않습니다. 플릿 ID 또는 별칭을 사용하여 대기열 대상을 식별할 수 있습니다.

주제

- [Amazon GameLift Servers 별칭 생성](#)

Amazon GameLift Servers 별칭 생성

이 주제에서는 게임 세션 배치에 사용할 Amazon GameLift Servers 별칭을 생성하는 방법을 설명합니다.

별칭 생성

Amazon GameLift Servers 콘솔 또는 AWS Command Line Interface(AWS CLI)을 사용하여 별칭을 생성합니다.

Console

[Amazon GameLift Servers 콘솔](#)의 탐색 창에서 별칭 페이지를 실행합니다.

1. 별칭 생성을 선택합니다.
2. 별칭 이름을 입력합니다. 별칭 목록에서 식별할 수 있는 특성을 별칭 이름에 포함하는 것이 좋습니다.
3. 필요에 따라 별칭 설명을 입력합니다.
4. 별칭에 대한 라우팅 전략을 선택합니다.

- a. 단순 라우팅 전략을 선택하는 경우 목록에서 플릿 ID를 선택하여 이 별칭과 연결합니다. 목록에는 현재 선택된 AWS 리전의 모든 플릿이 포함됩니다. 플릿과 동일한 리전에서 별칭을 지정해야 합니다.
 - b. 터미널 라우팅 전략을 선택하는 경우 게임 세션 요청에 따라 Amazon GameLift Servers가 게임 클라이언트로 반환할 문자열 값을 입력합니다. 터미널 별칭이 있는 요청은 메시지가 포함된 예외를 발생시킵니다.
5. (선택 사항) 별칭 리소스에 태그를 추가합니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 용도, 소유자, 환경 등의 기준에 따라 태그를 지정해 AWS 리소스를 유용하게 분류할 수 있습니다. 추가하려는 각 추가 태그에 대해 새 태그 추가를 선택합니다.
 6. 새 플릿을 배포할 준비가 되면 생성을 선택합니다.

AWS CLI

[create-alias](#) 명령을 사용하여 별칭을 생성합니다. Amazon GameLift Servers는 현재 기본 AWS 리전에서 별칭 리소스를 생성합니다(또는 다른 AWS 리전에 리전 태그 추가 지정 가능).

최소한 별칭 이름과 라우팅 전략을 포함해야 합니다. 단순 라우팅 전략의 경우 별칭과 동일한 리전에 있는 플릿의 ID를 지정합니다. 터미널 라우팅 전략의 경우 메시지 문자열을 제공합니다.

Amazon GameLift Servers 호스팅 솔루션 관리(시작 및 이후)

이 섹션에서는 Amazon GameLift Servers 리소스를 장기적으로 관리하는 방법에 대한 지침을 제공합니다. 멀티플레이어 게임을 성공적으로 시작하는 것은 시작에 불과합니다. 프로덕션 호스팅 솔루션을 관리하려면 성능, 비용 최적화, 보안 업데이트, 플레이어 경험에 지속적으로 주의를 기울여야 합니다. 이 섹션에서는 1일 차 시작 활동부터 장기 유지 관리 및 최적화에 이르기까지 프로덕션 환경에서 Amazon GameLift Servers를 실행하는 운영 측면에 대한 지침을 제공합니다.

시작을 준비하거나, 라이브 게임을 관리하거나 또는 확립된 호스팅 솔루션을 최적화하든 이러한 주제는 플레이어 기반에 따라 확장되는 안정적이고 비용 효율적인 게임 호스팅을 유지하는 데 도움이 됩니다.

Amazon GameLift Servers를 사용하는 게임 호스팅 리소스 관리

빌드, 플릿, 게임 세션 대기열, 매치메이커를 포함한 Amazon GameLift Servers 리소스를 관리합니다. Amazon GameLift Servers 콘솔을 사용하여 리소스 목록을 스캔하거나 지표를 포함한 리소스 세부 정보를 볼 수 있습니다. 용 AWS SDK 또는 AWS CLI를 사용하여 리소스 정보를 검색할 수도 있습니다.

Amazon GameLift Servers

Amazon GameLift Servers 콘솔 기능

Amazon GameLift Servers 콘솔을 사용하여 게임 호스팅 리소스와 지속적인 호스팅 활동을 거의 실시간으로 보고 관리할 수 있습니다. 콘솔은 Amazon GameLift Servers용 서비스 API의 대부분의 기능을 위한 그래픽 인터페이스를 제공합니다. 다음으로 콘솔을 사용할 수 있습니다.

- 상위 수준 스냅샷을 보려면 대시보드를 사용합니다. 모든 Amazon GameLift Servers 호스팅 리소스의 번호와 현재 상태를 확인하고 링크를 따라 개별 리소스에 대한 세부 정보를 가져올 수 있습니다.
- 개별 호스팅 리소스를 관리합니다. 모든 Amazon GameLift Servers 리소스를 생성, 확인, 삭제하고 변경 가능한 속성을 업데이트할 수 있습니다. 이벤트 및 성능 지표와 같은 특정 유형의 호스팅 활동을 볼 수도 있습니다.
- 게임 및 플레이어 세션 활동과 상호 작용합니다. 플릿별로 게임 세션 및 플레이어 세션 활동을 추적하고 이 정보를 사용하여 게임 세션 문제를 해결할 수 있습니다. 게임 세션에 대한 세부 정보를 확인하고, 각 게임 세션의 플레이어 세션을 보며, 여러 게임 세션에서 플레이어 활동을 조회합니다. 필요에 따라 개별 게임 세션을 종료할 수도 있습니다.

주제

- [Amazon GameLift Servers 콘솔에서 대시보드 호스팅](#)
- [게임 서버 빌드](#)
- [Amazon GameLift Servers 콘솔의 플릿](#)
- [Amazon GameLift Servers 콘솔의 별칭](#)
- [Amazon GameLift Servers 콘솔의 게임 세션 대기열](#)

Amazon GameLift Servers 콘솔에서 대시보드 호스팅

Amazon GameLift Servers 콘솔 대시보드를 사용하여 AWS 계정에서 Amazon GameLift Servers 호스팅 리소스의 현재 상태에 대한 상위 수준 스냅샷을 가져옵니다. Amazon GameLift Servers 대시보드에서는 다음 보기를 제공합니다.

- 준비 완료, 초기화된, 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 Servers 대시보드 열기

- [Amazon GameLift Servers 콘솔](#)의 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.

GameLift > Dashboard

Dashboard

Build status overview

Viewing data for all builds in N. Virginia region.

View builds

Ready

1

Initialized

0

Failed

0

Fleet status overview

Viewing data for all fleets in N. Virginia region.

View fleets

Active

0

New

0

Deleting

0

Error

0

In progress

0

Terminated

0

Resources (1)

View service quotas

Resource type	Count
Builds	1
Scripts	0
Fleets	0
Aliases	0
Queues	0
Matchmaking rule sets	0
Matchmaking configurations	0

Prepare for your game launch

[Learn more](#)

Fill out a launch questionnaire

Fill out our game launch questionnaire and email it to the GameLift launch team to ensure a smooth launch. The GameLift launch team will verify your GameLift setup and service limits, preparing you for launch.

[Prepare to launch](#)

Features spotlight

Updates on features available in N. Virginia region

March 22, 2022

Updates to Amazon GameLift FlexMatch for greater flexibility

October 28, 2021

New Asia Pacific (Osaka) region and Graviton2 support for Amazon GameLift

게임 서버 빌드

빌드 리소스는 게임 서버 소프트웨어를 나타냅니다. 관리형 EC2 플릿에 배포하기 위해 빌드 패키지를 Amazon GameLift Servers에 업로드합니다.

Amazon GameLift Servers 콘솔에서 또는 또는 AWS SDK for를 사용하여 게임 서버 빌드에 대한 정보를 봅니다Amazon GameLift Servers.

Console

[Amazon GameLift Servers 콘솔](#)의 빌드 페이지에서 관리형 EC2 플릿에 배포하기 위해 Amazon GameLift Servers에 업로드한 모든 게임 서버 빌드에 대한 정보를 확인하고 관리할 수 있습니다. 탐색 창에서 호스팅, 관리형 EC2, 빌드를 선택합니다.

빌드 페이지에는 각 빌드에 대한 다음 정보가 표시됩니다. 기본 설정 도구(테이블 오른쪽 상단 모서리의



아이콘)를 사용하여 필요에 따라 테이블 콘텐츠를 조정할 수 있습니다. 사용자 지정 기본 설정은 AWS 계정 사용자에게 저장되며 이 페이지를 볼 때마다 자동으로 적용됩니다.

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 GameLift Servers에서 바이너리가 수신되기 전에 빌드 시간을 초과했습니다.
- 생성 시간 - Amazon GameLift Servers에 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 Servers 운영 체제가 결정됩니다.
- 크기 - Amazon GameLift Servers에 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 페이지](#)와 동일한 요약 정보를 포함하여 빌드를 실행하는 플릿 목록이 표시됩니다.

AWS SDK

다음 AWS CLI 명령을 사용하여이 리소스에 대한 정보를 검색합니다.

- [ListBuild](#)
- [DescribeBuild](#)

게임 서버 빌드 업데이트

Amazon GameLift Servers 관리형 EC2 호스팅을 위해 게임 서버 빌드를 배포하면 게임 서버 소프트웨어를 업로드하고 Amazon GameLift Servers 빌드 리소스를 생성합니다. Amazon GameLift Servers 빌드를 생성한 후에는 빌드의 메타데이터를 업데이트할 수 있지만 빌드 파일 자체는 업데이트할 수 없습니다. 게임 서버 빌드에 업데이트를 배포하려면 [upload-build](#) 명령을 사용하여 새 파일 세트를 업로드하고 새 Amazon GameLift Servers 빌드를 AWS CLI 생성해야 합니다. 또는 [create-build](#) 명령을 사용하여 제어하는 Amazon S3 버킷에서 새 빌드를 업로드할 수 있습니다. 그런 다음 새 플릿을 생성하여 새 빌드를 배포합니다.

이름과 설명을 포함하여 빌드의 메타데이터를 업데이트할 수 있습니다. 이러한 작업의 경우 Amazon GameLift Servers 콘솔 또는 [update-build](#) AWS CLI 명령을 사용합니다.

게임 빌드 업데이트 자동화

아래의 팁을 활용해 Amazon GameLift Servers 관리형 플릿의 게임 서버 빌드 업데이트 프로세스를 자동화하고 간소화하세요.

- 필요에 따라 게임 세션 대기열을 사용하고 플릿을 전환합니다. Amazon GameLift Servers로 게임 세션 요청을 보낼 때 특정 플릿 대신 게임 세션 대기열을 지정합니다. 대기열을 사용하면 새 빌드를 포함한 플릿을 추가하고 기존 플릿을 제거할 수 있습니다. 자세한 내용은 [게임 세션 배치 구성](#) 섹션을 참조하세요.

- 별칭을 사용하여 플레이어를 새 게임 빌드로 이전합니다. Amazon GameLift Servers로 게임 세션 요청을 보낼 때 플릿 ID 대신 플릿 별칭을 지정합니다. 자세한 내용은 [Amazon GameLift Servers 별칭 생성](#) 단원을 참조하십시오.
- 반복 개발을 위한 설정을 준비합니다. 게임 개발 중에 빠른 반복 개발을 지원하는 호스팅 테스트 환경을 설정하기 위한 옵션을 탐색합니다. [Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정을\(를\)](#) 참조하세요.

GitHub의 [Amazon GameLift Servers 도구 키트](#)에서 다음 리소스를 사용해 보세요.

Fast Build Update Tool(개발 전용)

이 도구를 사용하면 관리형 EC2 플릿의 컴퓨팅에 이미 배포된 게임 서버 빌드를 수정하여 빠른 개발 반복 중에 시간을 절약할 수 있습니다. 이 도구에는 여러 옵션이 있습니다. 전체 게임 빌드를 교체하거나 특정 파일을 변경할 수 있으며 업데이트 후 게임 서버 프로세스를 다시 시작하는 방법을 관리할 수 있습니다. 플릿의 모든 컴퓨팅 또는 대상 개별 컴퓨팅을 업데이트하는 데도 사용할 수 있습니다.

GitHub의 Amazon GameLift Servers 도구 키트 리포지토리를 방문하여 GitHub의 [빠른 빌드 업데이트 도구를](#) 가져오고 사용 방법에 대해 자세히 알아보세요.

프로덕션 배포 샘플 스크립트

이 스크립트는 프로덕션의 관리형 EC2 플릿에 배포된 게임 서버 빌드를 업데이트하는 프로세스를 자동화하는 방법을 보여줍니다. 이 스크립트를 사용하려면 Amazon GameLift Servers 호스팅 솔루션이 별칭을 사용하여 플릿 ID를 추상화해야 합니다. 샘플 스크립트는 업데이트된 빌드 업로드, 새 빌드 생성 및 새 플릿에 배포, 기존 플릿에서 새 플릿으로 플레이어 트래픽 리디렉션, 이전 플릿 삭제와 같은 기본 단계를 처리합니다. 특정 배포 요구 사항에 맞게 샘플 스크립트를 사용자 지정합니다.

GitHub의 Amazon GameLift Servers Toolkit 리포지토리를 방문하여 GitHub의 [프로덕션 배포 샘플 스크립트](#)를 가져오고 사용 방법에 대해 자세히 알아보세요.

Amazon GameLift Servers 콘솔의 플릿

플릿 리소스는 게임 서버와 함께 배포되고 이를 실행하는 일련의 컴퓨팅 시스템을 나타냅니다. 호스팅 솔루션에 따라 관리형 EC2 플릿, 관리형 컨테이너 플릿, Anywhere 플릿 또는 이들의 조합이 있을 수 있습니다.

Amazon GameLift Servers 콘솔에서 또는 또는 AWS SDK for를 사용하여 플릿에 대한 정보를 봅니다
Amazon GameLift Servers.

Console

AWS 계정Amazon GameLift Servers에서에서 게임을 호스팅하기 위해 생성된 모든 플릿에 대한 정보를 볼 수 있습니다. 콘솔의 왼쪽 탐색 창에서 원하는 호스팅 옵션(Anywhere, 관리형 EC2, 관리형 컨테이너)을 찾아 플릿을 선택합니다.

각 호스팅 옵션에 대해 플릿 페이지에는 현재 선택한 AWS 리전에 있는 플릿이 나열됩니다. 플릿 페이지에서 새 플릿을 생성하거나 플릿 하나에 대한 추가 세부 정보를 볼 수 있습니다. 플릿의 [세부 정보 페이지](#)에는 사용 정보, 지표, 게임 세션 데이터, 플레이어 세션 데이터가 포함됩니다. 플릿 기록을 편집하거나 플릿을 삭제할 수도 있습니다.

플릿 페이지에는 기본적으로 다음 요약 정보가 표시됩니다. 기본 설정 도구(테이블 오른쪽 상단 모서리의



아이콘)를 사용하여 필요에 따라 테이블 콘텐츠를 조정할 수 있습니다. 사용자 지정 기본 설정은 AWS 계정 사용자에게 저장되며 이 페이지를 볼 때마다 자동으로 적용됩니다.

- ID - 플릿에 할당된 식별자입니다. 이 ID는 플릿 AWS 리전 이 생성되는 내에서 고유합니다.
- 이름 - 플릿에 부여된 표시 이름입니다.
- 상태 - 플릿의 상태로, 신규, 다운로드 중, 빌드 중, 활성 중 하나일 수 있습니다.
- 생성 시간 - 플릿이 생성된 날짜와 시간입니다.

Note

생성된 지 90일이 넘은 플릿에는 경고 아이콘이 표시됩니다. 모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#)를 참조하세요.

- 플릿 유형 - 게임을 호스팅하는 데 사용되는 인스턴스의 가용성으로, 호스팅 비용에 잠재적으로 영향을 미칠 수 있습니다. 관리형 플릿은 온디맨드(항상 사용 가능) 또는 스팟(가용성은 다름) 인스턴스를 사용할 수 있습니다.
- 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.

- 플레이어 세션 - 플릿의 활성 플레이어 세션 수입니다. 데이터는 5분 지연됩니다.

AWS SDK

다음 AWS CLI 명령을 사용하여이 리소스에 대한 정보를 검색합니다.

- 관리형 EC2 플릿
 - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)
 - [DescribeFleetCapacity](#)
 - [DescribeFleetUtilization](#)
- 관리형 컨테이너 플릿
 - [ListContainerFleets](#)
 - [DescribeContainerFleet](#)
 - [ListContainerGroupDefinitions](#)
 - [DescribeContainerGroupDefinition](#)
- - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)

Amazon GameLift Servers 플릿 구성 업데이트

Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 플릿 설정을 업데이트하거나, 원격 위치를 변경하거나, 플릿을 삭제합니다. 관리형 플릿의 경우 플릿의 게임 서버 빌드 또는 인스턴스 유형을 변경할 수 없습니다. 대신 플릿을 교체해야 합니다.

Fast Build Update Tool(개발 전용)

관리형 EC2 플릿을 사용하면 게임 서버 빌드 업데이트를 배포할 때 각 신규 빌드를 Amazon GameLift Servers에 업로드하고 이를 위한 새 플릿을 생성해야 합니다.

Fast Build Update Tool을 사용하면 개발 중에 이러한 단계를 우회하여 시간을 절약하고 더 빠른 개발 반복을 수행할 수 있습니다. 이 도구를 사용하면 기존 플릿의 모든 컴퓨팅에서 게임 빌드 파일을 빠르게 업데이트할 수 있습니다. 이 도구에는 여러 옵션이 있습니다. 전체 게임 빌드를 교체하거나 6개의 특정 파일을 변경할 수 있으며 업데이트 후 게임 서버 프로세스를 다시 시작하는 방법을 관리할 수 있습니다. 플릿의 개별 컴퓨팅을 업데이트하는 데도 사용할 수 있습니다.

Fast Build Update Tool을 다운로드하고 사용 방법에 대해 자세히 알아보려면 GitHub의 [The Fast Build Update Tool](#)에 대한 Amazon GameLift Servers Toolkit 리포지토리를 참조하세요.

Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 변경 가능한 플릿 속성, 포트 설정 및 런타임 구성을 업데이트할 수 있습니다. 크기 조정 제한을 변경하려면 [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#) 섹션을 참조하세요.

Console

1. [Amazon GameLift Servers 콘솔](#)의 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 Servers 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 Servers 콘솔 또는 AWS CLI를 사용하여 플릿의 원격 위치를 추가하거나 제거할 수 있습니다. 플릿의 홈 리전은 변경할 수 없습니다.

Console

1. [Amazon GameLift Servers 콘솔](#)의 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 GameLift Servers는 상태가 NEW로 설정된 새 위치를 목록에 추가합니다. 그런 다음 Amazon GameLift Servers는 추가된 각 위치에 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 DELETING으로 설정된 상태로 목록에 남게 됩니다. 그런 다음 Amazon GameLift Servers는 제거된 위치에서 활동을 종료하는 프로세스를 시작합니다. 게임 세션을 호스팅하는 활성 인스턴스가 있는 경우 Amazon GameLift Servers는 게임 서버 종료 프로세스를 사용하여 게임 세션을 정상적으로 종료하고, 게임 서버와 인스턴스를 종료합니다.

AWS CLI

다음 AWS CLI 명령을 사용하여 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 [CreateVpcPeeringAuthorization](#)을 호출하여 권한 부여를 요청합니다. Amazon GameLift Servers는 플릿을 삭제하는 동안 VPC 피어링 연결을 삭제합니다.

Amazon GameLift Servers 콘솔 또는 AWS CLI 도구를 사용하여 플릿을 삭제할 수 있습니다.

Console

1. [Amazon GameLift Servers 콘솔](#)의 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 플릿을 삭제합니다.

- [delete-fleet](#)

Amazon GameLift Servers 콘솔의 플릿 세부 정보

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 식별자입니다. 이 ID는 플릿 AWS 리전 이 생성되는 내에서 고유합니다.
- 플릿 이름 - 플릿에 부여된 표시 이름입니다.
- ARN - 이 플릿에 할당된 고유 식별자입니다. 플릿의 ARN은 Amazon GameLift Servers 리소스로 플릿을 식별하고 리전 및 AWS 계정을 지정합니다.
- 설명 - 플릿에 대한 간략한 설명을 입력합니다.
- 상태 - 플릿의 현재 상태로, 신규, 다운로드 중, 빌드 중, 활성화될 수 있습니다.
- 생성 시간 - 플릿이 생성된 때의 날짜와 시간입니다.

Note

생성된 지 90일이 넘는 플릿에는 경고 아이콘이 표시됩니다. 모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#)를 참조하세요.

- 컴퓨팅 유형 - 게임을 호스팅하는 데 사용되는 컴퓨팅 유형입니다. 플릿은 관리형 EC2, 관리형 컨테이너 또는 Anywhere 플릿일 수 있습니다.
- 운영 체제 - 플릿의 모든 인스턴스에 배포되는 Amazon Machine Image(AMI)입니다. AMI 버전은 플릿이 생성될 때 사용할 수 있는 최신 버전입니다.
- TLS 인증서 - 게임 서버를 인증하고 모든 클라이언트/서버 통신을 암호화하기 위해 TLS 인증서를 사용할 플릿의 역량 상태입니다.
- 플릿 유형 - 게임을 호스팅하는 데 사용되는 인스턴스의 가용성으로, 호스팅 비용에 잠재적으로 영향을 미칠 수 있습니다. 플릿은 온디맨드(항상 사용 가능) 또는 스팟(가용성은 다름) 인스턴스를 사용할 수 있습니다.

- EC2 인스턴스 유형 - 플릿을 생성할 때 플릿으로 선택한 Amazon EC2 [인스턴스 유형](#)입니다.
- 인스턴스 역할 ARN - 플릿 생성 중에 다른 AWS 리소스에 대한 액세스를 관리하는 AWS IAM 역할입니다.
- 인스턴스 역할 자격 증명 공급자 - 플릿 생성 중에 다른 AWS 리소스에 대한 액세스를 관리하는 AWS IAM 역할입니다.
- 지표 그룹 이름 - 여러 플릿에 대한 지표를 집계하는 데 이 그룹이 사용됩니다.
- 게임 크기 조정 보호 정책 - 플릿의 게임 세션 보호가 활성화되어 있는지 여부를 나타내며, 이는 스케일 다운 이벤트 중에 활성 게임 세션이 조기에 종료되지 않도록 합니다.
- 플레이어당 최대 게임 세션 - 정책 기간 동안 플레이어가 생성할 수 있는 최대 세션 수입니다.
- 정책 기간 - 플레이어가 생성한 세션 수를 제한하는 데 사용되는 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.

Metrics

지표 탭은 시간 경과에 따른 플릿 측정치를 그래픽으로 보여줍니다. Amazon GameLift Servers의 지표 사용에 대한 자세한 내용은 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#) 섹션을 참조하세요.

이벤트

이벤트 탭은 플릿에 발생한 모든 이벤트의 로그를 이벤트 코드, 메시지, 타임스탬프와 함께 제공합니다. Amazon GameLift Servers 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 Servers를 사용하여 게임 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.

Auto Scaling 정책

이 섹션에서는 플릿에 적용된 Auto Scaling 정책에 대한 정보를 다룹니다. 대상 기반 정책을 설정하거나 업데이트할 수 있습니다. AWS SDK 또는 CLI를 사용하여 정의해야 하는 플릿의 규칙 기반 정책이 여기에 표시됩니다. 크기 조정에 대한 자세한 내용은 [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#) 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 플레이어 세션을 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 [Amazon GameLift Servers 콘솔의 게임 및 플레이어 세션](#) 섹션을 참조하세요.

Amazon GameLift Servers 콘솔의 게임 및 플레이어 세션

Amazon GameLift Servers 콘솔을 사용하여 게임 세션 및 플레이어 세션으로 작업할 수 있습니다. 게임 세션 및 플레이어 세션에 대한 자세한 내용은 [Amazon GameLift Servers 및 플레이어 경험](#) 섹션을 참조하세요. Amazon GameLift Servers 콘솔은 게임 세션 관련 문제를 조사하는 데 도움이 되는 정보와 도구를 제공합니다.

할 수 있는 작업:

- 특정 플릿에서 호스팅되는 게임 세션 및 플레이어 세션 활동을 살펴봅니다.
- 여러 플릿에서 특정 플레이어의 게임 세션 활동을 조회합니다.
- 특정 게임 세션을 종료합니다.

게임 세션 세부 정보 보기

게임 세션 및 플레이어 세션 데이터는 게임 세션을 호스팅하는 플릿별로 구성됩니다.

게임 세션 및 플레이어 세션에 액세스하려면

1. [Amazon GameLift Servers 콘솔](#)에서 왼쪽 탐색 창을 엽니다. 호스팅 솔루션 유형을 선택하고 플릿 페이지를 엽니다. 예제:
 - 호스팅, Anywhere, 플릿

- 호스팅, 관리형 EC2, 플릿
 - 호스팅, 관리형 컨테이너, 플릿
2. 각 플릿 페이지에는 현재 선택한 플릿 목록이 표시됩니다 AWS 리전. 게임 세션 데이터를 확인할 플릿을 선택합니다.
 3. 플릿의 세부 정보 페이지에서 게임 세션 탭을 엽니다. 이 탭에는 플릿에서 호스팅되는 모든 게임 세션이 요약 정보와 함께 나열됩니다. 기본 설정 도구(테이블 오른쪽 상단 모서리의 )를 사용하여 필요에 따라 테이블 콘텐츠를 조정할 수 있습니다. 사용자 지정 기본 설정은 AWS 계정 사용자에게 저장되며 이 페이지를 볼 때마다 자동으로 적용됩니다.
 4. 추가 정보를 보려면 목록에서 게임 세션을 선택합니다.
 5. 게임 세션에 플레이어 세션 데이터가 포함된 경우 플레이어 세션 보기를 선택하여 게임 세션 ID가 자동으로 채워진 플레이어 세션 조회 도구를 엽니다.

게임 세션 세부 정보에는 다음 정보가 포함되어 있습니다.

- 상태 - 게임 세션 상태입니다.
 - 활성화하는 중 - 인스턴스가 게임 세션을 시작합니다.
 - 활성 - 게임 세션이 실행 중이며 플레이어를 수락할 수 있습니다(세션의 [플레이어 생성 정책](#)에 따라).
 - 종료됨 - 게임 세션이 종료되었습니다.
- ARN - 게임 세션의 Amazon 리소스 이름(ARN)입니다.
- 이름 - 게임 세션에 대해 생성된 이름입니다.
- 위치 - Amazon GameLift Servers가 게임 세션을 호스팅한 위치입니다.
- 생성 시간 - Amazon GameLift Servers가 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 GameLift Servers가 플릿에서 시작하는 모든 새 인스턴스를 설정하는 게임 세션 보호 유형입니다.

게임 데이터

시작 시 게임 세션에 전송하는 문자열 형식의 게임 속성 데이터입니다.

게임 속성

시작 시 게임 세션에 전송하는 키/값 페어 형식 형식의 게임 속성 데이터입니다.

매치메이킹 데이터

FlexMatch로 생성한 게임 세션의 경우 매치메이킹 데이터가 매치메이킹 구성 및 규칙 세트에 대한 정보를 설명합니다. 여기에는 각 매치의 플레이어 속성과 팀 할당이 포함됩니다. 데이터는 JSON 형식입니다. FlexMatch 매치메이킹에 대한 자세한 내용은 [매치메이커 구축](#)을 참조하세요.

플레이어 세션 데이터 조회

게임 호스팅 솔루션이 플레이어 세션을 사용하고 고유한 플레이어 ID를 제공하는 경우 여러 플릿의 과거 또는 현재 게임 세션에 대한 플레이어별 활동을 탐색할 수 있습니다. 다음 방법 중 하나를 사용하여 플레이어 세션 조회 도구를 엽니다.

- Amazon GameLift Servers 콘솔에서 왼쪽 탐색 창을 열고 플레이어 세션 조회를 선택한 다음 사용할 필터 유형을 선택합니다.
- 플릿의 게임 세션 세부 정보를 볼 때 플레이어 세션 보기를 선택합니다. 게임 세션 ID 필터가 미리 선택되고 게임 세션 값이 채워진 게임 세션과 함께 조회 도구가 열립니다.

조회 도구를 사용할 때 다음 정보를 제공할 수 있습니다.

- 특정 플레이어 세션에 대한 정보를 가져오는 플레이어 세션 ID입니다.
- 요청된 게임 세션의 모든 플레이어 세션에 대한 정보를 가져오는 게임 세션 ID입니다. 결과는 슬롯을 예약했거나 게임 세션에 연결된 모든 플레이어를 나타냅니다. 선택적으로 플레이어 세션 상태를 기준으로 결과를 필터링할 수 있습니다.
- 요청된 플레이어의 모든 플레이어 세션에 대한 정보를 가져오는 플레이어 ID입니다. 결과는 플레이어가 참여한 모든 게임 세션을 나타냅니다.

Note

조회 도구는 현재 선택한에서 모든 플레이어 세션 활동을 검색합니다 AWS 리전. 리전에 여러 플릿이 있는 경우 결과에는 모든 플릿의 플레이어 세션 활동이 포함됩니다. 다중 위치 플릿의 경우 결과에는 플릿의 원격 위치에 대한 플레이어 세션 활동도 포함됩니다.

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.
- 게임 세션 ID - 게임 세션에 할당된 식별자입니다.
- 플릿 ID - 게임 세션을 호스팅한 플릿에 할당된 식별자입니다.
- 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.
- 연결 데이터 - 플레이어가 게임 세션에 연결하는 데 사용한 IP 주소, DNS 이름 및 포트입니다.
- 플레이어 데이터 - 플레이어 세션 생성 중에 제공된 플레이어에 대한 정보입니다.

게임 세션 종료

Amazon GameLift Servers 콘솔을 사용하여 특정 게임 세션을 종료합니다. 이 기능은 게임 세션을 찾고 종료 신호를 보내는 빠르고 간단한 방법을 제공합니다. 또 다른 종료 방법을 사용하려면 게임 세션이 실행 중인 플릿 인스턴스를 찾고, 인스턴스에 원격으로 액세스하며, 게임 세션을 수동으로 종료해야 합니다.

게임 세션은 이유와 상관없이 종료할 수 있습니다. 가장 일반적인 이유는 자연스럽게 종료되지 않는 게임 세션을 해결하기 위한 것입니다. 따라서 게임 세션의 호스팅 리소스에서 새 게임 세션을 호스팅할 수 없으며 플릿의 호스팅 용량이 저하됩니다.

Note

이 기능은 호스팅 솔루션의 특정 구성 설정에 의존합니다. 여기에는 다음과 같은 제한 사항이 있습니다.

- 게임 세션은 Amazon GameLift Servers v5 이상용 서버 SDK를 사용하여 게임 서버 빌드를 실행하는 플릿에서 호스팅되어야 합니다. 게임 서버가 이전 버전으로 배포된 경우 원격 액세스를 사용하여 게임 세션을 삭제해야 합니다.
- 게임 세션이 Anywhere 플릿에서 호스팅되는 경우 플릿은 Amazon GameLift Servers 에이전트를 사용하여 게임 서버 프로세스를 관리해야 합니다.

게임 세션을 종료하려면

1. [Amazon GameLift Servers 콘솔](#)에서 왼쪽 탐색 창을 엽니다. 호스팅 솔루션 유형을 선택하고 플릿 페이지를 엽니다. 예제:
 - 호스팅, Anywhere, 플릿
 - 호스팅, 관리형 EC2, 플릿
 - 호스팅, 관리형 컨테이너, 플릿
2. 각 플릿 페이지에는 현재 선택한 플릿 목록이 표시됩니다 AWS 리전. 종료하려는 게임 세션을 호스팅하는 플릿을 선택합니다.
3. 플릿의 세부 정보 페이지에서 게임 세션 탭을 엽니다. 게임 세션 목록에서 종료하려는 세션을 선택하고 종료 버튼을 선택합니다.
4. Terminate game session? 창에서 올바른 게임 세션을 종료하고 있는지 확인하고 종료 방법을 선택합니다.
 - 정상 게임 세션 종료 - 이 옵션은 종료하기 위해 게임 세션을 호스팅하는 서버 프로세스에 신호를 보냅니다. 게임 서버 빌드가 Amazon GameLift Servers에 제대로 통합된 경우 서버 프로세스는 게임 세션 종료 시퀀스를 시작하고 Amazon GameLift Servers가 종료되었음을 알리고 중지합니다. 게임 설계에 따라 종료 시퀀스에는 데이터를 저장하고 활성 플레이어에게 알리는 등 게임 세션을 정상적으로 완료하는 단계가 포함될 수 있습니다. 이 방법을 사용하면 게임 세션 종료 시퀀스를 완료하는 데 약간 지연될 수 있습니다.
 - 즉각적인 게임 세션 종료 - 이 옵션은 프로세스 관리자에게 신호를 보내 게임 세션을 호스팅하는 서버 프로세스를 종료합니다. 이 옵션은 정상 게임 세션 종료를 우회합니다. 서버 프로세스가 응답할 수 없는 경우에도 게임 세션을 종료할 수 있습니다.

5. 게임 세션 종료를 확인합니다. 게임 세션 콘솔 페이지에서 종료 진행 상황을 추적할 수 있습니다. 종료가 완료되면 게임 세션 상태가 '종료 중'에서 '종료됨'으로 변경됩니다.

관련 주제

- AWS SDK 및를 사용하여 게임 세션을 종료할 수도 있습니다 AWS CLI. 자세한 내용과 예제는 Amazon GameLift Servers API 참조의 [TerminateGameSession](#) 주제를 참조하세요.
- 게임 서버 통합 및 서버 프로세스가 Amazon GameLift Servers 서비스의 신호에 응답하는 방식에 대한 자세한 내용은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 섹션을 참조하세요.

Amazon GameLift Servers 관리형 컨테이너 플릿 업데이트

컨테이너 그룹 정의를 포함하여 관리형 컨테이너 플릿의 대부분의 속성을 업데이트할 수 있습니다. 업데이트 중인 설정에 따라 플릿 업데이트가 새 플릿 배포를 시작할 수 있습니다. 플릿 배포에서는 플릿의 모든 인스턴스가 제거되고 새 구성의 인스턴스로 대체됩니다. 배포가 필요한 설정은 다음과 같습니다.

- 컨테이너 이미지 업데이트를 포함한 컨테이너 그룹 정의
- 연결 포트 범위 및 인바운드 권한
- 로그 구성

[Amazon GameLift Servers 콘솔](#) 또는 AWS Command Line Interface(AWS CLI)에서 플릿 배포 상태를 추적하여 컨테이너 플릿을 생성할 수 있습니다.

Note

플릿의 런타임 환경은 업데이트할 수 없습니다. 플릿 생성 시 플릿의 Amazon Machine Image(AMI)는 사용 가능한 최신 버전의 Linux AMI로 설정됩니다. 이 플릿에 배포된 모든 컨테이너 이미지는 이 버전과 호환되어야 합니다. 플릿의 AMI를 변경하거나 최신 버전으로 업그레이드하려면 새 플릿을 생성해야 합니다. 모범 사례에 따라, 호스팅된 게임 서버의 안전한 최신 런타임 환경을 유지하기 위해 30일마다 플릿을 교체하는 것이 좋습니다. 자세한 지침은 [Amazon GameLift Servers의 보안 모범 사례](#) 단원을 참조하십시오.

Console

[Amazon GameLift Servers 콘솔](#)에서 플릿을 생성할 AWS 리전을 선택합니다. 컨테이너 그룹 정의는 플릿을 생성하려는 리전과 동일한 리전에 있어야 합니다.

콘솔의 왼쪽 탐색 모음을 열고 관리형 컨테이너: 플릿을 선택합니다. 관리형 컨테이너 플릿 페이지의 목록에서 플릿을 선택하고 편집을 선택합니다.

1. 필요에 따라 컨테이너 플릿 설정을 업데이트합니다. 작업을 마쳤으면 생성을 선택합니다.
2. 업데이트에 플릿 배포가 필요한 경우 다음과 같이 배포 옵션을 지정하라는 메시지가 표시됩니다.
 - 게임 세션 보호 활성화 게임 세션(안전한 배포)이 있는 플릿 인스턴스를 보호하도록 선택할 수 있습니다. 이 설정을 사용하면 게임 세션이 종료될 때까지 플릿 인스턴스가 교체되지 않습니다. 게임 세션 활동(안전하지 않은 배포)에 관계없이 플릿 인스턴스를 교체하도록 선택할 수도 있습니다. 안전하지 않은 배포는 배포 시간을 줄이기 위해 개발 및 테스트 단계에서 유용합니다.
 - 최소 건전 백분율. 플릿의 인스턴스가 교체되는 속도를 관리할 수 있습니다. 이 설정을 사용하면 배포 중의 정상 작업을 최소화할 수 있습니다. 값이 낮을수록 배포 속도의 우선 순위가 지정되고 값이 높을수록 배포 전반에 걸쳐 게임 서버 가용성이 높게 유지됩니다.
 - 배포 실패 전략. 배포에 실패할 경우 수행할 작업을 결정합니다. 배포 실패는 업데이트된 컨테이너 중 일부가 상태 확인에 실패했으며 손상된 것으로 간주됨을 의미합니다. 모든 플릿 인스턴스를 이전에 배포된 상태로 자동으로 롤백하도록 배포를 설정할 수 있습니다. 또는 디버깅에 사용할 손상된 플릿 인스턴스 중 일부를 유지 관리하도록 선택할 수 있습니다.

요청이 성공하면 콘솔에 관리형 컨테이너 플릿용 배포 탭이 표시됩니다. 이 탭을 사용하여 각 배포의 상태를 추적합니다. 플릿에 대한 새 배포를 시작하면 이 작업은 플릿에 대해 현재 진행 중인 모든 배포를 자동으로 취소합니다.

AWS CLI

AWS CLI로 컨테이너 플릿을 생성하려면 명령줄 창을 열고 `update-container-fleet` 명령을 사용합니다. 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 [update-container-fleet](#) 섹션을 참조하세요.

다음 예시에서는 기존 컨테이너 플릿을 다음과 같은 특성으로 업데이트합니다.

- 버전 2를 사용하도록 게임 서버 컨테이너 그룹 정의를 업데이트합니다.
- 안전한 배포 옵션을 지정합니다.

```
{
  "DeploymentConfiguration": {
    "ImpairmentStrategy": "ROLLBACK",
    "MinimumHealthyPercentage": 75,
    "ProtectionStrategy": "WITH_PROTECTION"
  },
  "FleetId": "containerfleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "GameServerContainerGroupDefinitionName": "arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2"
}
```

Amazon GameLift Servers 컨테이너 플릿에 대한 컨테이너 그룹 정의 업데이트

개별 컨테이너 정의를 포함하여 컨테이너 그룹 정의의 대부분의 속성을 업데이트할 수 있습니다. 컨테이너 그룹 정의에는 버전 번호가 있습니다. 컨테이너 그룹 정의를 업데이트하면 Amazon GameLift Servers가 업데이트를 저장하고 정의의 버전 번호를 증가시킵니다. 컨테이너 플릿을 구성할 때 배포할 컨테이너 그룹 정의에 대한 버전을 지정할 수 있습니다.

컨테이너 그룹 정의를 업데이트한 후 새 버전을 새 컨테이너 플릿 또는 기존 컨테이너 플릿에 배포할 수 있습니다.

게임 서버 컨테이너 그룹 정의 업데이트

이 주제에서는 Amazon GameLift Servers 콘솔 또는 AWS CLI 도구를 사용하여 게임 서버 컨테이너 그룹 정의를 업데이트하는 방법을 설명합니다. 선택적 기능에 대한 자세한 내용은 [Amazon GameLift Servers 컨테이너 플릿 사용자 지정](#) 섹션을 참조하세요.

컨테이너 그룹 정의를 업데이트하려면:

Console

[Amazon GameLift Servers 콘솔](#)에서 컨테이너 그룹을 생성할 AWS 리전 를 선택합니다.

콘솔의 왼쪽 탐색 모음을 열고 관리형 컨테이너: 그룹 정의를 선택합니다. 컨테이너 그룹 정의 페이지에서 업데이트할 컨테이너 그룹 정의 및 버전을 선택합니다.

업데이트를 저장한 후 새 버전을 사용하여 새 컨테이너 플릿을 생성하거나 기존 컨테이너 플릿에 업데이트를 배포할 수 있습니다.

1단계: 컨테이너 그룹 정의 세부 정보 정의

- 총 메모리 및 vCPU 제한 설정을 업데이트할 수 있습니다.

2단계: 컨테이너 정의 추가

다음과 같은 컨테이너 정의 업데이트를 수행할 수 있습니다.

- 기존 컨테이너 정의를 업데이트합니다.
 - 새 지원 컨테이너 정의를 추가합니다.
 - 지원 컨테이너 정의를 제거합니다.
1. ECR 이미지 URI를 업데이트할 수 있습니다. 새 이미지와 일치하도록 서버 SDK 버전 설정을 업데이트해야 합니다.
 2. 필요에 따라 내부 컨테이너 포트 범위를 업데이트할 수 있습니다. 이러한 설정을 변경하면 이러한 변경 사항이 플릿에 배포될 때 컨테이너 플릿의 연결 포트 설정에 영향을 미칠 수 있습니다. 자세한 내용은 [네트워크 연결 구성](#) 섹션을 참조하세요.

3단계: 종속성 구성

- 필요에 따라 종속성을 변경할 수 있습니다. 자세한 내용은 [컨테이너 종속성 설정](#) 단원을 참조하십시오.

3단계: 검토 및 생성

- 컨테이너 그룹 정의 업데이트를 검토합니다. 편집을 사용하여 모든 섹션에 추가 변경 사항을 적용합니다. 완료되면 생성을 선택하여 컨테이너 그룹 정의의 새 버전을 생성합니다.

요청이 성공하면 콘솔에 새 컨테이너 그룹 정의 리소스에 대한 세부 정보 페이지가 표시됩니다. Amazon GameLift Servers가 그룹에 대한 모든 컨테이너 이미지의 스냅샷을 생성하기 시작하므로 처음에 상태는 COPYING입니다. 이 단계가 완료되면 컨테이너 그룹 정의 상태가 READY로 변경됩니다. 컨테이너 그룹 정의가 READY 상태여야 컨테이너 플릿을 생성할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 컨테이너 그룹 정의를 생성하거나 업데이트할 때 컨테이너 정의 구성을 별도의 JSON 파일로 유지 관리합니다. CLI 명령에서 파일을 참조할 수 있습니다. 스키마 예제는 [컨테이너 정의 JSON 파일 생성](#) 섹션을 참조하세요.

정의를 업데이트할 때 업데이트할 값만 지정하면 됩니다. Amazon GameLift Servers는 업데이트 요청에 포함되지 않은 모든 값을 유지합니다. 컨테이너 정의를 변경하는 경우, 그러나 컨테이너 정의를 변경할 때는 전체 세트를 제공합니다.

컨테이너 그룹 정의를 업데이트하려면

새 컨테이너 그룹 정의를 업데이트하려면 `update-container-group-definition` CLI 명령을 사용합니다. 이 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 [update-container-group-definition](#)을 참조하세요.

Example: 게임 서버 컨테이너 그룹

컨테이너 그룹 정의를 검색, 업데이트 또는 삭제할 때 또는 컨테이너 플릿을 생성 또는 업데이트할 때 컨테이너 그룹 정의 버전을 지정할 수 있습니다. 각 컨테이너 그룹 정의에는 버전 속성이 있습니다. 또한 및 정의의 ARN 값은 버전 번호를 지정합니다.

이 예제에서는 게임 서버 컨테이너 그룹 정의에 대한 변경 요청을 보여줍니다. 이 그룹에 대한 컨테이너 정의가 포함된 JSON 파일을 생성했다고 가정합니다. 이 예제에서는 정의 이름에 ARN 값을 사용하고 업데이트가 버전 1이 되도록 지정합니다.

```
aws gamelift update-container-group-definition \
  --name arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/
  MyAdventureGameContainerGroup:1 \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --container-definitions file://SimpleServer.json
```

컨테이너 그룹 정의 복제

Amazon GameLift Servers 콘솔을 사용하여 기존 컨테이너 그룹 정의를 복제할 수 있습니다.

컨테이너 그룹을 복제하려면

1. [Amazon GameLift Servers 콘솔](#)의 왼쪽 탐색 창에서 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 미리 채워진 설정이 있는 컨테이너 그룹 생성 마법사가 열립니다.
4. 복제된 컨테이너 그룹의 새 이름을 입력합니다. 동일한 리전의 컨테이너 그룹에는 고유한 이름이 있어야 합니다.
5. 컨테이너 그룹 및 컨테이너 정의 페이지를 단계별로 살펴보고 새 컨테이너 그룹을 검토 및 생성합니다.

Amazon GameLift Servers 컨테이너 플릿에 대한 컨테이너 그룹 정의 삭제

컨테이너 그룹 정의를 삭제하는 몇 가지 옵션이 있습니다. 컨테이너 그룹 정의를 삭제하면 이 작업은 컨테이너 그룹의 모든 컨테이너 정의도 삭제합니다.

컨테이너 그룹 정의에는 여러 버전이 있을 수 있습니다. 컨테이너 그룹 버전은 이름이 동일하지만 버전 번호는 다릅니다. 컨테이너 그룹 정의의 ARN 이름과 버전을 모두 지정합니다.

컨테이너 그룹 정의를 검색, 업데이트 또는 삭제할 때 또는 컨테이너 플릿을 생성 또는 업데이트할 때 컨테이너 그룹 정의 버전을 지정할 수 있습니다. 각 컨테이너 그룹 정의에는 버전 속성이 있습니다. 또한 정의의 ARN 값은 버전 번호를 지정합니다.

컨테이너 그룹 정의를 삭제하는 방법에는 여러 가지가 있습니다.

- 특정 정의의 모든 버전을 삭제할 수 있습니다.
- 특정 정의의 모든 버전을 삭제할 수 있습니다.
- 일부 최신 버전을 유지하고 특정 정의의 이전 버전을 삭제할 수 있습니다. 예를 들어 버전 5 이전의 모든 버전을 삭제할 수 있습니다.

컨테이너 그룹 정의 버전은 컨테이너 플릿에서 사용되지 않는 경우에만 삭제할 수 있습니다.

컨테이너 그룹 정의를 삭제하려면

Console

[Amazon GameLift Servers 콘솔](#)에서 컨테이너 그룹을 생성할 AWS 리전을 선택합니다.

콘솔의 왼쪽 탐색 모음을 열고 관리형 컨테이너: 그룹 정의를 선택합니다. 컨테이너 그룹 정의 페이지에서 수정하려는 정의를 선택하고 삭제를 선택합니다.

생성하려는 삭제 유형을 선택하고 삭제 유형에 따라 추가 설정을 지정하라는 메시지가 표시됩니다.

AWS CLI

- 컨테이너 그룹 정의를 삭제하려면 `delete-container-group-definition` CLI 명령을 사용하고 수행하려는 삭제 유형에 대한 값을 제공합니다. 이 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 [delete-container-group-definition](#)을 참조하세요.

이 예제는 게임 서버 컨테이너 그룹 정의에서 버전 5 이전의 모든 버전을 삭제하라는 요청을 보여줍니다.

Example

```
aws gamelift delete-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --version-count-to-retain 5 \
```

Amazon GameLift Servers 콘솔의 별칭

별칭 리소스는 플릿 또는 터미널 대상에 대한 포인터를 나타냅니다. 플레이어 트래픽을 특정 호스팅 대상으로 전송하는 데 사용됩니다. 별칭을 사용하여 대상을 추상화하면 호스팅 플릿을 업데이트할 때 트래픽을 보다 쉽게 리디렉션할 수 있습니다.

Amazon GameLift Servers 콘솔에서 또는 또는 AWS SDK for를 사용하여 별칭에 대한 정보를 봅니다 Amazon GameLift Servers.

Console

별칭 페이지에는 트래픽별 호스팅 대상을 지정하는 Amazon GameLift Servers 별칭에 대한 정보가 표시됩니다. 별칭을 보려면 탐색 창에서 호스팅, 별칭을 선택합니다.

별칭 페이지에서 다음 작업을 수행할 수 있습니다.

- 새 별칭을 만듭니다. 별칭 생성을 선택합니다.
- 별칭 테이블을 필터링하고 정렬합니다. 표 맨 위의 컨트롤을 사용합니다. 기본 설정 도구(테이블 오른쪽 상단 모서리의



아이콘)를 사용하여 필요에 따라 테이블 콘텐츠를 조정할 수 있습니다. 사용자 지정 기본 설정은 AWS 계정 사용자에게 저장되며 이 페이지를 볼 때마다 자동으로 적용됩니다.

- 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. 언제든지 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.

AWS SDK

다음 AWS CLI 명령을 사용하여이 리소스에 대한 정보를 검색합니다.

- [ListAliases](#)
- [DescribeAlias](#)

별칭 편집

Amazon GameLift Servers 콘솔을 사용하거나 AWS CLI 명령 [update-alias](#)를 사용하여 별칭을 편집할 수 있습니다.

이 주제에서는 게임 세션 배치에 사용할 Amazon GameLift Servers 별칭을 편집하는 방법을 설명합니다. 다음과 같이 편집할 수 있습니다.

별칭 편집

Amazon GameLift Servers 콘솔 또는 AWS Command Line Interface(AWS CLI)을 사용하여 별칭을 편집합니다.

Console

[Amazon GameLift Servers 콘솔](#)의 탐색 창에서 별칭 페이지를 실행합니다.

1. 편집할 별칭을 선택하고 편집을 선택합니다. 편집하려는 별칭이 보이지 않으면 현재 선택한 AWS 리전을 확인합니다.
2. 별칭 편집 페이지에서 다음을 편집할 수 있습니다.
 - 별칭 이름 변경
 - 별칭 설명 변경
 - 라우팅 전략을 단순에서 터미널로, 또는 터미널에서 단순으로 변경
 - 단순 라우팅 전략의 별칭에 연결된 플릿 ID 변경
 - 터미널 라우팅 전략의 별칭에 대한 메시지 텍스트 변경
3. 변경 사항 저장을 선택합니다. 단순 라우팅 전략의 별칭에 대한 플릿 ID를 업데이트할 때 전환이 완료되는 데 최대 2분이 걸릴 수 있습니다. 완료되는 동안 이전 플릿에서 새 게임 세션 배치가 발생할 수 있습니다.

AWS CLI

[update-alias](#) 명령을 사용하여 별칭 리소스를 변경합니다. 현재 기본 AWS 리전에서 별칭 리소스를 업데이트하거나 `--region` 태그를 추가하여 다른 AWS 리전을 지정할 수 있습니다.

다음 속성을 변경할 수 있습니다.

- 별칭 이름

- 별칭 설명
- 라우팅 전략 유형. 새 라우팅 전략에 대한 플릿 ID 또는 메시지 문자열을 제공해야 합니다.
- 기존 단순 라우팅 전략의 플릿 ID. 플릿 ID는 별칭과 동일한 리전에 있어야 합니다.
- 기존 터미널 라우팅 전략의 메시지 문자열

Amazon GameLift Servers 콘솔의 게임 세션 대기열

대기열 리소스는 게임 세션 배치 구성 요소의 구성을 나타냅니다. 구성 선택에 따라 Amazon GameLift Servers가 새 게임 세션을 호스팅할 수 있는 최상의 컴퓨팅 리소스를 검색하고 선택하는 방법을 결정합니다.

Amazon GameLift Servers 콘솔에서 또는 또는 AWS SDK for를 사용하여 게임 세션 대기열에 대한 정보를 봅니다Amazon GameLift Servers.

Console

새 게임 세션에 대한 요청을 처리하는 데 사용되는 모든 대기열에 대한 정보를 볼 수 있습니다. 대기열 페이지에는 현재 선택된 게임 세션 대기열이 표시됩니다 AWS 리전. 대기열 페이지에서 새 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 배치 구성](#) 섹션을 참조하세요.

대기열 페이지에 각 대기열에 대해 다음과 같은 요약 정보가 표시됩니다. 기본 설정 도구(테이블 오른쪽 상단 모서리의



아이콘)를 사용하여 필요에 따라 테이블 콘텐츠를 조정할 수 있습니다. 사용자 지정 기본 설정은 AWS 계정 사용자에게 저장되며 이 페이지를 볼 때마다 자동으로 적용됩니다.

- 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 나열된 플릿 수입니다. Amazon GameLift Servers는 대기열의 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 섹션에는 대기열의 Amazon 리소스 이름(ARN) 및 제한 시간이 표시됩니다. Amazon GameLift Servers의 다른 작업 또는 영역에서 대기열을 참조할 때 ARN을 사용할 수 있습니다. 제한 시간은 게임 세션 배치 요청이 제한 시간 전에 대기열에 남아 있는 최대 시간입니다.

이벤트 알림

이벤트 알림 섹션에는 Amazon GameLift Servers가 이벤트 알림을 게시하는 SNS 주제 및 이 대기열에서 생성되는 모든 이벤트에 추가되는 이벤트 데이터가 나열됩니다.

Tags

태그 테이블에는 리소스에 태그를 지정하는 데 사용된 키와 값이 표시됩니다. 태그 지정에 대한 자세한 내용은 [AWS 리소스 태그 지정](#)을 참조하세요.

Metrics

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 [대기열에 대한 Amazon GameLift Servers 지표](#) 섹션을 참조하세요.

대상

대상 탭에는 대기열에 대해 나열된 모든 집합 또는 별칭이 표시됩니다.

Amazon GameLift Servers가 대상에서 새 게임 세션 호스팅에 사용할 수 있는 리소스를 검색할 때는 여기 나열된 기본 순서로 검색합니다. 나열된 첫 번째 대상에 용량이 있으면 Amazon GameLift Servers가 새 게임 세션을 거기에 배치합니다. 플레이어 지연 시간 데이터를 제공하여 개별 게임 세션 배치 요청이 기본 순서보다 우선하도록 할 수 있습니다. 이 데이터는 Amazon GameLift Servers에 플레이어 평균 대기 시간이 가장 낮은 가용 대상을 검색하도록 지시합니다. 대기열에 대한 자세한 내용은 [게임 세션 대기열 사용자 지정](#) 섹션을 참조하세요.

세션 배치

게임의 배치 프로세스를 사용자 지정하는 구성 설정입니다.

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.

우선순위

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.

AWS SDK

다음 AWS CLI 명령을 사용하여이 리소스에 대한 정보를 검색합니다.

- [DescribeGameSessionQueues](#)

Amazon GameLift Servers 호스팅으로 출시 준비

다음 체크리스트를 사용하여 게임 배포의 각 단계를 검증합니다. [중요]로 표시된 항목은 프로덕션 출시에 매우 중요합니다.

[Amazon GameLift Servers 콘솔](#)에서 사용할 수 있는 Amazon GameLift Servers 출시 질문지를 다운로드하여 작성합니다. Amazon GameLift Servers를 사용하는 모든 게임 개발자의 원활한 출시를 지원하기 위해 제출하신 정보를 바탕으로 향후 로드 테스트, 소프트 출시 및 정식 출시를 준비하는 데 도움이 되고자 합니다. 첫 번째 로드 테스트를 수행하기 최소 3개월 전에 질문지를 작성하고 제출할 수 있도록 하세요.

주제

- [게임 준비](#)
- [테스트 준비](#)
- [출시 준비](#)
- [출시 후 업데이트 계획](#)

게임 준비

- [중요] 호스팅 솔루션에 대한 모든 [개발 로드맵 단계](#)를 완료했고 통합 게임 서버, 게임 클라이언트를 위한 백엔드 서비스, 호스팅 플릿, 게임 세션 배치 방법(예: 대기열)을 포함하여 필요한 모든 구성 요소가 있는지 확인합니다.
- [중요] 실행 중에 게임 서버가 다른 AWS 리소스에 액세스할 수 있도록 하는 [AWS Identity and Access Management \(IAM\) 역할](#)을 생성합니다.
- [중요] 필요에 따라 다른 호스팅 리소스에 대한 장애 조치를 설계하고 구현합니다.
- 게임의 대기열과 플릿 구조를 고려하여 [대상 위치로의 플릿 롤아웃을 계획](#)합니다.
- CloudFormation 및를 사용하여 코드형 인프라(IaC)를 사용하여 [배포를 자동화합니다](#) AWS Cloud Development Kit (AWS CDK).
- Amazon CloudWatch 및 Amazon Simple Storage Service(S3)를 사용하여 [로그와 분석을 수집](#)합니다.

테스트 준비

- [중요] 라이브 환경이 프로덕션 요구 사항에 맞게 확장할 수 있도록 [Amazon GameLift Servers 서비스 할당량 및 기타 할당량 증가를 요청](#)합니다. AWS 서비스
- [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.
- 사용 중인 컴퓨팅 리소스가 각 컴퓨팅에서 동시에 실행하려는 서버 프로세스 수를 지원할 수 있는지 확인합니다.
- 처음에는 좀 더 보수적으로 [조정 정책을 조정](#)하고 필요하다고 생각하는 것보다 더 많은 유틸리티 용량을 제공합니다. 나중에 비용을 최적화할 수 있습니다. 유틸리티 용량이 20%인 대상 기반 크기 조정 정책을 사용하는 것을 고려합니다.
- FlexMatch에는 [지연 규칙](#)을 사용하여 지리적으로 가까운 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.
- 문제 또는 중단 시가 대응할 AWS 수 있도록 지원 계획 수준을 비즈니스 또는 엔터프라이즈로 높입니다.

출시 준비

- [중요] 모든 라이브 플릿에 대해 [플릿 보호 정책을 전체 보호로 설정](#)하여 규모를 축소해도 활성 게임 세션이 중단되지 않도록 합니다.
- [중요] 최소한으로 최대 예상 수요를 수용할 수 있을 만큼 높은 [최대 플릿 크기를 크게 설정](#)합니다. 예상치 못한 수요에 대비하여 최대 크기를 두 배로 늘리는 것이 좋습니다.
- 개발 팀 전체가 출시 이벤트에 참여하고 런칭 룸에서 게임 출시를 모니터링하도록 권장합니다.
- 플레이어 지연 시간과 플레이어 경험을 모니터링합니다.

출시 후 업데이트 계획

- 플레이어 사용량에 따라 유틸리티 용량을 최소화하도록 [규모 정책을 조정](#)합니다.
- 플레이어 지연 데이터 및 수정된 요구 사항에 따라 [FlexMatch 규칙을 수정](#)하거나 [호스팅 위치를 추가](#)합니다.
- 각 컴퓨팅 리소스에서 가능한 한 많은 게임 세션을 실행하도록 런타임 구성을 최적화합니다. 이러한 방식으로 성능 효율성을 극대화하면 동일한 컴퓨팅 리소스로 더 많은 서버 프로세스를 실행할 수 있기 때문에 플릿 비용에 직접적인 영향을 미칠 수 있습니다.

- [분석 데이터를 사용](#)하여 지속적인 개발을 주도하고, 플레이어 경험과 게임 수명을 개선하며, 수익 창출을 최적화합니다.

를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation

CloudFormation 를 사용하여 Amazon GameLift Servers 리소스를 관리할 수 있습니다. 에서 각 리소스를 모델링하는 템플릿을 CloudFormation생성한 다음 템플릿을 사용하여 리소스를 생성합니다. 리소스를 업데이트하려면 템플릿을 변경하고 CloudFormation 를 사용하여 업데이트를 구현합니다. 리소스를 스택 및 스택 세트라는 논리 그룹으로 구성할 수 있습니다.

CloudFormation 를 사용하여 Amazon GameLift Servers 호스팅 리소스를 유지 관리하면 AWS 리소스 세트를 보다 효율적으로 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.

에 대한 자세한 내용은 [AWS CloudFormation 사용 설명서를](#) CloudFormation참조하세요. Amazon GameLift Servers 리소스에 대한 템플릿 정보를 보려면 [Amazon GameLift Servers 리소스 유형 참조](#)를 참조하세요.

모범 사례

사용에 대한 자세한 지침은 AWS CloudFormation 사용 설명서의 [CloudFormation 모범 사례를](#) CloudFormation참조하세요. 또한 이러한 모범 사례는 Amazon GameLift Servers와 특별한 관련성이 있습니다.

- CloudFormation을 사용하여 리소스를 일관적으로 관리합니다. 리소스 외부에서 리소스를 변경하면 CloudFormation 리소스 템플릿과 동기화되지 않습니다.
- 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 대체하는 경우는 빌드에 연결된 플릿을 CloudFormation 대체합니다. CloudFormation 그런 다음 새 플릿을 가리키도록 기존 별칭을 업데이트합니다. 자세한 내용을 알아보려면AWS CloudFormation 사용 설명서의 [스택 작업](#)을 참조하세요.

- 여러 리전 또는 AWS 계정에 동일한 CloudFormation 스택을 배포하는 경우 스택 세트를 사용합니다. 자세한 내용을 알아보려면 AWS CloudFormation 사용 설명서의 [스택 세트 작업](#)을 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.

CloudFormation 스택 사용

Amazon GameLift Servers 리소스에 대한 CloudFormation 스택을 설정할 때 다음 구조를 사용하는 것이 좋습니다. 최적의 스택 구조는 게임을 한 리전에만 배포하는지 또는 여러 리전에 배포하는지에 따라 다릅니다.

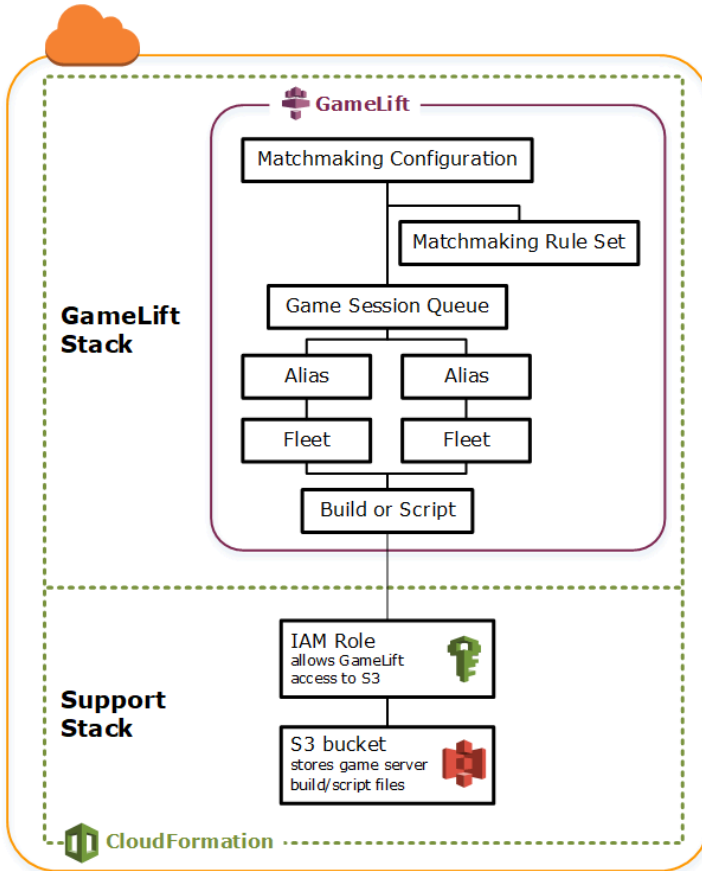
단일 위치용 스택

Amazon GameLift Servers 리소스를 단일 위치에서 관리하려면 2스택 구조를 사용하는 것이 좋습니다.

- 지원 스택 - 이 스택에는 Amazon GameLift Servers 리소스가 종속되는 리소스가 포함되어 있습니다. 최소한 이 스택에는 사용자 지정 게임 서버 또는 Realtime 스크립트 파일을 저장하는 S3 버킷이 포함되어야 합니다. 스택에는 Amazon GameLift Servers 빌드 또는 스크립트 리소스를 생성할 때 S3 버킷에서 파일을 검색할 수 있는 Amazon GameLift Servers 권한을 부여하는 IAM 역할도 포함되어야 합니다. 이 스택에는 DynamoDB 테이블, Amazon Redshift 클러스터 및 Lambda 함수와 같이 게임에 사용되는 다른 AWS 리소스도 포함될 수 있습니다.
- Amazon GameLift Servers 스택 - 이 스택에는 빌드 또는 스크립트, 플릿 세트, 별칭 및 게임 세션 대기열을 포함한 모든 Amazon GameLift Servers 리소스가 포함되어 있습니다. S3 버킷 위치에 저장된 파일로 빌드 또는 스크립트 리소스를 CloudFormation 생성하고 빌드 또는 스크립트를 하나 이상의 플릿 리소스에 배포합니다. 각 플릿에는 해당 별칭이 있어야 합니다. 게임 세션 대기열은 플릿 별

칭 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 보여줍니다.



여러 리전에 대한 스택

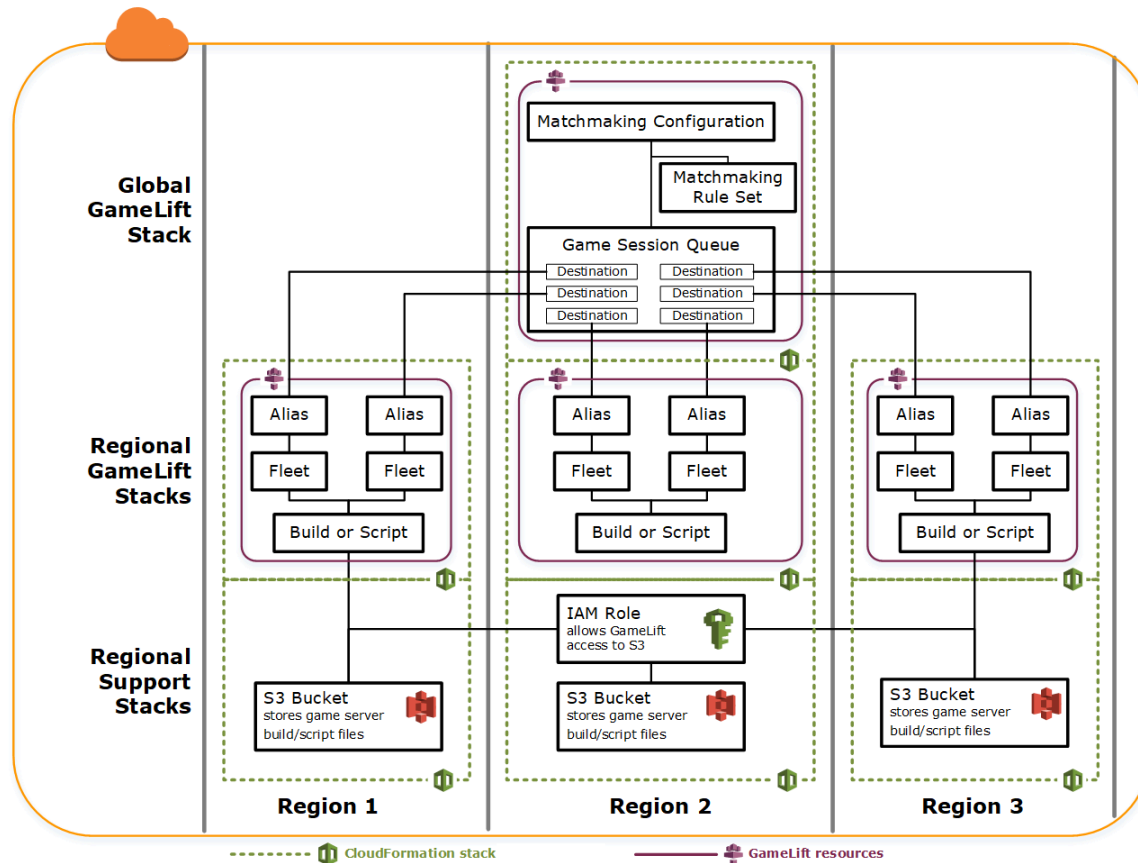
두 개 이상의 리전에 게임을 배포할 때는 리소스가 리전 간에 상호 작용할 수 있는 방식을 명심해야 합니다. Amazon GameLift Servers 플릿과 같은 일부 리소스는 동일한 리전의 기타 리소스만 참조할 수 있습니다. Amazon GameLift Servers 대기열과 같은 다른 리소스는 리전에 구매받지 않습니다. 여러 리전에서 Amazon GameLift Servers 리소스를 관리하려면 다음 구조를 사용하는 것이 좋습니다.

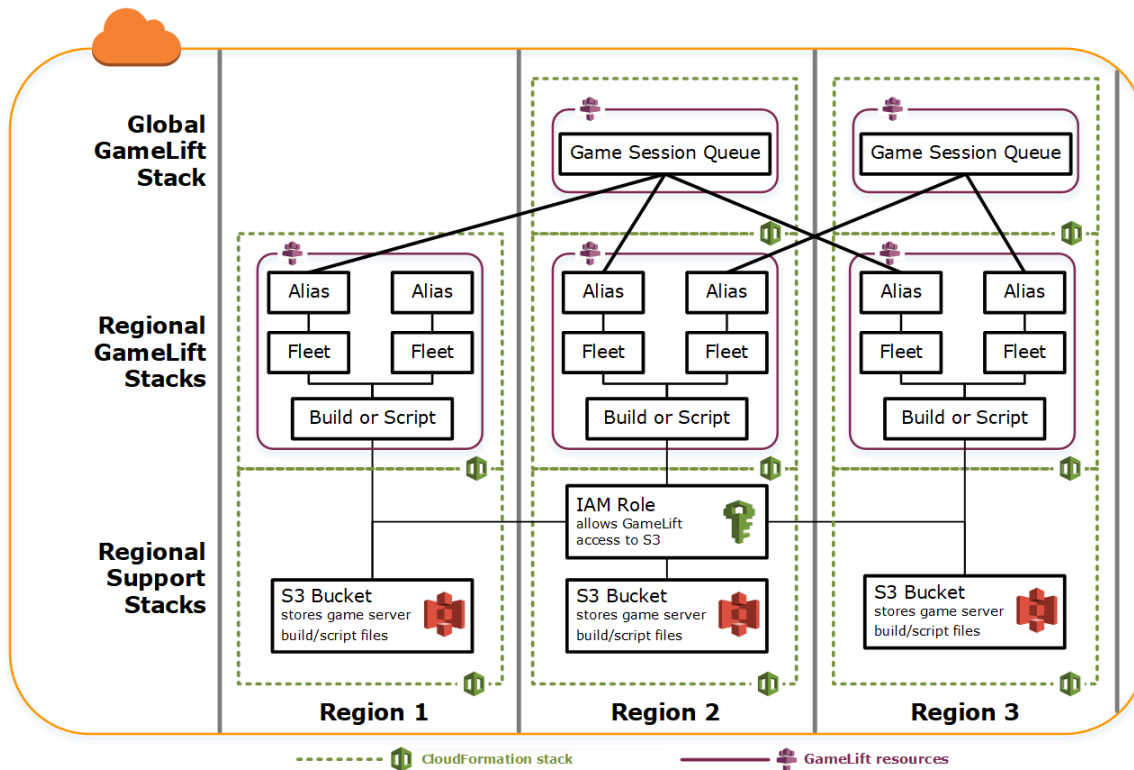
- 리전별 지원 스택 - 이 스택에는 Amazon GameLift Servers 리소스가 종속되는 리소스가 포함되어 있습니다. 이 스택에는 사용자 지정 게임 서버 또는 Realtime 스크립트 파일을 저장하는 S3 버킷이 포함되어야 합니다. DynamoDB 테이블, Amazon Redshift 클러스터 및 Lambda 함수와 같은 게임에 대한 다른 AWS 리소스도 포함될 수 있습니다. 이러한 리소스 중 대부분은 리전별로 특정하므로 모든 리전에서 리소스를 생성해야 합니다. Amazon GameLift Servers에는 이러한 지원 리소스에 액세스

스하도록 허용하는 IAM 역할도 필요합니다. IAM 역할은 리전과 무관하므로 어떠한 리전에도 배치되고 기타 모든 지원 스택에서 참조되는 역할 리소스 하나만 있으면 됩니다.

- 리전 Amazon GameLift Servers 스택 -이 스택에는 빌드 또는 스크립트, 집합 집합 및 별칭을 포함하여 게임이 배포되는 각 리전에 있어야 하는 Amazon GameLift Servers 리소스가 포함되어 있습니다. 는 S3 버킷 위치에 파일이 있는 빌드 또는 스크립트 리소스를 CloudFormation 생성하고 빌드 또는 스크립트를 하나 이상의 집합 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 Servers 스택 - 이 스택에는 게임 세션 대기열과 매치메이킹 리소스가 포함되어 있습니다. 이러한 리소스는 어떠한 리전에도 위치할 수 있으며 일반적으로 동일한 리전에 배치됩니다. 대기열은 어떠한 리전에도 위치하는 플릿 또는 별칭을 참조할 수 있습니다. 추가 대기열을 다른 리전에 배치하려면 추가 글로벌 스택을 생성합니다.

아래 다이어그램은 여러 AWS 리전에 리소스를 배포하기 위한 멀티스택 구조를 보여줍니다. 첫 번째 다이어그램은 단일 게임 세션 대기열의 구조를 보여 줍니다. 두 번째 다이어그램은 여러 개의 대기열이 있는 구조를 보여 줍니다.





빌드 업데이트

Amazon GameLift Servers 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 정보는 AWS CloudFormation 사용 설명서의 [스택 리소스의 업데이트 동작](#)을 참조하세요.

자동으로 빌드 업데이트 배포

관련 빌드, 플릿 및 별칭 리소스가 포함된 스택을 업데이트할 때 기본 CloudFormation 동작은 이러한 단계를 순서대로 자동으로 수행하는 것입니다. 먼저 새 빌드 파일을 새 S3 위치에 업로드하여 이 업데이트를 트리거합니다. 그런 다음 새 S3 위치를 가리키도록 CloudFormation 빌드 템플릿을 수정합니다. 스택을 새 S3 위치로 업데이트하면 다음 CloudFormation 시퀀스가 트리거됩니다.

1. S3에서 새 파일을 검색하고, 파일의 유효성을 검사한 다음, 새 Amazon GameLift Servers 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.

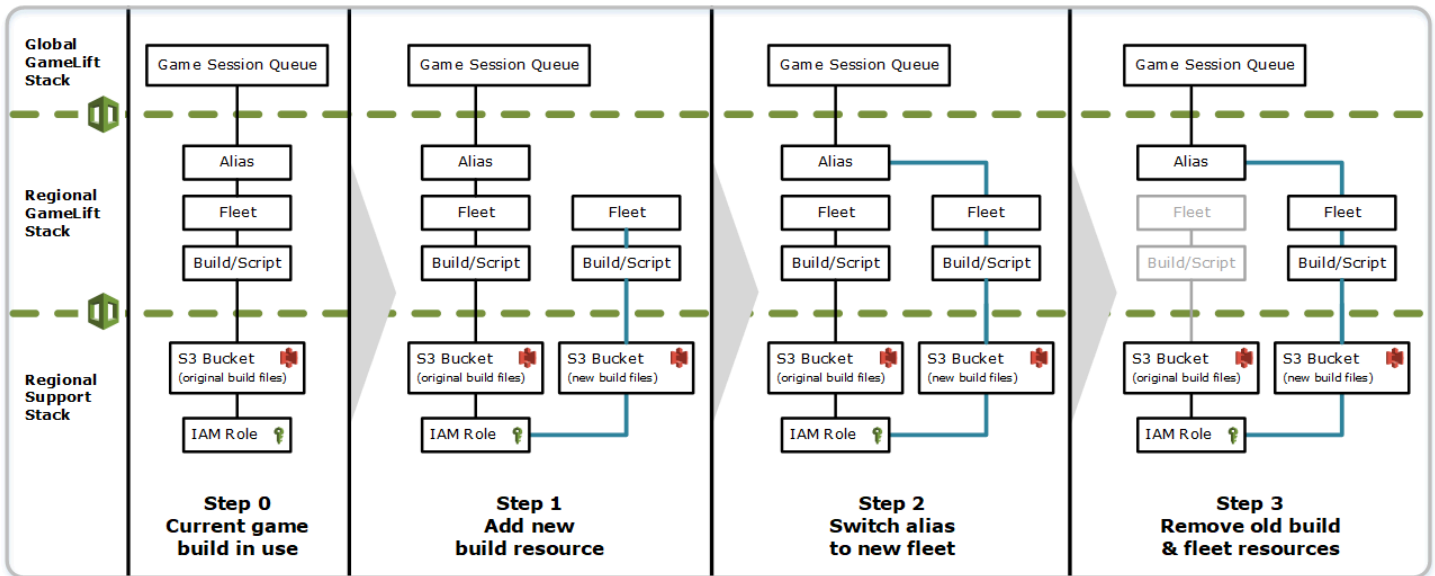
리소스를 삭제하는 대신 CloudFormation 보존할 수도 있습니다. 자세한 내용은 AWS CloudFormation API 참조의 [RetainResources](#)를 참조하세요.

수동으로 빌드 업데이트 배포

새 플릿이 플레이어를 위해 실행되는 시간을 더 세밀하게 제어하려는 경우 몇 가지 옵션을 선택할 수 있습니다. Amazon GameLift Servers 콘솔 또는 CLI를 사용하여 수동으로 별칭을 관리하도록 선택할 수 있습니다. 또는 빌드와 플릿을 교체하도록 빌드 템플릿을 업데이트하는 대신, 두 번째 빌드 및 플릿 정의의 세트를 템플릿에 추가할 수 있습니다. 템플릿을 업데이트하면가 두 번째 빌드 리소스와 해당 플릿을 CloudFormation 생성합니다. 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 CloudFormation 에서 자동으로 롤백이 시작됩니다. 이 프로세스는 각 단계를 순차적으로 반전하여 새로 생성된 리소스를 삭제합니다.

롤백을 수동으로 트리거해야 하는 경우 빌드 템플릿의 S3 위치 키를 다시 원래 위치로 변경하고 스택을 업데이트합니다. 새 Amazon GameLift Servers 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 계속하기](#)를 참조하세요.

를 사용한 DDoS 보호 Amazon GameLift Servers

Amazon GameLift Servers 플레이어 게이트웨이는에서 호스팅되는 게임 클라이언트와 게임 서버 간에 UDP 트래픽을 라우팅하는 릴레이 기반 네트워크입니다Amazon GameLift Servers. 플레이어 게이트웨이는 게임 서버에 도달하기 전에 트래픽을 검증하고, 플레이어 트래픽을 제한하고, 퍼블릭에서 게임 서버 IP 주소를 숨기고, 릴레이 엔드포인트가 비정상일 때 업데이트된 엔드포인트를 제공하여 선제적 DDoS 보호를 제공합니다. 이 기능은 추가 지연 시간이 거의 없는 볼륨 측정 공격으로부터 게임 서버를 보호하는 데 도움이 됩니다.

플레이어 게이트웨이를 사용하려면 Linux 기반 Amazon GameLift Servers 관리형 EC2 플릿 또는 컨테이너 플릿에서 게임 서버를 실행해야 합니다. 플릿 생성 중에 플레이어 게이트웨이를 활성화하고이 기능을 사용하려면 게임 클라이언트와 게임 백엔드를 업데이트해야 합니다.

주요 이점

플레이어 게이트웨이는 다음과 같은 보안 및 운영 이점을 제공합니다.

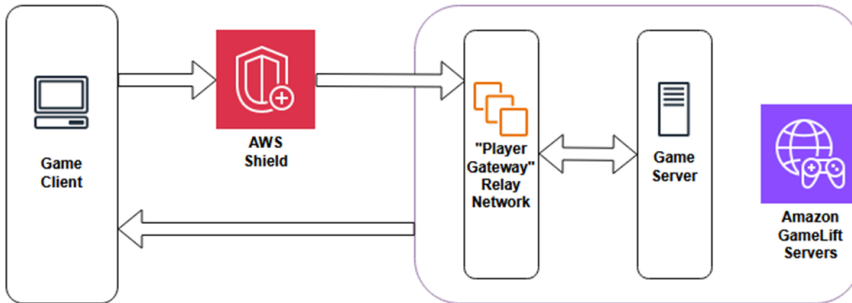
- 게임 서버 IP 주소 숨기기 - 게임 클라이언트는 게임 서버에 직접 연결하는 대신 릴레이 엔드포인트를 통해 연결하여 게임 서버 주소를 퍼블릭에서 숨깁니다.
- 트래픽 검증 - 플레이어 게이트웨이를 통한 모든 트래픽에는 플레이어 게이트웨이 토큰이 필요하므로 인증된 플레이어의 트래픽만 게임 서버에 도달할 수 있습니다.
- 동적 엔드포인트 교체 - 릴레이 엔드포인트가 비정상일 되면는 엔드포인트를 Amazon GameLift Servers 빠르게 교체하고 다음 호출 시 업데이트된 정상 엔드포인트를 제공하여 플레이어 연결 세부 정보를 새로 고칩니다.
- 플레이어 트래픽 분산 - 릴레이 엔드포인트는 플레이어마다 다르므로 동일한 게임 세션 내에서 단일 비정상 릴레이 엔드포인트가 다른 플레이어에게 미치는 영향이 줄어듭니다.

플레이어 게이트웨이 작동 방식

플레이어 게이트웨이는 릴레이 네트워크를 사용하여 게임 클라이언트와 게임 서버 간에 UDP 트래픽을 라우팅합니다. 이렇게 하면 게임 서버에 도달하기 전에 트래픽을 검증하고, 플레이어 트래픽을 제한하고, 게임 서버 IP 주소를 퍼블릭에서 숨기고, 릴레이 엔드포인트가 비정상일 때 업데이트된 엔드포인트를 제공하여 DDoS 공격으로부터 보호됩니다.

트래픽 흐름

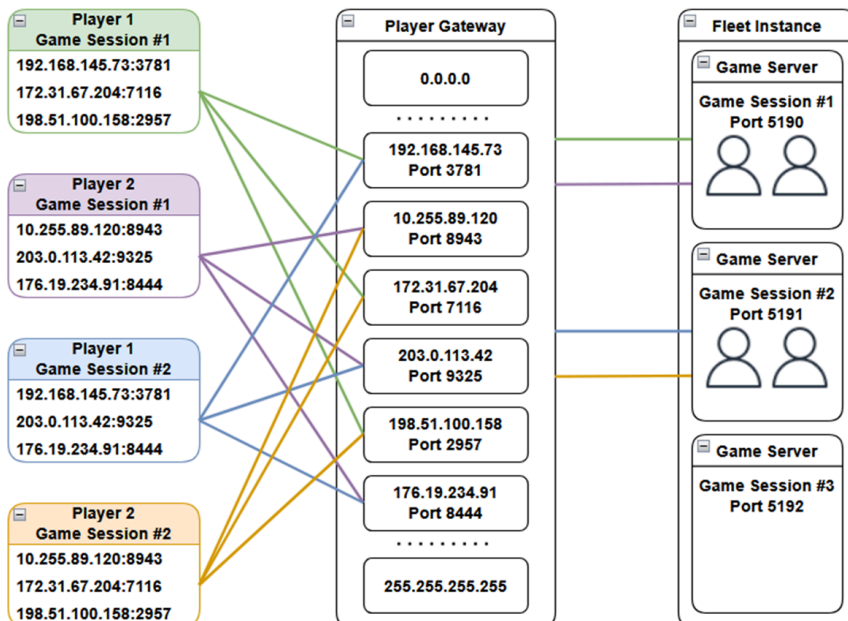
플레이어가 게임 세션에 연결되면 게임 백엔드는 GetPlayerConnectionDetails API에서 릴레이 엔드포인트와 플레이어 게이트웨이 토큰을 검색하여 게임 클라이언트로 전송합니다. 게임 클라이언트는 플레이어 게이트웨이 토큰을 UDP 패킷에 프리펜딩하고 패킷을 릴레이 엔드포인트로 전송합니다. 릴레이 네트워크는 토큰을 검증하고 합법적인 트래픽을 게임 서버로 라우팅합니다. 전송 전에 릴레이 네트워크는 플레이어 게이트웨이 토큰을 제거하여 게임 서버가 원시 게임 클라이언트 페이로드를 수신하고 코드 변경이 필요하지 않도록 합니다. 게임 서버에서 게임 클라이언트로의 통신은 수정 없이 릴레이 네트워크를 통해 반환됩니다.



핵심 개념

릴레이 엔드포인트

릴레이 엔드포인트는 게임 클라이언트가 플레이어 게이트웨이를 통해 트래픽을 전송하는 데 사용하는 IP 주소 및 포트 조합입니다. 각 플레이어는 서로 다른 여러 엔드포인트를 수신하여 트래픽을 분산하고 동일한 게임 세션에서 다른 플레이어에 대한 공격의 영향을 줄입니다.



플레이어 게이트웨이 토큰

플레이어 게이트웨이 토큰은 플레이어가 플레이어 게이트웨이를 통해 게임 세션으로 트래픽을 전송하도록 승인하는 암호화된 바이트 배열입니다. `GetPlayerConnectionDetails` API는 토큰을 base64로 인코딩된 문자열로 반환합니다. 게임 클라이언트는 플레이어 게이트웨이 토큰 앞에 모든 UDP 패킷을 추가해야 합니다. 릴레이 네트워크는 토큰을 검증하고 유효하지 않거나 누락된 토큰이 있는 패킷을 거부합니다.

Important

플레이어 게이트웨이 토큰은 암호화해서는 안 되며 게임 클라이언트가 전송한 각 UDP 패킷의 시작 부분에 나타나야 합니다. 게임이 페이로드를 암호화하는 경우 게임 데이터를 암호화한 후 수정되지 않은 플레이어 게이트웨이 토큰 앞에 추가합니다.

플레이어 게이트웨이 토큰은 최소 3분 동안 유효합니다. 릴레이 엔드포인트가 비정상일 때 플레이어가 업데이트된 엔드포인트를 수신하도록 60초마다 토큰을 새로 고치는 것이 좋습니다.

GetPlayerConnectionDetails API

게임 백엔드는 `GetPlayerConnectionDetails` API를 호출하여 게임 세션의 플레이어에 대한 연결 세부 정보를 검색합니다. API는 릴레이 엔드포인트와 플레이어 게이트웨이 토큰을 반환하거나 직접 연결을 위해 게임 서버의 IP 주소 및 포트로 돌아갑니다. 게임 클라이언트는 두 연결 유형을 모두 처리하도록 설계되어야 합니다. 릴레이 엔드포인트가 비정상일 때 업데이트된 엔드포인트를 수신하려면 게임 세션 전체에서 주기적으로 이 API를 호출합니다(60초마다 권장).

자세한 내용은 [GetPlayerConnectionDetails API](#) 단원을 참조하십시오.

플레이어 게이트웨이 모니터링

플레이어 게이트웨이는 Amazon CloudWatch에 지표를 게시하여 네트워크 트래픽 패턴을 모니터링하고, 잠재적 DDoS 공격을 식별하고, 릴레이 성능을 추적하는 데 도움이 됩니다. 지표에는 패킷 및 바이트 입/출력, 제한된 트래픽 및 플레이어 세션이 포함됩니다. 플레이어 게이트웨이 지표의 전체 목록은 섹션을 참조하세요 [DDoS 보호\(플레이어 게이트웨이\) 지표](#).

IPv4 및 IPv6 호환성

게임 클라이언트는 IPv4를 사용하여 통신합니다. 플레이어 게이트웨이는 IPv6를 사용하여 게임 서버와 통신합니다.는 플릿 구성에 따라 IPv4와 IPv6 간의 변환을 Amazon GameLift Servers 자동으로 처리합니다.

플릿에서 플레이어 게이트웨이를 구성하는 방법에 대한 자세한 내용은 [CreateFleet](#) 또는 [CreateContainerFleet](#)을 참조하세요. IPv4 및 IPv6 지원에 대한 자세한 내용은 섹션을 참조하세요 [IPv4 및 IPv6 호환성](#).

플릿에서 플레이어 게이트웨이 활성화

플릿을 생성할 때 플레이어 게이트웨이를 활성화하여 들어오는 UDP 트래픽을 검증하고 게임 서버로 속도를 제한하는 릴레이 엔드포인트를 통해 게임 트래픽을 라우팅합니다.

사전 조건

플레이어 게이트웨이를 활성화하기 전에 다음 요구 사항을 확인합니다.

- 지원되는 위치 - 플레이어 게이트웨이를 지원하는 위치 목록은 AWS 섹션을 참조하세요 [Amazon GameLift Servers 서비스 위치](#).
- 플릿 유형 - 플레이어 게이트웨이는 Linux 기반 관리형 EC2 플릿 및 관리형 컨테이너 플릿을 지원합니다.
- Server SDK - 플레이어 게이트웨이에는 Amazon GameLift Servers Server SDK 5.0 이상이 필요합니다.

플레이어 게이트웨이 모드

PlayerGatewayMode 속성을 사용하여 플릿에 대한 플레이어 게이트웨이 호환성을 구성합니다.

비활성화됨(기본값)

플릿은 플레이어 게이트웨이와 호환되지 않습니다.

ENABLED

플릿은 지원되는 위치에서 플레이어 게이트웨이와 호환됩니다. 기능을 지원하지 않는 플릿 위치는 여전히 플릿에 추가할 수 있지만 게임 서버 IP 주소를 직접 활용하여 플레이어 게이트웨이를 활용하지는 않습니다.

필수

플릿은 플레이어 게이트웨이와 호환됩니다. 기능을 지원하지 않는 플릿 위치는 플릿에 추가할 수 없습니다.

IPv4 및 IPv6 호환성

게임 클라이언트는 IPv4를 통해 플레이어 게이트웨이와 통신해야 합니다. 플레이어 게이트웨이 릴레이 네트워크는 IPv6를 사용하여 게임 서버와 통신합니다. 관리형 컨테이너 플릿은 IPv6 트래픽을 IPv4로 자동 변환합니다. 관리형 EC2 플릿의 경우의 `GameServerIpProtocolSupported` 속성을 사용하여 플릿이 IPv6 트래픽을 처리하는 방법을 구성합니다 `PlayerGatewayConfiguration`.

IPv4(기본값)

게임 서버는 수신 IPv4 트래픽만 허용합니다. 인스턴스에 경량 IP 변환 소프트웨어가 설치되고 실행되어 수신 IPv6 트래픽을 수신하고 IPv4로 변환합니다.

이중 스택

게임 서버는 수신 IPv6 트래픽과 호환됩니다. IP 번역 소프트웨어는 인스턴스에 설치되거나 실행되지 않습니다. IPv6를 기본적으로 지원하는 게임 서버는 성능이 약간 더 높을 수 있습니다.

플레이어 게이트웨이를 게임에 통합

플레이어 게이트웨이 지원 플릿을 생성한 후 플레이어 게이트웨이와 통합되도록 게임 클라이언트 및 게임 백엔드를 업데이트해야 합니다. 게임 백엔드는 릴레이 엔드포인트와 플레이어 게이트웨이 토큰을 검색한 다음 게임 클라이언트에 제공합니다. 게임 클라이언트는 모든 UDP 트래픽을 이러한 릴레이 엔드포인트로 전송합니다.

통합은 다음 호출 경로를 따릅니다.

1. 게임 백엔드는 `GetPlayerConnectionDetails` API를 호출하여 게임 세션의 각 플레이어에 대한 릴레이 엔드포인트 및 플레이어 게이트웨이 토큰을 검색합니다.
2. 게임 백엔드는 릴레이 엔드포인트와 플레이어 게이트웨이 토큰을 게임 클라이언트로 전송합니다.
3. 게임 클라이언트는 플레이어 게이트웨이 토큰을 모든 발신 UDP 패킷에 우선 적용하고 패킷을 릴레이 엔드포인트로 전송합니다.
4. 릴레이 네트워크는 플레이어 게이트웨이 토큰을 검증하고 트래픽을 게임 서버로 라우팅합니다.
5. 게임 서버는 동일한 릴레이 경로를 통해 트래픽을 게임 클라이언트로 다시 전송합니다.

백엔드 통합

게임 백엔드는 GetPlayerConnectionDetails API를 호출하여 플레이어에 대한 릴레이 엔드포인트 및 플레이어 게이트웨이 토큰을 검색해야 합니다. 그러면 백엔드가 게임 클라이언트에 이 정보를 제공합니다.

GetPlayerConnectionDetails API

GetPlayerConnectionDetails API는 게임 세션의 플레이어에 대한 연결 세부 정보를 반환합니다.

릴레이 엔드포인트 및 토큰

위치에서 플레이어 게이트웨이가 활성화되고 지원되는 경우는 다음을 반환합니다.

- 릴레이 엔드포인트 - 플레이어마다 다른 여러 릴레이 엔드포인트(IP 주소 및 포트)
- 플레이어 게이트웨이 토큰 - 클라이언트가 모든 UDP 패킷 앞에 추가해야 하는 토큰(최소 3분 동안 유효)
- 만료 - 플레이어 게이트웨이 토큰 만료 타임스탬프

직접 연결

플레이어 게이트웨이가 활성화되지 않았거나 위치에서 지원되지 않는 경우는 게임 서버의 IP 주소 및 포트를 반환합니다.

게임 클라이언트는 두 연결 유형을 모두 처리하도록 설계되어야 합니다.

API 호출 예제:

```
// C++ example using AWS SDK
Aws::GameLift::GameLiftClient client;
Aws::GameLift::Model::GetPlayerConnectionDetailsRequest request;
request.SetGameSessionId(gameSessionId);
request.SetPlayerIds(playerIds); // Vector of player IDs

auto outcome = client.GetPlayerConnectionDetails(request);
if (outcome.IsSuccess()) {
    auto result = outcome.GetResult();
    auto connectionDetails = result.GetPlayerConnectionDetails();

    // Process each player's connection details
    for (const auto& detail : connectionDetails) {
```

```

        std::string playerId = detail.GetPlayerId();

        // Get relay endpoints (IP address and port)
        auto endpoints = detail.GetEndpoints();
        for (const auto& endpoint : endpoints) {
            std::string ipAddress = endpoint.GetIpAddress();
            int port = endpoint.GetPort();
        }

        // Get player gateway token
        auto token = detail.GetPlayerGatewayToken();

        // Get expiration time
        auto expiration = detail.GetExpiration();

        // Send endpoints and token to game client
    }
}

```

API 참조의 [GetPlayerConnectionDetails](#)를 참조하세요. Amazon GameLift Servers

릴레이 엔드포인트 새로 고침

릴레이 엔드포인트가 비정상이 되면 `GetPlayerConnectionDetails` 주기적으로 호출하여 플레이어에게 업데이트된 엔드포인트를 제공합니다. 게임 세션의 모든 플레이어에 대해 60초마다 API를 호출하는 것이 좋습니다. API 사용을 최적화하려면 단일 API 호출에서 여러 플레이어 IDs를 일괄 처리합니다.

Important

정기적인 새로 고침 호출은 업데이트된 엔드포인트를 플레이어에게 전달하는 기본 메커니즘입니다. 플레이어 게이트웨이 토큰은 최소 3분 동안 유효하지만 60초마다 새로 고치면 플레이어가 업데이트된 엔드포인트를 정기적으로 수신할 수 있습니다.

클라이언트 통합

게임 클라이언트는 모든 발신 UDP 패킷에 플레이어 게이트웨이 토큰을 추가하고 제공된 릴레이 엔드포인트로 패킷을 전송해야 합니다. Amazon GameLift Servers 클라이언트 SDK for C++ 및 Unreal Engine은 이러한 통합을 간소화하는 유틸리티를 제공합니다.

클라이언트 요구 사항

플레이어 게이트웨이를 통해 트래픽을 라우팅하려면 게임 클라이언트가 다음을 수행해야 합니다.

- 플레이어 게이트웨이 토큰 준비 - 모든 발신 UDP 패킷에 플레이어 게이트웨이 토큰을 준비합니다. 플레이어 게이트웨이 토큰은 암호화해서는 안 되며 각 패킷의 시작 부분에 나타나야 합니다. 유효한 플레이어 게이트웨이 토큰이 없는 패킷은 삭제됩니다.
- 릴레이 엔드포인트로 전송 - 제공된 릴레이 엔드포인트로 모든 UDP 패킷을 전송합니다.
- 연결 활동 유지 - 게임 클라이언트가 게임 서버로 패킷을 전송하거나 게임 서버가 플레이어로 패킷을 전송하는지 최소 30초에 한 번씩 확인합니다. 이 활동은 릴레이 네트워크를 통해 연결을 유지합니다.
- 엔드포인트 업데이트 처리 - 백엔드에서 업데이트된 릴레이 엔드포인트 및 플레이어 게이트웨이 토큰을 수락하고(60초마다 권장) 연결을 끊지 않고 새 엔드포인트로 전환합니다.

클라이언트 SDK

C++ 및 Unreal Engine용 Amazon GameLift Servers 클라이언트 SDK는 플레이어 게이트웨이 통합을 간소화하는 유틸리티를 제공합니다.

- 토큰 관리 - 모든 발신 UDP 패킷에 플레이어 게이트웨이 토큰을 준비합니다.
- 엔드포인트 선택 - 구성 가능한 알고리즘을 사용하여 트래픽을 릴레이 엔드포인트로 라우팅합니다.
- 엔드포인트 새로 고침 - 정기적인 콜백을 예약하여 백엔드에서 업데이트된 릴레이 엔드포인트 및 플레이어 게이트웨이 토큰을 검색합니다.

엔드포인트 선택 알고리즘

클라이언트 SDK는 사용할 릴레이 엔드포인트를 선택하기 위한 두 가지 기본 제공 알고리즘을 제공합니다.

폴백 알고리즘

비정상 상태가 될 때까지 하나의 엔드포인트를 사용한 다음 다른 엔드포인트로 전환합니다. 짧은 중단이 허용되는 메뉴, 로비 및 턴 기반 게임에 가장 적합합니다. 엔드포인트 장애 조치 중에 구성된 제한 시간(기본값: 2초) 동안 패킷이 손실될 수 있습니다.

예측 회전 알고리즘

사용 가능한 모든 엔드포인트를 지속적으로 교체하고 장애가 발생하기 전에 예측합니다. 일관된 패킷 전송이 중요한 1인칭 슈팅 게임 및 레이싱 게임과 같은 실시간 게임 플레이에 가장 적합합니다. 게임 서버가 일관된 빈도로 메시지를 전송해야 합니다.

SDK의 기본 알고리즘 클래스를 확장하여 사용자 지정 알고리즘을 구현할 수도 있습니다.

클라이언트 SDK 리소스

전체 통합 지침 및 샘플 코드는 다음 리소스를 참조하세요.

C++

클라이언트 SDK for C++는 [Amazon GameLift Servers 클라이언트 SDK for C++](#)에서 사용할 수 있습니다.

Unreal Engine

Unreal Engine용 플러그인은 [Amazon GameLift Servers Unreal Engine용 클라이언트 SDK](#)에서 사용할 수 있습니다. 액세스하려면 GitHub의 Epic Games 조직에 가입해야 합니다. 자세한 내용은 [GitHub의 Unreal Engine](#)을 참조하세요.

통합 테스트

플릿에 배포하기 전에 플레이어 게이트웨이 테스트 도구를 사용하여 플레이어 게이트웨이 통합을 로컬에서 테스트합니다. 이 도구는 플레이어 게이트웨이 UDP 프록시 인프라를 시뮬레이션하고 게임 클라이언트가 플레이어 게이트웨이 토큰을 올바르게 우선시하고, 여러 엔드포인트를 통해 트래픽을 라우팅하고, 네트워크 성능 저하를 처리하는지 확인하는 데 도움이 됩니다.

설정 및 사용 지침은 GitHub의 [플레이어 게이트웨이 테스트 앱을](#) 참조하세요.

모범 사례

플레이어 게이트웨이를 통합할 때 다음 모범 사례를 따르세요.

- 60초마다 엔드포인트 새로 고침 - 플레이어가 업데이트된 엔드포인트를 정기적으로 수신하도록 60초GetPlayerConnectionDetails마다를 호출합니다.
- 배치 API 호출 - 동일한 게임 세션에서 여러 플레이어GetPlayerConnectionDetails를 호출할 때 호출을 함께 배치 처리하여 API 오버헤드를 줄입니다.

- 연결 활동 유지 - 최소 30초에 한 번씩 게임 클라이언트와 게임 서버 간에 트래픽이 흐르는지 확인합니다. 이는 client-to-server 또는 server-to-client 트래픽일 수 있습니다.
- 엔드포인트 업데이트 정상 처리 - 게임 클라이언트가 백엔드에서 업데이트된 엔드포인트를 수신하면 플레이어의 연결을 끊지 않고 새 엔드포인트로 전환합니다.
- 올바른 알고리즘 선택 - 게임 요구 사항에 가장 적합한 엔드포인트 선택 알고리즘을 선택합니다. 턴 기반 게임에는 폴백 알고리즘을 사용하고 실시간 게임에는 예측 회전 알고리즘을 사용합니다.

Amazon GameLift Servers 테스트 및 문제 해결

이 섹션에서는 Amazon GameLift 호스팅 솔루션 관련 문제를 진단하고 해결하기 위한 지침을 제공합니다. 개발, 배포 또는 프로덕션 작업 중에 문제가 발생하는 경우 이러한 리소스는 근본 원인을 식별하고 효과적인 솔루션을 구현하는 데 도움이 됩니다.

- 개발 및 테스트 환경 - 프로덕션 조건을 미러링하는 반복적인 개발 워크플로를 설정하여 개발 주기 초기에 통합 문제를 파악하고 해결할 수 있습니다.
- 플릿 배포 및 관리 - 게임 서버가 제대로 실행되지 못하게 할 수 있는 일반적인 플릿 생성 실패, 서버 프로세스 활성화 문제, 규모 조정 문제, 배포 오류를 진단합니다.
- 런타임 진단 및 모니터링 - 실시간 문제 해결, 성능 분석, 구성 검증을 위해 라이브 플릿 인스턴스에 액세스하여 플레이어 경험에 영향을 미치는 문제를 해결합니다.

Amazon GameLift의 모니터링 및 로깅 기능과 함께 이러한 리소스를 사용하여 멀티플레이어 게임의 최적의 성능과 가용성을 유지합니다.

Amazon GameLift Servers Anywhere로 반복 개발을 위한 설정

Amazon GameLift Servers는 게임 개발 중에 사용할 호스팅 테스트 환경을 설정하는 데 도움이 되는 도구와 솔루션을 제공합니다. 이러한 도구를 사용하면 Amazon GameLift Servers를 사용한 관리형 호스팅의 실제 플레이어 경험을 미러링하고 빠르고 반복적인 개발 프로세스를 지원하는 환경을 만들 수 있습니다.

별도의 테스트 환경을 사용하면 테스트 중에 Amazon GameLift Servers 관리형 플릿의 오버헤드를 제거할 수 있습니다. 더 이상 각 새 게임 서버 빌드 반복을 업로드하고 새 플릿을 생성한 다음 활성화할 때까지 15분 이상 기다릴 필요가 없습니다. 대신 새 빌드를 생성하고, 새 빌드로 테스트 플릿을 빠르게 업데이트하고, 시작하고, 테스트를 시작할 수 있습니다.

Amazon GameLift Servers Anywhere 플릿을 사용하면 개발 워크스테이션과 같은 로컬 디바이스를 사용하여 테스트 환경을 설정할 수 있습니다. 클라우드 기반 호스팅 리소스를 사용하여 테스트 환경을 설정할 수도 있습니다.

Anywhere 테스트 환경을 설정하여 다음을 포함한 다양한 시나리오를 개발하고 테스트합니다.

- Amazon GameLift Servers 서버 SDK로 게임 서버 통합을 테스트합니다. AWS CLI 호출을 사용하여 새 게임 세션을 시작하고 게임 세션 이벤트를 추적하여 작동 중인 게임 클라이언트 없이도 테스트할 수 있습니다.

- 게임 클라이언트, 백엔드 서비스 및 Amazon GameLift Servers 서비스 간의 상호 작용을 테스트하여 게임용 구성 요소를 개발할 수 있습니다. 게임에 참여하기 위한 플레이어 환경을 미세 조정합니다.
- FlexMatch 매치메이커 설계로 다양한 실험이 가능합니다. 규칙 세트 변형 및 기타 매치메이킹 기능 구현을 사용해 보세요. 매치메이킹 채우기를 설정하고 테스트합니다.
- 게임 서버 수명 주기 관리를 위한 런타임 구성 설정(Amazon GameLift Servers 에이전트 사용)과 같은 다른 Amazon GameLift Servers 호스팅 기능을 사용해 보세요.
- 멀티플레이어 상호 작용을 포함하여 게임 플레이어 경험의 모든 면을 라이브 호스팅 환경에서 빠르게 빌드, 테스트 및 반복하여 검증합니다.

나중에 게임 출시를 준비할 때 Amazon GameLift Servers 관리형 플릿을 추가하여 호스팅 구성을 미세 조정하고 다음을 포함한 추가 시나리오를 테스트해야 합니다.

- 다중 위치 플릿, 스팟 및 온디맨드 플릿, 여러 인스턴스 유형의 사용을 포함하여 게임 세션 대기열 설계를 실험하고 테스트합니다.
- 선택적 지연 시간 정책 및 플릿 우선 순위 설정 사용을 포함하여 관리형 플릿을 사용하여 게임 세션 배치 옵션을 사용해 보세요.
- 자동 또는 수동 조정 옵션을 사용하여 플레이어 수요에 맞게 용량 조정을 구성합니다.
- Amazon GameLift Servers 관리형 플릿 AWS CloudFormation 으로 설정하여 호스팅 리소스를 장기 관리합니다.

Fast Build Update Tool(개발 전용)

관리형 EC2 플릿을 사용하면 게임 서버 빌드 업데이트를 배포할 때 각 신규 빌드를 Amazon GameLift Servers에 업로드하고 이를 위한 새 플릿을 생성해야 합니다.

Fast Build Update Tool을 사용하면 개발 중에 이러한 단계를 우회하여 시간을 절약하고 더 빠른 개발 반복을 수행할 수 있습니다. 이 도구를 사용하면 기존 플릿의 모든 컴퓨팅에서 게임 빌드 파일을 빠르게 업데이트할 수 있습니다. 이 도구에는 여러 옵션이 있습니다. 전체 게임 빌드를 교체하거나 6개의 특정 파일을 변경할 수 있으며 업데이트 후 게임 서버 프로세스를 다시 시작하는 방법을 관리할 수 있습니다. 플릿의 개별 컴퓨팅을 업데이트하는 데도 사용할 수 있습니다.

Fast Build Update Tool을 다운로드하고 사용 방법에 대해 자세히 알아보려면 GitHub의 [The Fast Build Update Tool](#)에 대한 Amazon GameLift Servers Toolkit 리포지토리를 참조하세요.

주제

- [클라우드 기반 테스트 환경 빌드](#)
- [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)
- [Amazon GameLift Servers Local을 사용하여 통합 테스트](#)

클라우드 기반 테스트 환경 빌드

Note

이 주제에서는 Amazon GameLift Servers용 서버 SDK 버전 5.x와 통합된 게임에 대한 반복 테스트를 다룹니다. 게임에서 서버 SDK 버전 4.x 이하를 사용하는 경우 [Amazon GameLift Servers Local을 사용하여 통합 테스트](#) 페이지를 참조하세요.

Amazon GameLift Servers Anywhere 플릿을 사용하여 클라우드 기반 호스팅 환경에서 게임 구성 요소를 반복적으로 빌드하고 테스트합니다. 호스팅 리소스와 Amazon GameLift Servers 서비스에 대한 연결을 사용하여 Anywhere 플릿을 생성하고, 해당 플릿에서 게임 서버를 실행하고, 필요에 따라 게임 기능을 테스트합니다.

Amazon GameLift Servers 에이전트를 사용하여 Anywhere 플릿 배포

게임 서버 빌드가 Amazon GameLift Servers SDK 5.x 이상과 통합된 경우 Amazon GameLift Servers 에이전트를 사용하여 클라우드 기반 Anywhere 플릿에 배포할 수 있습니다. 에이전트는 플릿의 각 컴퓨팅에서 게임 서버 수명 주기 및 기타 작업을 관리하는 백그라운드 프로세스입니다. 이러한 작업에는 Anywhere 플릿에 컴퓨팅 등록, 인증 토큰 획득, 일련의 지침에 따른 게임 서버 프로세스 시작/종지가 포함됩니다. 에이전트는 플릿의 런타임 구성에 의해 제어되며, 플릿 수명 동안 언제든지 구성을 업데이트할 수 있습니다. (에이전트는 관리형 EC2 플릿에 자동으로 배포됩니다.) 자세한 내용을 알아보고 에이전트를 다운로드하려면 [Amazon GameLift Servers GitHub 리포지토리](#)를 방문하세요.

Amazon EC2를 사용한 반복 테스트 설정

본 [Amazon GameLift Servers 툴킷 솔루션](#)의 안내형 워크플로를 사용하여 Amazon GameLift Servers의 관리형 호스팅 환경을 미러링하는 클라우드 기반 호스팅 환경을 설정합니다.

GitHub 리포지토리는 Amazon GameLift Servers Anywhere 및 Amazon GameLift Servers 에이전트를 사용하여 테스트 환경을 설정하는 대부분의 프로세스를 자동화하는 스크립트 세트를 제공합니다. 또한 테스트할 새 게임 서버 빌드가 있을 때마다 환경을 업데이트하는 지침을 제공합니다. 샘플 게임 서버 빌드로 테스트 환경을 배포하는 단일 스크립트를 실행하거나 각 단계를 안내하여 자체 게임 서버 빌드로 설정할 수 있습니다.

이 워크플로에서는 AWS Management Console 사용하여 스크립트 AWS CloudShell 를 실행하고 명령줄 작업을 완료하는 방식으로에서 완전히 작업합니다.

 Note

이 자습서의 작업의 경우 ,, Amazon S3 Amazon GameLift Servers AWS CloudShell, AWS Systems Manager, Amazon EC2 및 서비스에 대한 권한이 있는 AWS 계정 사용자가 필요합니다 AWS Identity and Access Management. AWS 계정에 대한 관리자 수준의 액세스 권한이 있는 사용자에게는 이미 필요한 권한이 있습니다.

워크플로에는 다음 작업이 포함됩니다.

- Amazon GameLift Servers를 위해 게임 서버 빌드를 패키징합니다. 워크플로는 Amazon GameLift Servers용 서버 SDK 버전 5.x와 이미 통합되어 호스팅할 준비가 된 샘플 C++ 게임 서버를 빌드하는 스크립트를 제공합니다. 또는 통합을 완료한 경우에 자체 게임 프로젝트로 작업할 수 있습니다.
- Amazon Simple Storage Service 버킷을 설정하여 게임 서버 빌드 및 종속성을 저장합니다. 게임 빌드의 새 버전을 생성할 때 S3에 저장하고 스크립트를 사용하여 게임 테스트를 위한 Anywhere 플릿을 업데이트할 수 있습니다.
- Amazon GameLift Servers 에이전트를 가져와 빌드합니다. 에이전트가 구성에 따라 호스팅 리소스에서 게임 서버 프로세스를 관리합니다. 동일한 로직을 사용하고 Amazon GameLift Servers 관리형 EC2 호스팅과 동일하게 작동합니다.
- 호스팅 리소스를 위한 Anywhere 플릿을 설정합니다. Anywhere 플릿을 사용하면 Amazon GameLift Servers 서비스를 사용하여 Amazon GameLift Servers에서 관리하지 않는 리소스를 호스팅할 수 있습니다. 이 단계에서는 Amazon GameLift Servers 에이전트에 게임 서버 프로세스를 시작하는 시기와 방법을 알려주는 런타임 구성도 설정합니다.
- Amazon EC2 인스턴스를 설정합니다. 반복 테스트를 위한 테스트 환경입니다. 완전 관리형 Amazon GameLift Servers 인스턴스(프로덕션 수준 사용에 최적화됨) 대신 표준 EC2 인스턴스를 사용하는 것이 훨씬 빠릅니다. 표준 EC2 인스턴스를 사용하면 필요에 따라 게임 서버를 빠르고 지속적으로 업데이트할 수 있습니다.
- 게임 서버 빌드와 Amazon GameLift Servers 에이전트를 Amazon EC2 인스턴스에 배포합니다. 워크플로는 최신 버전의 게임 빌드와 모든 종속성을 가져와 EC2 인스턴스에 설치하는 스크립트를 제공합니다. 이 워크플로에서 Amazon GameLift Servers 에이전트와 CloudWatch 에이전트가 종속 항목에 포함됩니다.
- Amazon GameLift Servers 에이전트를 시작합니다. 설치가 완료되면 에이전트가 자동으로 명령을 시작하고 실행을 시작합니다. 다음이 포함됩니다.

- Amazon GameLift Servers Anywhere 플릿에서 EC2 인스턴스를 컴퓨팅으로 등록합니다.
- Amazon GameLift Servers 서비스와 WebSocket 연결을 설정하고 최신 런타임 구성을 가져옵니다.
- 런타임 구성의 지침에 따라 게임 서버 프로세스를 시작합니다. 이 워크플로에서 에이전트가 게임 서버 실행 파일의 단일 프로세스를 시작하라는 지시를 받습니다.
- 게임 시나리오를 테스트합니다. 테스트 환경을 설정하고 최신 게임 서버 빌드를 설치한 상태에서 테스트를 시작할 수 있습니다. 워크플로가 게임 세션 시작을 포함하여 테스트를 위한 몇 가지 단계를 안내합니다. CloudWatch 게임 서버 로그에 액세스하여 게임 세션이 시작되고 플레이어 수락을 준비할 때 진행 상황을 추적합니다.

게임 클라이언트 및 클라이언트 측 백엔드 서비스를 포함하여 게임 구성 요소를 개발할 때 해당 내용을 테스트 시나리오에 포함할 수 있습니다. 게임 클라이언트를 사용하여 게임 세션을 요청하고 Amazon GameLift Servers 서비스에서 연결 정보를 검색한 다음 게임 세션에 직접 연결합니다.

- 새 게임 서버 빌드를 배포하고 테스트를 반복합니다. 게임을 개발할 때 새 게임 서버 빌드를 생성한 다음 테스트를 위해 EC2 테스트 환경에 빠르게 배포할 수 있습니다. Amazon S3 버킷에 업로드한 다음 워크플로 스크립트를 사용하여 테스트 환경을 업데이트합니다.

게임을 Amazon GameLift Servers 관리형 플릿으로 전환

개발 테스트를 완료하고 시작할 준비가 되면 Amazon GameLift Servers 관리형 플릿으로 전환할 수 있는 좋은 시점입니다. 관리형 플릿을 사용하여 게임 호스팅 리소스를 미세 조정하고 테스트합니다. 게임 세션 배치 솔루션(대기열 및 매치메이커)을 구현하고, 최적의 호스팅 하드웨어(스팟 플릿 포함) 및 위치를 선택하고, 용량 조정 전략을 선택합니다. 를 사용하여 플릿, 대기열, 매치메이커를 포함한 모든 게임 호스팅 리소스의 수명 주기를 AWS CloudFormation 보다 효율적으로 관리할 수도 있습니다.

클라우드 기반 Anywhere 테스트 플릿에서 Amazon GameLift Servers 관리형 플릿으로 전환하는 데에 최소한의 작업만이 요구됩니다. 게임 코드를 변경할 필요가 없으며 동일한 대기열과 매치메이커를 재 사용할 수 있습니다. 다음 작업을 수행합니다.

- Amazon GameLift Servers 빌드 리소스를 생성합니다. Anywhere 테스트 플릿을 사용하면 게임 서버 빌드와 종속 항목을 각 플릿 컴퓨팅에 수동으로 배포해야 합니다. 관리형 플릿을 사용하여 게임 빌드 패키지를 Amazon GameLift Servers에 업로드하고 모든 플릿 컴퓨팅에 자동으로 배포합니다. 게임 빌드 파일을 패키징하고 Amazon S3 버킷의 파일을 사용하여 빌드 리소스를 생성하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers의 게임 서버 빌드 생성](#) 페이지를 참조하세요.
- 관리형 플릿을 생성합니다. 콘솔 또는 AWS CLI를 사용하여 플릿을 생성하고 EC2 관리형 플릿을 지정합니다. 이 유형의 플릿에는 빌드 리소스 및 인스턴스 유형 지정을 포함하여 추가 구성 설정이 필

요합니다. 동일한 런타임 구성을 사용하여 각 플릿 컴퓨팅에서 게임 서버 수명 주기를 관리할 수 있습니다. 관리형 플릿 생성에 대한 자세한 내용은 [Amazon GameLift Servers 관리형 EC2 플릿 생성](#) 페이지를 참조하세요.

- 플릿 별칭을 리디렉션합니다(선택 사항). Anywhere 플릿에 사용할 별칭을 설정한 경우 관리형 플릿에 동일한 별칭을 재사용할 수 있습니다. 별칭 생성 또는 업데이트에 대한 자세한 내용은 [Amazon GameLift Servers 별칭 생성](#) 페이지를 참조하세요.

Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정

Note

이 주제에서는 Amazon GameLift Servers용 서버 SDK 버전 5.x와 통합된 게임에 대한 로컬 테스트를 다룹니다. 게임에서 서버 SDK 버전 4.x 이하를 사용하는 경우 [Amazon GameLift Servers Local을 사용하여 통합 테스트](#) 페이지를 참조하세요.

Amazon GameLift Servers Anywhere 플릿과 자체 하드웨어를 사용하여 시뮬레이션된 호스팅 환경에서 게임 구성 요소를 반복적으로 빌드하고 테스트합니다. Anywhere 플릿을 설정하고 로컬 디바이스를 등록하여 Amazon GameLift Servers 서비스에 대한 연결을 설정합니다. 디바이스에 게임 서버 빌드를 설치하고, 게임 서버 프로세스를 시작하고, 필요에 따라 게임 기능을 테스트합니다. 게임 서버 빌드를 필요한 만큼 자주 업데이트하여 각 새 빌드 반복을 테스트할 수 있습니다.

Anywhere 플릿을 사용하면 AWS CLI 또는 테스트 스크립트를 사용하여 테스트할 수 있습니다. Amazon GameLift Servers와 게임 클라이언트를 통합한 경우 동일한 로컬 디바이스 또는 다른 디바이스에서 클라이언트를 실행할 수 있습니다.

Anywhere 플릿을 사용한 로컬 테스트는 Amazon GameLift Servers와의 게임 서버 통합을 테스트하는데 특히 유용합니다. 로컬 시스템의 모든 호스팅 활동과 이벤트 및 로깅 데이터를 완벽하게 파악할 수 있습니다.

Note

Unreal Engine 또는 Unity에 Amazon GameLift Servers 플러그인을 사용하고 있다면 이러한 도구에 Anywhere 플릿을 사용하여 로컬 테스트를 설정하기 위한 안내 워크플로가 포함됩니다. [Unity용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬 테스트 설정](#) 또는 [Unreal용 플러그인: Amazon GameLift Servers Anywhere를 사용하여 로컬에서 게임 호스팅에 대한 설명서를 따릅니다.](#)

주제

- [로컬 Anywhere 플릿 설정](#)
- [게임 서버 업데이트 및 설치](#)
- [게임 세션 활동 테스트](#)
- [게임 서버의 반복](#)
- [게임을 Amazon GameLift Servers 관리형 플릿으로 전환](#)

로컬 Anywhere 플릿 설정

다음 단계에 따라 로컬 워크스테이션에 대한 Anywhere 플릿을 생성합니다. AWS CLI 또는 용를 사용하는 자세한 지침은 섹션을 Amazon GameLift Servers참조 AWS Management Console 하세요 [Amazon GameLift Servers Anywhere 플릿 생성](#).

Anywhere 플릿 생성

1. 로컬 워크스테이션에 대한 사용자 지정 위치를 생성합니다. (AWS CLI 또는 콘솔). 사용자 지정 위치는 Anywhere 플릿에 포함할 컴퓨팅 리소스의 레이블입니다. 사용자 지정 위치 이름은 custom-으로 시작해야 합니다. 예를 들어 custom-my_laptop입니다. [사용자 지정 위치 생성](#)을(를) 참조하세요.
2. Anywhere 플릿(AWS CLI 또는 콘솔)을 생성합니다. 이 단계에서는 로컬 워크스테이션의 사용자 지정 위치로 플릿 리소스를 생성합니다. [Anywhere 플릿 생성](#)을(를) 참조하세요.

새 플릿의 ID 또는 ARN 값을 기록해 둡니다. 다음 단계에서 이 값을 사용합니다.

3. 로컬 워크스테이션을 플릿 컴퓨팅으로 등록합니다(AWS CLI만 해당). 게임 서버를 호스팅하려면 Anywhere 플릿에 컴퓨팅 리소스가 하나 이상 있어야 합니다. [플릿에 컴퓨팅 추가](#)을(를) 참조하세요. 플릿에 컴퓨팅을 추가하려면 다음 정보가 필요합니다.
 - 컴퓨팅 이름입니다. 플릿의 컴퓨팅마다 고유한 이름이 있어야 합니다.
 - Anywhere 플릿 식별자입니다. FleetID 또는 FleetArn를 사용할 수 있습니다.
 - 컴퓨팅의 연결 정보입니다. IpAddress 또는 DnsName를 지정합니다. Amazon GameLift Servers와 게임 클라이언트가 게임 서버에 연결하는 방법입니다.
 - Anywhere 플릿의 사용자 지정 위치입니다.

GameLiftServiceSdkEndpoint 반환 값을 기록해 둡니다. Anywhere 플릿에서 실행되도록 게임 서버를 업데이트할 때 이 값이 필요합니다.

게임 서버 업데이트 및 설치

이 작업은 게임 서버 빌드를 Amazon GameLift Servers 서버 SDK 5.x와 이미 통합했다고 가정합니다. 통합 프로세스에는 Amazon GameLift Servers 서비스와 상호 작용하여 게임 세션을 시작하고 관리할 수 있도록 게임 서버에 코드를 추가하는 작업이 포함됩니다.

Anywhere 플릿의 경우 특정 게임 서버 설정을 수동으로 구성해야 합니다. Amazon GameLift Servers 관리형 플릿에서는 이러한 설정이 자동으로 구성됩니다.

Anywhere 플릿을 위한 게임 서버 준비

1. 인증 토큰을 가져옵니다. 게임 서버에 Amazon GameLift Servers 서비스와 통신할 때마다 인증 토큰이 포함되어야 합니다. Amazon GameLift Servers 인증 토큰은 수명이 짧으며 정기적으로 새로 고쳐야 합니다.

가장 좋은 방법은 스크립트를 생성하여 다음 작업을 완료하는 것입니다.

- AWS CLI 작업을 호출합니다 `get-compute-auth-token`.
- 반환된 토큰 값을 로컬 컴퓨팅의 환경 변수에서와 같이 게임 서버 프로세스가 검색할 수 있는 위치에 저장합니다.

컴퓨팅에 게임 서버와 함께 스크립트를 설치합니다. 첫 번째 게임 서버 프로세스를 시작하기 전에 실행되도록 스크립트를 설정합니다. 게임 서버 프로세스가 활성 상태인 동안 스크립트를 정기적으로 실행하여 유효한 인증 토큰을 유지합니다. 컴퓨팅의 모든 게임 서버 프로세스는 동일한 인증 토큰을 사용할 수 있습니다.

2. Amazon GameLift Servers 게임 서버 코드를 업데이트합니다. 게임 서버 코드를 Amazon GameLift Servers용 서버 SDK와 통합할 때 `InitSdk()` 작업에 호출을 추가했습니다. 게임 서버가 Anywhere 플릿에서 실행되는 경우 이 호출에는 추가 서버 파라미터가 필요합니다. 자세한 내용은 [서버 프로세스 초기화](#) 및 [Amazon GameLift Servers용 서버 SDK 5.x](#)에서 개발 언어를 참조하세요. 서버 파라미터는 다음과 같습니다.

- `webSocketUrl` - 플릿에 컴퓨팅을 등록할 때 반환되는 `GameLiftServiceSdkEndpoint` 값으로 이 파라미터를 설정합니다.
- `hostId` - Anywhere 플릿에 컴퓨팅을 등록할 때 지정하는 컴퓨팅 이름으로 이 파라미터를 설정합니다.
- `fleetId` - Anywhere 플릿의 ID로 이 파라미터를 설정합니다.
- `authToken` - 컴퓨팅에 대한 인증 토큰을 검색하라는 요청에 응답하여 반환되는 토큰으로 이 파라미터를 설정합니다.

- `processId` - 로컬 컴퓨팅에서 실행 중인 게임 서버 프로세스를 식별하려면 이 파라미터를 설정합니다. 각 동시 게임 서버 프로세스에는 고유한 프로세스 ID가 있어야 합니다.

각 게임 서버 프로세스에서 사용하는 서버 파라미터 값은 프로세스가 실행 중인 Anywhere 플릿 컴퓨팅 전용이어야 합니다. 컴퓨팅에 적합한 값을 가져오는 방법에 대한 자세한 내용은 [플릿에 컴퓨팅 추가](#) 페이지를 참조하세요. 가장 좋은 방법은 `webSocketUrl`, `hostId`, `fleetId`, `authToken`을 로컬 컴퓨팅의 환경 변수로 설정하는 것입니다. 컴퓨팅에서 실행되는 모든 서버 프로세스는 이러한 값을 사용합니다.

3. 로컬 컴퓨팅에 게임 서버 빌드를 설치합니다. 게임 서버를 실행하는 데 필요한 모든 종속 항목을 포함합니다.
4. 로컬 컴퓨팅에서 실행되는 하나 이상의 게임 서버 프로세스를 시작합니다. 게임 서버 프로세스가 서버 SDK 작업 `ProcessReady()`를 호출하면 프로세스가 게임 세션을 호스팅할 준비가 된 것입니다.

게임 세션 활동 테스트

게임 세션을 사용하여 게임 서버 통합을 테스트합니다. Amazon GameLift Servers 기능과 통합된 게임 클라이언트가 없는 경우 AWS CLI를 사용하여 게임 세션을 시작할 수 있습니다. 다음 시나리오를 사용해볼 수 있습니다.

- 게임 세션을 생성합니다. [create-game-session](#) 명령(또는 [CreateGameSession](#) API 작업)을 호출합니다. Anywhere 플릿의 ID와 사용자 지정 위치를 지정합니다. 이 호출은 새 게임 세션의 고유 식별자를 반환합니다.
- 게임 세션 상태를 확인합니다. [describe-game-sessions](#) 명령(또는 [DescribeGameSessions](#) API 작업)을 호출합니다. 게임 세션 ID를 지정합니다. 이 호출은 게임 세션 상태를 포함한 자세한 게임 세션 정보를 반환합니다. 활성 상태의 게임 세션이 플레이어에 연결할 준비가 됩니다. 플릿의 모든 게임 세션 목록을 가져오려면 [list-game-sessions](#) 명령(또는 [ListGameSessions](#) API 작업)을 호출합니다.
- 게임 세션에 연결합니다. 게임 클라이언트가 게임 세션에 연결할 수 있는 경우 게임 세션 정보에 포함된 연결 정보를 사용합니다.

게임 서버의 반복

동일한 Anywhere 플릿과 컴퓨팅을 사용하여 게임 서버 빌드의 다른 버전을 테스트할 수 있습니다.

1. 기존 **GameSession**을 정리합니다. 게임 서버 프로세스가 충돌하거나 `ProcessEnding()`을 호출하지 않는 경우, Amazon GameLift Servers는 게임 서버의 상태 확인 전송을 중단한 후 **GameSession**을 정리합니다.
2. 새 게임 서버 빌드를 생성합니다. 게임 서버를 변경하고 수정된 빌드를 패키징합니다.
3. 로컬 컴퓨팅에서 게임 서버 빌드를 업데이트합니다. 이전 Anywhere 플릿은 활성 상태이고 랩톱은 플릿의 컴퓨팅 리소스로 등록되어 있습니다.
4. 업데이트된 권한 부여 토큰을 가져옵니다. [get-compute-auth-token](#) CLI 명령을 호출하고 토큰을 로컬 컴퓨팅에 저장합니다.
5. 로컬 컴퓨팅에서 실행되는 하나 이상의 게임 서버 프로세스를 시작합니다. 게임 서버 프로세스가 `ProcessReady()`를 호출하면 테스트에 사용할 준비가 된 것입니다.

게임을 Amazon GameLift Servers 관리형 플릿으로 전환

개발 테스트를 완료하고 시작할 준비가 되면 Amazon GameLift Servers 관리형 플릿으로 전환할 수 있는 좋은 시점입니다. 관리형 플릿을 사용하여 게임 호스팅 리소스를 미세 조정하고 테스트합니다. 게임 세션 배치 솔루션(대기열 및 매치메이커)을 구현하고, 최적의 호스팅 하드웨어(스팟 플릿 포함) 및 위치를 선택하고, 용량 조정 전략을 선택합니다. 를 사용하여 플릿, 대기열, 매치메이커를 포함한 모든 게임 호스팅 리소스의 수명 주기를 AWS CloudFormation 보다 효율적으로 관리할 수도 있습니다.

로컬 Anywhere 테스트 플릿에서 Amazon GameLift Servers 관리형 플릿으로 전환하려면 간단하게 몇 가지를 수정해야 합니다. 동일한 대기열과 매치메이커를 재사용할 수 있습니다. 다음 작업을 수행합니다.

- 게임 서버 코드 호출을 **InitSdk()**로 변경합니다. 서버 파라미터를 제거합니다. 관리형 플릿의 경우 Amazon GameLift Servers는 이 정보를 자동으로 추적합니다.
- Amazon GameLift Servers 빌드 리소스를 생성합니다. Anywhere 테스트 플릿을 사용하면 게임 서버 빌드와 종속 항목을 각 플릿 컴퓨팅에 수동으로 배포해야 합니다. 관리형 플릿을 사용하여 게임 빌드 패키지를 생성한 후 Amazon GameLift Servers에 업로드하고 모든 플릿 컴퓨팅에 자동으로 배포합니다. 게임 빌드 파일을 패키징하고 Amazon S3 버킷의 파일을 사용하여 빌드 리소스를 생성하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers의 게임 서버 빌드 생성](#) 페이지를 참조하세요. Amazon GameLift Servers는 관리형 플릿으로 이러한 작업을 자동으로 처리하기 때문에 컴퓨팅을 등록하고 인증 토큰을 가져오는 스크립트는 포함하지 마세요.
- 관리형 플릿을 생성합니다. EC2 관리형 플릿을 지정하여 콘솔 또는 AWS CLI를 사용하여 플릿을 생성합니다. 이 유형의 플릿에는 빌드 리소스 및 인스턴스 유형 지정을 포함하여 추가 구성 설정이 필요합니다. 동일한 런타임 구성을 사용하여 각 플릿 컴퓨팅에서 게임 서버 수명 주기를 관리할 수 있

도록 런타임 구성도 필요합니다. 관리형 플릿 생성에 대한 자세한 내용은 [Amazon GameLift Servers 관리형 EC2 플릿 생성](#) 페이지를 참조하세요.

- 플릿 별칭을 리디렉션합니다(선택 사항). Anywhere 플릿에 사용할 별칭을 설정한 경우 관리형 플릿에 동일한 별칭을 재사용할 수 있습니다. 별칭 생성 또는 업데이트에 대한 자세한 내용은 [Amazon GameLift Servers 별칭 생성](#) 페이지를 참조하세요.

Amazon GameLift Servers Local을 사용하여 통합 테스트

Note

이 주제에서는 Amazon GameLift Servers용 서버 SDK 버전 3.x 및 4.x으로만 통합된 게임에 대한 로컬 테스트를 다룹니다. 서버 SDK 패키지에는 호환 가능한 버전의 Amazon GameLift Servers Local이 포함되어 있습니다. 서버 SDK 버전 5.x를 사용하는 경우 Amazon GameLift Servers Anywhere 플릿을 사용한 로컬 테스트는 [GameLift 서버 SDK 5.x를 사용한 로컬 테스트](#)를 참조하세요. SDK 4.x에서 5.x로 마이그레이션하는 방법에 대한 지침은 [GameLift 서버 SDK 5.x로 마이그레이션](#)을 참조하세요.

Amazon GameLift Servers Local을 사용하여 로컬 디바이스에서 제한된 버전의 관리형 Amazon GameLift Servers 서비스를 실행하고 이를 기준으로 게임 통합을 테스트합니다. 이 도구는 게임 통합에서 반복적 개발을 수행할 때 유용합니다. 대신, 각각의 새 빌드를 Amazon GameLift Servers에 업로드하고 게임을 호스팅하도록 플릿을 구성하는 각 작업에는 다소 긴 시간이 소요될 수 있습니다.

Amazon GameLift Servers Local을 사용하여 다음을 확인할 수 있습니다.

- 게임 서버가 Server SDK와 올바르게 통합되며 Amazon GameLift Servers 서비스와 적절히 통신하여 새 게임 세션을 시작하고 새 플레이어를 수락하며 상태를 보고합니다.
- 게임 클라이언트가 Amazon GameLift Servers용 AWS SDK와 올바르게 통합되며 기존 게임 세션에 대한 정보를 검색하고 새 게임 세션을 시작하며 플레이어를 게임에 참가시키고 게임 세션에 연결할 수 있습니다.

Amazon GameLift Servers Local은 독립형 버전의 관리형 Amazon GameLift Servers 서비스를 시작하는 명령줄 도구입니다. Amazon GameLift Servers. Local은 서버 프로세스 초기화, 상태 확인, API 호출 및 응답의 실행 중인 이벤트 로그도 제공합니다. Amazon GameLift Servers. Local은 Amazon GameLift Servers에 대한 AWS SDK 작업의 하위 세트를 인식합니다. AWS CLI 또는 게임 클라이언트

에서 호출할 수 있습니다. 모든 API 작업은 Amazon GameLift Servers 웹 서비스에서와 마찬가지로 로컬에서 수행됩니다.

각 서버 프로세스는 단일 게임 세션만 호스팅해야 합니다. 게임 세션은 Amazon GameLift Servers Local에 연결하는 데 사용하는 실행 파일입니다. 게임 세션이 완료되면 `GameLiftServerSDK::ProcessEnding`을 호출한 다음 프로세스를 종료해야 합니다. Amazon GameLift Servers Local을 사용하여 로컬에서 테스트하는 경우 여러 서버 프로세스를 시작할 수 있습니다. 각 프로세스는 Amazon GameLift Servers 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 Servers Local은 다음 API를 지원합니다.

- `CreateGameSession`
- `CreatePlayerSession`
- `CreatePlayerSessions`
- `DescribeGameSessions`
- `DescribePlayerSessions`

Amazon GameLift Servers Local 설정

Amazon GameLift Servers Local은 [서버 SDK](#) 번들과 함께 .jar 실행 파일로 제공됩니다. 이는 Windows 또는 Linux에서 실행할 수 있으며 Amazon GameLift Servers 지원 언어와 함께 사용할 수 있습니다. SDK 버전 4.0.2의 경우 [공식 GitHub 릴리스](#)에서 다운로드할 수 있습니다. `GameLiftLocal.jar` 파일은 `GameLift-CSharp-ServerSDK-4.0.2.zip` 아티팩트에 포함되어 있습니다.

Local을 실행하기 전에 다음이 설치되어 있어야 합니다.

- Amazon GameLift Servers용 서버 SDK 버전 3.1.5~4.x의 빌드
- Java 8

게임 서버 테스트

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

- 게임 서버가 제대로 시작하고 Amazon GameLift Servers용 서버 SDK를 초기화합니다.

- 시작 프로세스의 일환으로 게임 서버가 게임 세션을 호스팅할 준비가 되었음을 Amazon GameLift Servers에 알립니다.
- 실행 도중 게임 서버가 Amazon GameLift Servers에게 1분마다 상태를 전송합니다.
- 게임 서버가 새 게임 세션 시작 요청에 응답합니다.

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

Local 명령 프롬프트 창에서 게임 서버가 Amazon GameLift Servers Local 서비스에 연결되었다는 로그 메시지가 표시됩니다. 이는 게임 서버가 Amazon GameLift Servers용 서버 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 GameLift Servers는 게임 서버의 각 상태 보고서를 로깅합니다. 다음 로그 메시지는 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. 을 시작합니다..AWS CLI

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 Servers 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 레퍼런스](#)에 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 Servers 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: gameId:
    "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"
```

AWS CLI 창에서 Amazon GameLift Servers는 게임 세션 ID를 포함한 게임 세션 객체를 사용하여 응답합니다. 새 게임 세션 상태가 Activating임에 유의하세요. 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 Servers 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 AWS Local 서비스에 성공적으로 Amazon GameLift Servers SDK 요청을 합니다.
- 플레이어가 게임 세션에 참가하려 할 때 게임 서버가 올바른 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 Servers 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 Servers 통합 게임 서버를 로컬에서 시작합니다. 메시지 로그에 대한 자세한 내용은 [게임 서버 테스트](#) 섹션을 참조하세요.

3. Local에 대해 게임 클라이언트를 구성하고 시작합니다.

Amazon GameLift Servers Local 서비스와 함께 게임 클라이언트를 사용하려면 [Amazon GameLift Servers API를 설정합니다](#).에 설명된 것처럼 게임 클라이언트의 설정을 다음과 같이 변경해야 합니다.

- ClientConfiguration 등 Local 엔드포인트를 가리키도록 `http://localhost:9080` 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 설정하세요.
- AWS 자격 증명을 설정합니다. Local의 경우, 실제 AWS 자격 증명에 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 Servers 서비스와 성공적으로 통신하고 있다는 로그 메시지가 표시되어야 합니다.

4. 게임 클라이언트의 Amazon GameLift Servers 서비스 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 Servers 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 `Reserved`에서 `Active`로 변경되어야 합니다.

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

Amazon GameLift Servers가 플레이어 수요를 관리하고 지표를 올바르게 보고하려면 게임 서버가 다양한 상태를 Amazon GameLift Servers에 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 Servers Local 로그 메시지가 표시되어야 합니다. AWS CLI가 **DescribePlayerSessions()**를 호출하면 Active에서 Completed로 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 Servers Local 로그 메시지가 표시되어야 합니다.

Note

이전 지침은 게임 세션을 종료할 때 **TerminateGameSession()**을 호출하는 것이었습니다. 이 메서드는 Amazon GameLift Servers 서버 SDK v4.0.1에서 더 이상 사용되지 않습니다. [게임 세션 종료](#)을(를) 참조하세요.

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

Local을 사용한 변형

Amazon GameLift Servers Local을 사용할 때는 다음 사항을 명심하십시오.

- Local은 Amazon GameLift Servers 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 Servers 플릿 문제 디버깅

이 주제에서는 Amazon GameLift Servers 관리형 EC2 플릿의 문제를 해결하는 방법에 대한 지침을 제공합니다.

플릿 생성 문제

관리형 EC2 플릿을 생성하면 Amazon GameLift Servers 서비스가 플릿 생성, 게임 서버 빌드가 설치된 EC2 인스턴스 배포, 각 인스턴스에서 게임 서버 프로세스 시작을 실행하는 워크플로를 시작합니다. 자세한 설명은 섹션을 참조하세요. 플릿은 활성 상태에 도달할 때까지 게임 세션과 플레이어를 호스팅할 수 없습니다.

문제가 발생한 플릿 생성 단계를 식별하고 플릿 생성 이벤트 및 로그를 검토하여 플릿이 활성화되지 않도록 하는 문제를 디버깅할 수 있습니다. 만약 로그에서 유용한 정보를 찾을 수 없다면, 문제는 내부 서비스 오류일 가능성이 있습니다. 이 경우, 플릿을 다시 생성해 보세요. 문제가 지속되면 게임 빌드를 다시 업로드하여 파일 손상 가능성을 해결하세요. Amazon GameLift Servers 지원에 문의하거나 포럼에 질문을 게시해도 됩니다.

빌드 다운로드 및 검증

이 단계에서 Amazon GameLift Servers는 업로드된 게임 서버 빌드를 가져오고, 파일을 추출하고, 모든 설치 스크립트를 실행합니다. 이러한 단계에서 플릿 생성이 실패하면 플릿 이벤트와 로그를 확인하여 정확한 문제를 파악합니다. 가능한 원인은 다음과 같습니다.

- Amazon GameLift Servers가 압축된 빌드 파일(이벤트 `FLEET_BINARY_DOWNLOAD_FAILED`)을 가져올 수 없습니다. 빌드의 스토리지 위치에 액세스할 수 있고, 빌드와 동일한에서 플릿 AWS 리전을 생성하고 있으며, 해당 플릿에 액세스할 수 있는 올바른 권한이 Amazon GameLift Servers 있는지 확인합니다.
- Amazon GameLift Servers가 빌드 파일(이벤트 `FLEET_CREATION_EXTRACTING_BUILD`)을 추출할 수 없습니다.
- 빌드 파일의 설치 스크립트가 성공적으로 완료되지 않았습니다(이벤트 `FLEET_CREATION_FAILED_INSTALLER`).

플릿 리소스 구축

이 단계의 문제는 일반적으로 플릿 리소스의 할당 및 배포와 관련이 있습니다. 가능한 원인은 다음과 같습니다.

- 요청한 인스턴스 유형을 사용할 수 없습니다.
- 요청한 플릿 유형(스팟 또는 온디맨드)을 사용할 수 없습니다.

게임 서버 프로세스 활성화

이 단계에서는 Amazon GameLift Servers가 게임 서버 실행 가능성, 런타임 구성 설정, 서버 SDK를 사용하여 Amazon GameLift Servers 서비스와 상호 작용하는 게임 서버 기능을 비롯해 여러 요소에 대한 테스트와 같은 주요 요소를 테스트하고 다양한 작업을 시도합니다.

Note

이 단계에서는 플릿 인스턴스에 원격으로 액세스하여 문제를 추가로 조사할 수 있습니다. [플릿 인스턴스에 연결을\(를\) 참조](#)하세요.

가능한 문제 양상은 다음과 같습니다.

- 서버 프로세스가 실행되지 않습니다. 이는 플릿의 런타임 구성 설정 (이벤트 `FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND` 또는 `FLEET_VALIDATION_EXECUTABLE_RUNTIME_FAILURE`)과 관련한 문제일 수 있습니다. 시작 경로와 선택적 시작 파라미터를 올바르게 설정했는지 확인합니다.
- 서버 프로세스가 시작되지만 플릿을 활성화할 수 없습니다. 서버 프로세스가 시작되고 계속 성공적으로 실행되지만 플릿이 활성 상태로 변경되지 않는 경우는 서버 프로세스가 Amazon GameLift Servers와 통신하는 데 실패했기 때문일 가능성이 높습니다. 게임 서버가 이러한 올바른 서버 SDK 호출을 수행하고 있는지 확인합니다([서버 프로세스 초기화](#) 참조).
- 서버 프로세스가 초기화되지 않습니다(이벤트 `SERVER_PROCESS_SDK_INITIALIZATION_TIMEOUT`). 서버 프로세스가 `InitSdk()`를 성공적으로 호출하지 못했습니다.
- 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 Amazon GameLift Servers에 알리지 못합니다(이벤트 `SERVER_PROCESS_PROCESS_READY_TIMEOUT`). 서버 프로세스가 초기화 되었지만 `ProcessReady()` 호출이 제시간에 이뤄지지 않았습니다.
- VPC 피어링 연결 요청에 실패했습니다. VPC 피어링 연결로 생성되는 플릿은([새 플릿으로 VPC 피어링을 설정하려면](#) 참조) 이 활성화 단계에서 VPC 피어링이 실행됩니다. 어떤 이유로 든 VPC 피어링에 실패할 경우 새 플릿이 활성 상태로 바뀌지 못합니다. [describe-vpc-peering-](#)

[connections](#)를 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 줄여보세요. Amazon GameLift Servers 콘솔을 사용하거나(플릿의 용량 할당 설정 편집) AWS CLI 명령 [update-runtime-configuration](#)을 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 Auto Scaling 기능을 해제해야 합니다.

VPC 작업이 허용되지 않습니다.

이 문제는 특별히 VPC 피어링 연결을 생성한 플릿에만 적용됩니다(참조 [Amazon GameLift Servers의 VPC 피어링 설정](#)). 이 시나리오는 플릿 삭제 프로세스에 플릿의 VPC 및 모든 VPC 피어링 연결 삭제도 포함되기 때문에 발생합니다. 먼저 Amazon GameLift Servers [CreateVpcPeeringAuthorization\(\)](#)에 대한 서비스 API를 호출하거나 AWS CLI 명령을 사용하여 인증을 받아야 합니다 `create-vpc-peering-authorization`. 권한을 받은 후에 플릿을 삭제할 수 있습니다.

플릿 리소스에 원격으로 액세스

활성 Amazon GameLift Servers 관리형 EC2 또는 관리형 컨테이너 플릿의 모든 인스턴스에 연결할 수 있습니다. 컨테이너 플릿의 경우 인스턴스에서 실행되는 개별 컨테이너에 연결하여 게임 세션 문제를

해결하고, 로그를 검사하고, 런타임 문제를 디버깅할 수도 있습니다. 인스턴스에 원격으로 액세스하는 일반적인 이유는 다음과 같습니다.

- 게임 서버 통합 문제를 해결합니다.
- 런타임 구성 및 기타 플릿별 설정을 미세 조정합니다.
- 로그 추적과 같은 실시간 게임 서버 활동을 모니터링합니다.
- 실제 플레이어 트래픽을 사용하여 벤치마킹 도구를 실행합니다.
- 게임 세션 또는 게임 서버 프로세스와 관련된 특정 문제를 조사합니다.

인스턴스에 연결할 때는 다음 사항에 유의하세요.

- 활성 플릿의 모든 인스턴스에 연결할 수 있습니다. 일반적으로 활성화를 진행 중이거나 오류 상태인 플릿과 같은 비활성 플릿에는 연결할 수 없습니다 (해당 플릿은 일시적 가용성 제한 가능성 있음). 플릿 활성화 문제에 대한 도움말은 [Amazon GameLift Servers 플릿 문제 디버깅](#) 섹션을 참조하세요.
- 활성 인스턴스에 연결해도 인스턴스의 호스팅 활동에는 영향을 주지 않습니다. 인스턴스는 런타임 구성을 기반으로 서버 프로세스를 계속 시작하고 중지합니다. 이를 통해 게임 세션을 활성화하고 실행합니다. 스케일 다운 이벤트 또는 기타 이벤트에 대한 응답으로 인스턴스가 종료될 수 있습니다.
- 인스턴스의 파일 또는 설정을 변경하면 인스턴스의 활성 게임 세션과 연결된 플레이어에 영향을 미칠 수 있습니다.

플릿 인스턴스에 연결

활성 Amazon GameLift Servers 관리형 EC2 또는 관리형 컨테이너 플릿 인스턴스에 원격으로 연결하여 게임 서버 문제를 해결하고, 로그를 검사하고, 런타임 동작을 디버깅할 수 있습니다. Amazon GameLift Servers 콘솔 또는 AWS CLI를 사용하여 연결합니다.

콘솔을 통한 원격 액세스

Amazon EC2 Systems Manager(SSM)를 사용하여 Amazon GameLift Servers 콘솔에서 직접 플릿 인스턴스에 연결할 수 있습니다. 이 방법을 통해 추가 설정 또는 자격 증명 관리 없이 보안 액세스를 제공합니다. 컨테이너 플릿의 경우 인스턴스에 연결한 후 인스턴스에서 실행되는 개별 컨테이너에 액세스할 수 있습니다. 자세한 내용은 [컨테이너에 연결](#) 단원을 참조하십시오.

1. Amazon GameLift Servers 콘솔의 탐색 창에서 관리형 EC2 또는 관리형 컨테이너를 선택한 다음 플릿을 선택합니다.
2. 액세스하려는 인스턴스가 포함된 플릿 ID를 선택합니다.

3. 플릿 세부 정보 페이지에서 인스턴스 탭을 선택하여 플릿의 모든 컴퓨팅 인스턴스를 봅니다.
4. 연결할 인스턴스를 선택한 다음 연결을 선택합니다. 그러면 연결 세부 정보를 알려주고 인스턴스에 연결하는 데 사용할 스크립트를 볼 수 있는 인스턴스에 연결 대화 상자가 표시됩니다. 연결을 다시 선택하여 확인합니다.
5. 연결 대화 상자에서 실행을 선택하여 새 SSM 세션을 생성합니다. 시스템은 AWS Key Management Service(AWS KMS)를 통해 세션을 인증하고 브라우저에서 터미널을 엽니다.

Note

콘솔 기반 원격 액세스는 서버 SDK 버전 5.x를 실행하는 플릿에 사용할 수 있습니다. 이전 SDK 버전을 실행하는 플릿의 경우 다음 섹션에 설명된 AWS CLI 메서드를 사용합니다.

AWS CLI를 사용한 원격 액세스

다음 지침은 AWS 명령줄 인터페이스(CLI)를 사용하여 인스턴스에 원격으로 연결하는 방법을 설명합니다. 예 [대한 서비스 API 참조Amazon GameLift Servers](#)에 설명된 대로 AWS SDK를 사용하여 프로그래밍 방식으로 호출할 수도 있습니다.

인스턴스 데이터 수집

Amazon GameLift Servers 관리형 EC2 플릿 인스턴스에 연결하려면 다음 정보가 필요합니다.

- 연결할 인스턴스의 ID. 인스턴스 ID 또는 ARN을 사용할 수 있습니다.
- 인스턴스에서 사용 중인 Amazon GameLift Servers용 서버 SDK 버전. 서버 SDK는 인스턴스에서 실행 중인 게임 빌드와 통합됩니다.

다음 지침은 AWS CLI를 사용하여 이러한 작업을 완료하는 방법을 설명합니다. 연결하려는 인스턴스의 플릿 ID를 알아야 합니다.

1. 컴퓨팅 이름을 가져옵니다. 플릿의 모든 활성 컴퓨팅 목록을 가져옵니다. 플릿 ID 또는 ARN으로 [list-compute](#)를 호출합니다. 단일 위치 플릿의 경우 플릿 식별자만 지정합니다. 다중 위치 플릿의 경우 플릿 식별자와 위치를 지정합니다. 관리형 EC2 플릿을 사용하면 `list-compute`가 플릿 인스턴스 목록을 반환하며 속성 `ComputeName`은 인스턴스 ID입니다. 액세스하려는 컴퓨팅을 찾습니다.

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.uwp57060n1k6dnlnw49b78hg1west-2.amazongamelift.com",
      "ComputeStatus": "Active",
      "Location": "sa-east-1",
      "CreationTime": "2023-07-09T22:51:45.931000-07:00",
      "OperatingSystem": "AMAZON_LINUX_2023",
      "Type": "c4.large"
    }
  ]
}
```

2. 서버 SDK 버전을 찾습니다. 이 정보를 보려면 플릿에 배포된 빌드를 조회해야 합니다. 서버 SDK 버전은 빌드 속성입니다.
 - a. 플릿 ID 또는 ARN으로 [describe-fleet-attributes](#)를 직접적으로 호출하여 플릿의 빌드 ID 및 ARN을 가져옵니다.
 - b. 빌드 ID 또는 ARN으로 [describe-build](#)를 직접적으로 호출하여 빌드의 서버 SDK 버전을 가져옵니다.

예제:

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_2023",
  "ServerSdkVersion": "5.1.1",
  . . .
}
```

인스턴스에 연결(서버 SDK 5)

연결하려는 인스턴스가 서버 SDK 버전 5.x로 게임 빌드를 실행하는 경우, Amazon EC2 Systems Manager(SSM)를 사용하여 인스턴스에 연결합니다. Windows나 Linux를 실행 중인 원격 인스턴스에 액세스할 수 있습니다. 컨테이너 플릿의 경우 인스턴스에 연결한 후 인스턴스에서 실행되는 개별 컨테이너에 액세스할 수 있습니다. 자세한 내용은 [컨테이너에 연결](#) 단원을 참조하십시오.

시작하기 전에

SSM 설정 단계를 완료하고 로컬 시스템에 SSM 플러그인을 설치합니다. 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 [SSM 설정](#) 및 CLI용 Session Manager 플러그인 설치를 참조하세요. [AWS](#) Amazon EC2 Systems Manager

1. 인스턴스에 대한 액세스 자격 증명을 요청합니다. 연결할 인스턴스의 플릿 ID와 컴퓨팅 이름을 사용하여 [get-compute-access](#)를 호출합니다. Amazon GameLift Servers는 인스턴스에 액세스하기 위한 임시 자격 증명 세트를 반환합니다. 예제:

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(선택 사항). 자격 증명을 환경 변수로 내보내고 이를 사용하여 기본 사용자에게 대한 AWS CLI를 구성할 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 [AWS CLI를 구성하기 위한 환경 변수](#)를 참조하세요.

```
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 리전 또는 위치를 포함합니다. SSM 및 SSM 플러그인을 설정하는 방법을 비롯한 자세한 내용은 Amazon EC2 Systems Manager 사용 설명서의 [세션 시작\(AWS CLI\)](#)을 참조하세요.

시작 세션 요청은 1단계에서 획득한 자격 증명 정보를 자동으로 사용합니다.

```
aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2 \
```

Note

액세스 거부 오류가 발생하면 AWS_PROFILE 환경 변수가 AWS 프로파일로 설정되어 AWS CLI가 원격 액세스에 잘못된 자격 증명을 사용하게 될 수 있습니다. 문제를 해결하려면 AWS_PROFILE 환경 변수를 일시적으로 설정 해제합니다. 또는 원격 액세스 자격 증명에 대한 사용자 지정 AWS 프로파일을 생성하고 --profile 명령줄 파라미터를 start-session 요청에 추가할 수 있습니다.

인스턴스에 연결(서버 SDK 4.x 이하)

연결하려는 인스턴스가 서버 SDK 버전 4 이하에서 게임 빌드를 실행하는 경우 다음 지침을 사용합니다. Windows나 Linux를 실행 중인 원격 인스턴스에 액세스할 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 [get-instance-access](#) 명령을 사용하여 액세스 자격 증명을 요청합니다. 요청이 수락되면 Amazon GameLift Servers가 인스턴스의 운영 체제, IP 주소, 자격 증명 세트(사용자 이름과 비밀번호)를 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 인스턴스의 경우 - Linux 인스턴스에 연결을 시도하면 SSH가 사용자 이름과 프라이빗 키를 요구합니다. Amazon GameLift Servers는 RSA 프라이빗 키를 발행하고 줄 바꿈을 나타내는 줄 바꿈 문자(\n)를 포함해 하나의 문자열로 반환합니다. 프라이빗 키를 사용할 수 있도록 하려면 (1)문자열을 .pem 파일로 변환한 후, (2)새 파일에 대한 권한을 설정해야 합니다. 예제 자격 증명이 반환되었습니다.

```
"Credentials": {
  "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEAY7WZhaDsra1W3mRlQtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdE0QW
\nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449i1x9X1F
\nG50TCFe0zf18dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW
\nnoPzev/D8V+x4+bHthfSjR9Y7DvQFjfbVwHXigBdtZcU2/wei8D/HYwIDAQABaoIBAGZ1kaEvnrrqu
\n/uler7vgIn5m71N5LKw4hJLAIW6tUT/fzvtCHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
ufGxbL1\nmb5qwmGUNepJaZD6QSSs3kICLwUUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA30zdXzMqexXVJ1TLZVEH0E7bh1Y9d801ozR
\noQs/FiZNAx2iijCWyv01pjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfq1
+1Ip1\nYkriL0DbLXlvRAH+yHPRit2hH0jtUNZh4Axv+cpG09qbUI3+43eEy24B7G/Uh
+GTfbjsXs0xQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ50BEKzet9XCkKw0NBVELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWV0Eihvm+xTtmaZ1Sp//1kq75XDwnU
\nwA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Un12ajLivWUt5pbBrKbUC
\nngYBjb0+0Zk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\nnoMo7yykABY70zd5wQewBQ4AdSlWSX4nGDtsiFxiWiI5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVwFFPrTkJNvJZzYt69qezx1sjgFKshy
\nWBhd4xHZtmCqpBP1AymEjr/T01bxyARMXmNIOWIANXMGb4KGSyl1mzSVAoQ+fqR+cJ3d0dyP11j
\nnjbb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvTW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjKpUfVTrW0YFjXkfcRr/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
RSA PRIVATE KEY-----",
  "UserName": "gl-user-remote"
}
```

AWS CLI를 사용할 때 `get-instance-access` 요청에 `--query` 및 `--output` 파라미터를 포함하여 .pem 파일을 자동으로 생성할 수 있습니다.

.pem 파일에 대한 권한을 설정하려면 다음 명령을 실행합니다.

```
$ chmod 400 MyPrivateKey.pem
```

2. 원격 연결을 위한 포트를 엽니다. Amazon GameLift Servers 플릿의 인스턴스는 플릿 구성에서 권한이 있는 포트를 통해서만 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 후에는 활성화되기 전에 포트 설정을 업데이트할 수 없습니다. 문제가 해결되지 않으면 포트 설정을 연 상태로 플릿을 다시 생성합니다.

[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

플릿 인스턴스에 연결되면 전체 사용자 및 관리자 액세스 권한이 부여됩니다. 즉 게임 호스팅으로 오류 또는 장애를 야기할 수도 있다는 뜻입니다. 인스턴스가 활성 플레이어로 게임을 호스팅하는 경우 게임

세션이 충돌하고 플레이어가 삭제되거나 게임 종료 프로세스가 중단되어 저장된 게임 데이터 및 로그에 오류가 발생할 위험이 있습니다.

컨테이너 플릿의 경우 컨테이너에 액세스한 후(참조 [컨테이너에 연결](#)) 컨테이너 파일 시스템 내에 다음 파일 위치가 적용됩니다. EC2 플릿의 경우 이러한 파일 위치가 인스턴스에 직접 적용됩니다.

호스팅 인스턴스에서 다음 리소스를 찾습니다.

- 게임 빌드 파일. Amazon GameLift Servers에 업로드한 게임 빌드 파일입니다. 여기에는 하나 이상의 게임 서버 실행 파일, 자산, 종속성 파일이 포함됩니다. 게임 빌드 파일은 game 루트 디렉터리에 있습니다.
 - Windows: c:\game
 - Linux: /local/game
- 게임 로그 파일. 지정한 디렉토리 경로에 위치한 game 루트 디렉토리 내에 있는 게임 서버가 생성한 로그 파일을 찾습니다.
- Amazon GameLift Servers 호스팅 리소스. Amazon GameLift Servers 서비스가 게임 호스팅을 관리하기 위해 사용한 파일은 Whitewater 루트 디렉터리에 있습니다. 어떤 이유로든 이 파일을 수정하지 마세요.
- 런타임 구성. 개별 인스턴스에 대한 런타임 구성에 액세스하지 마세요. 런타임 구성 속성을 변경하려면 플릿의 런타임 구성을 업데이트합니다(SDK 작업 [UpdateRuntimeConfiguration](#) 또는 the AWS CLI [update-runtime-configuration](#) 참조 AWS).
- 플릿 데이터. JSON 파일에는 인스턴스에서 실행되는 서버 프로세스에서 사용할 수 있도록 인스턴스가 속한 플릿에 대한 정보가 포함되어 있습니다. JSON 파일은 다음 위치에 있습니다.
 - Windows: C:\GameMetadata\gamelift-metadata.json
 - Linux: /local/gamemetadata/gamelift-metadata.json
- TLS 인증서. TLS 인증서 생성이 활성화되어 있는 플릿에 인스턴스가 있는 경우 인증서, 인증서 체인, 프라이빗 키, 루트 인증서를 포함한 인증서 파일은 다음 위치에 있습니다.
 - Windows: c:\\GameMetadata\Certificates
 - Linux: /local/gamemetadata/certificates/

컨테이너에 연결

Amazon GameLift Servers 컨테이너 플릿의 경우 플릿 인스턴스에서 실행되는 게임 서버 컨테이너에 액세스할 수 있습니다. 컨테이너 액세스를 사용하여 게임 세션 문제를 해결하고, 로그를 검사하고, 런타임 문제를 디버깅합니다.

컨테이너에 연결

시작하기 전에:

플릿 인스턴스에 연결합니다. 지침은 [플릿 인스턴스에 연결](#) 섹션을 참조하세요.

다음 명령을 실행하여 인스턴스에서 실행 중인 컨테이너를 나열합니다.

```
sudo docker ps
```

출력에는 게임 서버 컨테이너 및 내부 컨테이너를 포함하여 인스턴스에서 실행되는 모든 컨테이너가 나열Amazon GameLift Servers됩니다. 게임 서버 이미지가 있는 컨테이너를 찾아 게임 서버 컨테이너를 식별합니다.

출력 예제:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
b9676e9489f5	game-server-container	"/bin/sh -c ./ \$GAME_..."	2 days ago	Up 2 days
1d1c8443efe2	support-container	"/bin/sh -c ./ \$SUPP0..."	2 days ago	Up 2 days

게임 서버 컨테이너에 연결하려면 CONTAINER ID 열의 컨테이너 짧은 ID를 사용합니다. 이렇게 하면 컨테이너 파일 시스템에 대한 전체 읽기 및 쓰기 액세스 권한이 부여됩니다.

```
sudo docker exec -it container-short-id sh
```

콘솔을 통해 컨테이너에 연결

Amazon EC2 Systems Manager(SSM)를 사용하여 Amazon GameLift Servers 콘솔에서 게임 서버 컨테이너에 연결할 수 있습니다. 이 방법을 통해 추가 설정 또는 자격 증명 관리 없이 보안 액세스를 제공합니다. 컴퓨팅 탭 또는 플릿 세부 정보 페이지의 게임 세션 탭에서 컨테이너에 연결할 수 있습니다.

1. Amazon GameLift Servers 콘솔의 탐색 창에서 관리형 컨테이너를 선택한 다음 플릿을 선택합니다.
2. 액세스하려는 컨테이너 또는 게임 세션이 포함된 플릿 ID를 선택합니다.
3. 플릿 세부 정보 페이지에서 다음 탭 중 하나를 선택합니다.
 - 컴퓨팅 - 플릿에서 실행되는 컨테이너를 나열합니다. 연결할 컨테이너를 선택합니다.

- 게임 세션 - 플릿의 게임 세션을 나열합니다. 게임 세션을 선택하여 호스팅하는 컨테이너에 연결합니다.
4. 연결을 선택합니다. 표시된 명령을 복사하여 컨테이너에 연결한 다음 연결을 다시 선택합니다.
 5. 연결 대화 상자에서 실행을 선택하여 새 SSM 세션을 생성합니다. 시스템은 AWS Key Management Service(AWS KMS)를 통해 세션을 인증하고 브라우저에서 터미널을 엽니다.
 6. 인스턴스에 연결했으면 4단계의 docker 명령을 붙여 넣고 인스턴스에서 실행하여 컨테이너에 액세스합니다.

게임 세션을 호스팅하는 컨테이너에 연결

특정 게임 세션을 호스팅하는 게임 서버 컨테이너에 연결하려면 다음 단계를 따릅니다.

1. 컴퓨팅 이름을 가져옵니다. [describe-game-sessions](#)를 호출하여 게임 세션에 ComputeName 대항를 가져옵니다.

요청

```
aws gamelift describe-game-sessions \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --game-session-id arn:aws:gamelift:us-west-2::gamesession/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/gs-1111aaaa-2222-3333-4444-5555bbbb66cc
```

응답

```
{
  "GameSessions": [
    {
      "GameSessionId": "arn:aws:gamelift:us-west-2::gamesession/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/
gs-1111aaaa-2222-3333-4444-5555bbbb66cc",
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeName": "62c5ff7f7a9a445d84877074c80aeafc",
      "Status": "ACTIVE",
      . . .
    }
  ]
}
```

응답의 ComputeName 값을 기록해 둡니다(예: 62c5ff7f7a9a445d84877074c80aeafc).

2. 컴퓨팅 액세스 및 컨테이너 속성을 가져옵니다. 플릿 ID와 컴퓨팅 이름을 사용하여 [get-compute-access](#)를 호출합니다.

응답에는 다음 필드가 포함됩니다.

- **ContainerIdentifiers** - 각 컨테이너 ContainerRuntimeId의 ContainerName 및입니다.
- **GameServerContainerGroupDefinitionArn** - 컨테이너 그룹 정의의 ARN입니다.
- **Credentials** - 인스턴스에 연결하기 위한 임시 자격 증명입니다.

요청

```
aws gamelift get-compute-access \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --compute-name 62c5ff7f7a9a445d84877074c80aeafc
```

응답

```
{
  "ComputeName": "62c5ff7f7a9a445d84877074c80aeafc",
  "ContainerIdentifiers": [
    {
      "ContainerName": "game-server",
      "ContainerRuntimeId":
"02accb92cd9bef3373300e7151d5c2b3dcca3b06eff1bb4e345085fc008d4678"
    }
  ],
  "Credentials": {
    "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "AQoDYXdzEJr...<remainder of session token>"
  },
  "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "GameServerContainerGroupDefinitionArn": "arn:aws:gamelift:us-west-2::containergroupdefinition/MyGameServerGroup"
}
```

3. 인스턴스에 연결합니다. 2단계의 자격 증명을 사용하여 플릿 인스턴스에 연결합니다. 자세한 지침은 [플릿 인스턴스에 연결](#) 섹션을 참조하세요.

4. 게임 서버 컨테이너 이름을 찾습니다. 2단계GameServerContainerGroupDefinitionArn의 를 사용하여 [describe-container-group-definition](#)을 호출하여 게임 서버 컨테이너 이름을 식별합니다.

요청

```
aws gamelift describe-container-group-definition \
  --name arn:aws:gamelift:us-west-2::containergroupdefinition/MyGameServerGroup
```

응답

```
{
  "ContainerGroupDefinition": {
    "ContainerGroupDefinitionArn": "arn:aws:gamelift:us-west-2:123456789012:containergroupdefinition/MyGameServerGroup:3",
    "Name": "MyGameServerGroup",
    "ContainerGroupType": "GAME_SERVER",
    "GameServerContainerDefinition": {
      "ContainerName": "game-server",
      . . .
    },
    . . .
  }
}
```

GameServerContainerDefinition.ContainerName 값을 기록해 둡니다(예: game-server).

5. 게임 서버 컨테이너 런타임 ID를 식별합니다. 이전 단계의 게임 서버 컨테이너 이름을 사용하여 2 단계의 get-compute-access 응답ContainerIdentifiers에서에서 일치하는 항목을 찾습니다. ContainerRuntimeId 값을 기록해 둡니다.
6. 컨테이너에 연결합니다. 를 컨테이너 IDContainerRuntimeId로 사용하고 다음 명령을 실행합니다.

```
sudo docker exec -it
02accb92cd9bef3373300e7151d5c2b3dcca3b06eff1bb4e345085fc008d4678 sh
```

Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장

인스턴스 단위로 측정된 호스팅 용량은 Amazon GameLift Servers가 동시에 호스팅할 수 있는 게임 세션 수와 해당 게임 세션에서 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.

용량은 플릿 위치 수준에서 조정됩니다. 모든 플릿에는 플릿의 홈 AWS 리전이라는 위치가 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 기반 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.

Amazon GameLift Servers 콘솔을 사용하여 대부분의 플릿 조정 작업을 수행할 수 있습니다. 옹 서비스 API와 함께 AWS SDK 또는 AWS Command Line Interface (AWS CLI)를 사용할 수도 있습니다.

[Amazon GameLift Servers](#)

주제

- [콘솔에서 플릿 용량을 관리하려면](#)
- [Amazon GameLift Servers 용량 제한 설정](#)
- [Amazon GameLift Servers 플릿의 용량을 수동으로 설정](#)
- [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#)
- [0으로/0에서 Amazon GameLift Servers 플릿 크기 조정 관리](#)
- [Amazon GameLift Servers 컨테이너 플릿 확장](#)

콘솔에서 플릿 용량을 관리하려면

1. [Amazon GameLift Servers 콘솔](#)을 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.

Amazon GameLift Servers 용량 제한 설정

수동으로 또는 자동 조정을 통해 Amazon GameLift Servers 플릿 위치의 호스팅 용량을 조정할 때는 위치의 조정 제한을 고려하세요. 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 GameLift Servers는 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 경보](#)를 설정하면 상황을 평가해 필요에 따라 수동으로 조정을 할 수 있습니다. ([결제 경보를 생성](#)하여 AWS 비용을 모니터링할 수도 있습니다.) 최소 제한은 플레이어 수요가 적은 경우에도 호스팅 가용성을 유지하는 데 유용합니다.

[Amazon GameLift Servers 콘솔](#)에서 또는 AWS Command Line Interface ()를 사용하여 플릿 위치에 대한 용량 제한을 설정할 수 있습니다AWS CLI.

용량 제한을 설정하려면

Console

1. [Amazon GameLift Servers 콘솔](#)을 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 GameLift Servers가 플릿 ID를 반환됩니다. 새 max-size 또는 min-size 값이 현재 desired-instances 설정과 충돌하는 경우 Amazon GameLift Servers는 오류를 반환합니다.

Amazon GameLift Servers 플릿의 용량을 수동으로 설정

새 플릿을 생성하면 Amazon GameLift Servers가 원하는 인스턴스를 각 플릿 위치에 있는 하나의 인스턴스로 자동 설정합니다. 그런 다음 Amazon GameLift Servers는 각 위치에 새 인스턴스를 하나씩 배

포함합니다. 플릿 용량을 변경하려면 대상 기반 Auto Scaling 정책을 추가하거나 위치에 원하는 인스턴스 수를 수동으로 설정할 수 있습니다.

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 Servers 콘솔에서 또는 AWS Command Line Interface ()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 Servers 콘솔](#)을 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 자동 크기 조정 위치에서 자동 크기 조정을 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 GameLift Servers에 알려줍니다.

Amazon GameLift Servers는 추가 인스턴스를 배포하거나 불필요한 인스턴스를 종료하여 변화에 대응합니다. Amazon GameLift Servers에서 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 GameLift Servers가 플릿 ID를 반환됩니다. 새 원하는 인스턴스 설정이 최소/최대 제한 밖일 경우 Amazon GameLift Servers는 오류를 반환합니다.

Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale

Amazon GameLift Servers에서 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 GameLift Servers에 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 Servers 콘솔, AWS Command Line Interface (AWS CLI) 또는 AWS SDK를 사용하여 대상 기반 Auto Scaling을 관리할 수 있습니다. 콘솔에서 규칙 기반 조정 정책을 볼 수 있지만 AWS CLI 또는 AWS SDK만 사용하여 규칙 기반 Auto Scaling을 관리할 수 있습니다.

주제

- [대상 기반 Auto Scaling](#)
- [규칙 기반 정책을 사용한 Auto Scaling](#)

대상 기반 Auto Scaling

Amazon GameLift Servers의 대상 기반 Auto Scaling은 플릿 지표 PercentAvailableGameSessions을 기반으로 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. PercentAvailableGameSessions은 사용 가능한 리소스의 백분율을 측정하기 때문에 실제 버퍼 크기는 전체 플릿 용량의 백분율입니다. Amazon GameLift Servers는 인스턴스를 추가하거나 제거하여 대상 버퍼 크기를 유지합니다. 버퍼 크기가 클수록 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 Servers 콘솔](#)을 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 GameLift Servers는 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 GameLift Servers 용량 제한 설정](#) 단원을 참조하십시오.
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 GameLift Servers의 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

[용 서비스 API Amazon GameLift Servers](#)와 AWS Command Line Interface 함께 AWS SDK 또는 (AWS CLI)를 사용하여 규칙 기반 정책을 생성, 업데이트 또는 삭제합니다. Amazon GameLift Servers 콘솔에서 모든 활성 정책을 확인할 수 있습니다.

플릿에 대한 모든 조정 정책을 일시적으로 중지하려면 [stop-fleet-actions](#) AWS CLI 명령을 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 GameLift Servers 용량 제한 설정](#) 단원을 참조하십시오.
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.

AvailableGameSessions가 less than 50 동안 10 minutes로 유지되면 ChangeInCapacity를 이용하여 플릿 용량을 1 instances만큼 변경합니다.

지표 이름

조정 이벤트를 시작하려면 플릿에 특징적인 다음 지표 중 하나에 Auto Scaling 정책을 연결합니다. 지표에 대한 자세한 설명은 [플릿에 대한 Amazon GameLift Servers 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 GameLift Servers에 알려줍니다. 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 GameLift Servers가 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 넘어서 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.

또한 플릿 용량은 플릿의 인스턴스 유형에 대한 제한과의 서비스 할당량에 의해 제한됩니다 AWS 계정. 이러한 제한 사항 및 계정 할당량에서 벗어나는 최소 및 최대 용량을 설정할 수 없습니다.

용량 변경 후 측정치 추적

Amazon GameLift Servers는 Auto-scaling 정책에 따라 용량을 변경하고 10분을 대기한 후 동일한 정책의 트리거에 반응합니다. 이 대기 시간 동안 Amazon GameLift Servers는 새 인스턴스 추가, 게임 서버 시작, 플레이어 연결, 새 인스턴스로부터 데이터 수집 시작 등을 수행할 수 있습니다. 그 동안 Amazon GameLift Servers는 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.

0으로/0에서 Amazon GameLift Servers 플릿 크기 조정 관리

Amazon GameLift Servers는 게임 세션 활동에 따라 인스턴스 0개로 자동 조정을 지원합니다. 이 관리형 용량 옵션을 사용하면 정의된 기간 동안 게임 세션 활동이 없으면 플릿 위치가 0개의 인스턴스로 스케일 인하고 게임 세션이 요청되면 자동으로 스케일 아웃할 수 있습니다.

인스턴스 0개로 또는 인스턴스 0개에서 확장하면 다음과 같은 몇 가지 이점이 있습니다.

- 비용 최적화 - 게임 세션 활동이 없을 때 0개의 인스턴스를 실행하여 비활성 기간 동안 컴퓨팅 비용을 제거합니다.
- 자동 사후 대응 스케일 아웃 - 플릿 위치는 게임 세션이 요청될 때 하나의 인스턴스로 자동으로 스케일 아웃되므로 수동 개입이 필요하지 않습니다.
- 간소화된 관리 - 예상 플레이어 수요 또는 개발 요구 사항에 따라 플릿 용량을 0에서 0으로 수동으로 조정할 필요가 없습니다.

플릿에서 제로로 조정을 활성화하면 Amazon GameLift 서버가 게임 세션 활동을 모니터링하고 플릿 용량을 자동으로 조정합니다.

- 0으로 스케일 인 - 게임 세션 활동 없이 구성된 기간이 지나면는 플릿 위치를 0개의 인스턴스로 Amazon GameLift Servers 스케일 인합니다.
- 0에서 스케일 아웃 - 게임 세션 생성 요청이 수신되면는 플릿 위치를 하나의 인스턴스로 Amazon GameLift Servers 스케일 아웃하여 오토 스케일링을 재개할 수 있습니다.
- 지속적인 조정 - 확장 후 구성된 Auto Scaling 정책을 사용하여 플릿을 재개하여 용량을 관리합니다.

스케일 인 동작

Amazon GameLift Servers는 구성된 비활성 기간이 경과하고 게임 세션 활동 없이 플릿 위치에 대한 스케일 인 프로세스를 시작합니다. 이는 다음과 같은 기간으로 정의됩니다.

- 플릿 위치에 활성 게임 세션이 없습니다.
- 플릿 위치에서 새 게임 세션을 생성하라는 요청이 이루어지지 않았습니다.

스케일 인 중에 Amazon GameLift Servers는 플릿 위치의 최소 및 원하는 용량을 0으로 설정하고 비용 절감을 위해 신속하게 스케일 인합니다.

스케일 아웃 동작

플릿 위치가 0 인스턴스에 있는 동안 게임 세션 생성 요청이 수신되는 경우:

- Amazon GameLift Servers는 한 인스턴스의 스케일 아웃을 즉시 시작합니다.
- 게임 세션의 배치 시도는 대기열 구성(사용된 경우)에 따라 다른 플릿 또는 플릿 위치에서 계속될 수 있습니다.

Note

0에서 스케일 아웃하려면 인스턴스를 프로비저닝하고 초기화하는 데 시간이 걸립니다. 플레이 어는 일정 기간 동안 활동이 없으면 첫 번째 게임 세션의 대기 시간이 길어질 수 있습니다. 이러한 이유로 이 기능은 다중 위치 플릿 및/또는 대기열과 가장 잘 페어링됩니다.

제로로/제로에서 스케일링 구성

제로로/제로에서 조정은 기존 플릿을 업데이트하여 구성됩니다.

Console

1. [Amazon GameLift Servers 콘솔](#)을 엽니다.
2. 탐색 창의 호스팅에서 플릿을 선택합니다.
3. 플릿 페이지에서 플릿의 세부 정보 페이지를 열기 위해 활성 플릿의 이름을 클릭합니다.
4. 크기 조정에서 크기 조정을 0으로 구성하려는 각 위치를 선택한 다음 편집을 선택합니다.
5. 조정 용량 편집 대화 상자에서 최소 용량 전략의 경우 "자동"을 선택하고, 최소 용량 설정의 경우 원하는 값을 분 단위로 설정한 다음 확인을 선택합니다.

Amazon GameLift Servers 그러면는 구성된 기간 동안 게임 세션 활동이 없으면 선택한 플릿 위치를 0개의 인스턴스로 축소합니다. 그런 다음이 위치에서 게임 세션에 대한 요청이 이루어지면 Amazon GameLift Servers는 가능한 한 빨리 하나의 인스턴스를 스케일 아웃합니다. 이 프로세스에는 약간의 시간이 소요됩니다.

AWS CLI

- 0으로/0에서 스케일을 구성합니다. 명령줄 창에서 [update-fleet-capacity](#) 명령을 플릿 ID, 위치 및 관리형 용량 구성과 함께 사용하여 0으로/0에서 스케일을 구성합니다.

```
aws gamelift update-fleet-capacity \
    --fleet-id <fleet identifier> \
    --location <location name> \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

예제:

```
aws gamelift update-fleet-capacity \
    --fleet-id
    fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --location us-west-2 \
    --desired-instances 5 \
    --max-size 10 \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

요청이 성공하면는 업데이트된 용량 구성을 Amazon GameLift Servers 반환합니다. 그러면 Amazon GameLift Servers는 구성된 기간 동안 게임 세션 활동이 없으면 선택한 플릿 위치를 0개의 인스턴스로 축소합니다. 그런 다음이 위치에서 게임 세션에 대한 요청이 이루어지면 Amazon GameLift Servers는 가능한 한 빨리 하나의 인스턴스를 스케일 아웃합니다. 이 프로세스에는 약간의 시간이 소요됩니다.

모범 사례

Scale To/From Zero를 사용할 때는 다음 권장 사항을 고려하세요.

- 적절한 비활성 기간 설정 - 비용 절감과 스케일 인/스케일 아웃 주기 빈도의 균형을 맞춥니다. 대기 시간이 0으로 짧을수록 절감 효과가 극대화되지만 콜드 스타트가 더 자주 발생합니다.
- 예측 가능한 워크로드와 함께 사용 - Scale To/From Zero는 개발/테스트 환경 또는 사용량이 적은 시간이 다른 게임과 같이 일정 기간 동안 활동이 없는 게임에 가장 적합합니다.
- CloudWatch 지표 모니터링 - 플릿 조정 이벤트 및 게임 세션 배치 시간을 추적하여 구성을 최적화합니다.
- 조정 정책과 결합 - 포괄적인 용량 관리를 위해 대상 기반 또는 규칙 기반 Auto Scaling과 함께 Scale To/From Zero를 사용합니다.

Amazon GameLift Servers 컨테이너 플릿 확장

게임 호스팅에서 가장 어려운 작업 중 하나는 필요하지 않은 리소스에 비용을 낭비하지 않고 플레이어 수요에 맞게 용량을 확장하는 것입니다. 관리형 컨테이너 플릿에서는 플릿 인스턴스를 추가하거나 제거하여 플릿 용량을 조정합니다.

새 플릿을 생성할 때는 Amazon GameLift Servers가 원하는 플릿 용량을 하나의 인스턴스로 설정하고 플릿의 홈 리전에 하나의 인스턴스를 배포합니다. 다중 위치 플릿의 경우는 Amazon GameLift Servers가 홈 리전과 각 원격 위치에 인스턴스 하나를 배포합니다. 플릿 상태가 ACTIVE에 도달하면 원하는 용량을 조정하여 스케일 다운할 용량을 늘리거나 줄일 수 있습니다.

Amazon GameLift Servers 크기 조정 기능을 사용하여 용량을 수동으로 변경하거나 플레이어 수요에 따라 자동 크기 조정을 설정할 수 있습니다.

- 대상 추적을 사용하여 자동 조정을 설정합니다. [대상 기반 Auto Scaling](#)을(를) 참조하세요.
- 플릿의 용량을 수동으로 변경합니다. [Amazon GameLift Servers 플릿의 용량을 수동으로 설정](#)을(를) 참조하세요.

컨테이너 플릿을 확장할 때 인스턴스를 추가하거나 제거하는 것이 게임 세션 및 플레이어를 호스팅하는 플릿의 용량에 어떤 영향을 미치는지 고려합니다.

- 인스턴스당 게임 세션
 - 인스턴스에서 실행되는 각 게임 서버 프로세스는 하나의 게임 세션을 호스팅하는 용량을 나타냅니다.
 - 이 공식을 사용하여 컨테이너 플릿 인스턴스에서 동시에 실행되는 게임 세션 수를 계산합니다.

$$[\text{Game sessions per instance}] = [\text{\# of game server processes per game server container}] * [\text{\# of game server container groups per instance}]$$

컨테이너 아키텍처가 게임 서버 컨테이너에서 게임 서버 프로세스 하나를 동시에 실행하는 경우 인스턴스당 게임 세션은 인스턴스당 게임 서버 컨테이너 그룹 수와 같습니다.

- 인스턴스당 게임 서버 컨테이너 그룹의 경우 [DescribeContainerFleet](#)을 호출하여 `GameServerContainerGroupsPerInstance` 또는 `MaximumGameServerContainerGroupsPerInstance` 값을 가져옵니다.
- 인스턴스에 맞는 게임 서버 컨테이너 그룹의 수는 운영 체제 및 지원 서비스에 대해 일부를 Amazon GameLift Servers 예약한 후 사용할 수 있는 메모리의 양에 따라 달라집니다. 사용 가

능한 메모리 계산 방법과 일반적인 인스턴스 유형의 참조 테이블에 대한 자세한 내용은 섹션을 참조하세요 [컨테이너 플릿 메모리 할당 이해](#).

- 인스턴스당 플레이어
 - 각 게임 세션에서 허용할 플레이어 슬롯 수를 결정합니다. 호스팅 솔루션이 게임 세션 배치를 처리하는 방식에 따라 매치메이킹 구성 또는 호출에서 게임 세션당 플레이어를 정의하여 게임 세션 배치를 시작할 수 있습니다.
 - 이 공식을 사용하여 컨테이너 플릿 인스턴스에서 동시에 게임을 플레이할 수 있는 플레이어 수를 계산합니다.

```
[Players per instance] = [# of game sessions per instance] * [# of player slots per game session]
```

컨테이너 플릿의 현재 총 용량을 가져오려면 [DescribeFleetCapacity](#) 또는 [DescribeFleetLocationCapacity](#)를 호출하여 플릿의 게임 서버 컨테이너 그룹 수를 가져옵니다. 활성 그룹은 현재 게임 세션을 호스팅하는 그룹입니다. 유휴 그룹은 새 게임 세션을 호스팅할 준비가 되었습니다. 이러한 값에 게임 서버 컨테이너 그룹당 서버 프로세스 수를 곱합니다.

Amazon GameLift Servers 모니터링

모니터링은 Amazon GameLift Servers 및 다른 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. Amazon GameLift Servers의 지표에는 다음과 같이 세 가지 주요 용도가 있습니다. 첫 번째는 시스템 상태를 모니터링하고 경보를 설정하는 것이고, 두 번째는 게임 서버 성능 및 사용량을 추적하는 것이며, 마지막으로 수동 또는 Auto Scaling을 사용하여 용량을 관리하는 것입니다.

AWS 는를 감시하고Amazon GameLift Servers, 이상이 있을 때 보고하고, 필요한 경우 자동 조치를 취할 수 있는 다음과 같은 모니터링 도구를 제공합니다.

- Amazon GameLift Servers 콘솔 - 그래픽 인터페이스를 사용하여 Amazon GameLift Servers 리소스를 관리하고 게임 호스팅 활동을 추적합니다.
- 서버 원격 측정 지표 - Amazon GameLift Servers는 SDK 및 플러그인 통합을 통해 게임 서버에서 직접 사용자 지정 지표를 내보낼 수 있습니다. 기본 제공 성능, 네트워크, 메모리, 타이밍 데이터와 함께 고유한 게임별 지표를 정의하고 추적할 수 있습니다. 모든 지표를 [Amazon Managed Service for Prometheus](#)에 게시하고 완전히 사용자 지정 가능한 대시보드를 사용하여 Amazon Grafana에서 모니터링할 수 있습니다. 다른 AWS 서비스와의 통합을 위해 Amazon CloudWatch에 게시할 수도 있습니다. 원격 측정 시스템은 완전히 사용자 지정할 수 있습니다. 사용자 지정 Prometheus 쿼리를 생성하여 기본 제공 쿼리 외에도 추가 지표를 도출할 수 있습니다. 기술 스택과 관련된 구현 가이드는 [the section called “서버 원격 측정 지표로 모니터링”](#) 섹션을 참조하세요.
- Amazon CloudWatch - AWS 서비스에서 실행 중인 다른 AWS 리소스 및 애플리케이션에 대한 Amazon GameLift Servers 지표뿐만 아니라 지표도 실시간으로 모니터링할 수 있습니다. Amazon CloudWatch는 사용자 지정 대시보드를 생성하는 도구와 지표가 지정된 임계값에 도달하면 알림을 받거나 조치를 취하는 경보를 설정하는 기능을 포함한 모니터링 기능 모음을 제공합니다.
- AWS CloudTrail -는 및 기타 AWS 서비스에 대해 AWS 계정에 의해 또는 계정을 대신하여 수행된 모든 API 호출 Amazon GameLift Servers 및 관련 이벤트를 캡처합니다. 지정한 Amazon S3 버킷에 로그 파일로 데이터를 전달합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다.
- 게임 세션 로그 - 게임 세션의 사용자 지정 서버 메시지를 Amazon S3에 저장된 로그 파일로 출력할 수 있습니다.

모니터링 소스 간 지표 비교

Amazon GameLift Servers는 Amazon GameLift Servers 콘솔 플릿 활동 지표, 서버 원격 측정 지표, Amazon CloudWatch Amazon GameLift Servers 지표의 세 가지 기본 소스를 통해 지표를 제공합니다.

각 소스의 중복과 고유한 기능을 이해하면 필요에 맞는 적절한 모니터링 접근 방식을 선택하는 데 도움이 됩니다.

소스별 지표 가용성

다음 표에는 지표 범주별로 구성된 다양한 모니터링 소스에서 사용할 수 있는 지표를 보여 줍니다.

인스턴스 지표

플릿 용량 및 상태 모니터링을 위한 인스턴스 수준 지표:

인스턴스 지표 가용성

지표	콘솔	CloudWatch	원격 측정
활성 인스턴스	✓	✓	✓
유휴 인스턴스	✓	✓	✓*
유휴 인스턴스 백분율	✓	✓	✓*
원하는 인스턴스	✓	✓	
최대 인스턴스	✓	✓	
최소 인스턴스	✓	✓	
CPU 사용률		✓	✓
네트워크 입력/출력		✓	✓
디스크/스토리지 읽기/쓰기		✓	✓
인스턴스 중단	✓	✓	
휴지통 인스턴스	✓	✓	
대체된 비정상 인스턴스		✓	

* 원격 측정 데이터를 사용하는 사용자 지정 Prometheus 쿼리를 통해 사용할 수 있습니다.

게임 서버 지표

서버 프로세스 및 게임 세션 지표:

게임 서버 지표 가용성

지표	콘솔	CloudWatch	원격 측정
활성 게임 세션	✓	✓	✓*
게임 세션 활성화	✓	✓	
사용 가능한 게임 세션	✓	✓	
사용 가능한 게임 세션 백분율	✓	✓	
동시 활성화 가능 게임 세션	✓	✓	
게임 세션 중단	✓	✓	
활성 서버 프로세스			✓
정상 게임 서버			✓
충돌된 게임 세션			✓

* 원격 측정 데이터를 사용하는 사용자 지정 Prometheus 쿼리를 통해 사용할 수 있습니다.

플레이어 사용량 지표

플레이어 세션 및 동시 사용자 지표:

플레이어 사용량 지표 가용성

지표	콘솔	CloudWatch	원격 측정
현재 플레이어 세션	✓		
플레이어 세션 활성화	✓		
글로벌 동시 사용자(CCU)			✓

지표	콘솔	CloudWatch	원격 측정
위치 동시 사용자(CCU)			✓
위치 용량			✓

컨테이너 플릿 지표

컨테이너별 지표(컨테이너 플릿의 경우 CloudWatch에서만 사용 가능):

컨테이너 플릿 지표 가용성

지표	콘솔	CloudWatch	원격 측정
활성 컨테이너 그룹		✓	✓
유휴 컨테이너 그룹		✓	✓
컨테이너 CPU/메모리 사용률		✓	✓
컨테이너 네트워크 트래픽		✓	✓
컨테이너 스토리지 작업		✓	✓

올바른 모니터링 소스 선택

특정 요구 사항에 따라 모니터링 접근 방식을 선택합니다.

- 서버 원격 측정 지표를 사용하여 게임 서버의 포괄적인 성능 데이터를 모니터링하고 게임별 사용자 지정 지표를 내보냅니다. 이러한 지표는 모든 게임 엔진 및 서버 SDK의 서버 성능, 네트워크 활동, 메모리 사용량, 타이밍 데이터에 대한 자세한 인사이트를 제공합니다. 게임 플레이 이벤트, 비즈니스 로직 성능, 애플리케이션별 데이터 포인트에 대한 사용자 지정 지표를 정의할 수 있습니다. 모든 대시보드는 완전히 사용자 지정할 수 있으며, 사용자 지정 Prometheus 쿼리를 생성하여 수집된 데이터에서 추가 지표를 도출할 수 있습니다. 자세한 내용은 [the section called “서버 원격 측정 지표로 모니터링”](#) 단원을 참조하십시오.
- Amazon GameLift Servers 콘솔을 사용하여 플릿 관리, 용량 계획, 일반적인 운영 감독을 수행합니다. 콘솔은 플릿 상태 및 플레이어 활동에 대한 통합 보기를 제공합니다. 자세한 내용은 [the section called “게임 호스팅 리소스 관리”](#) 단원을 참조하십시오.

- Amazon CloudWatch를 사용하여 자동 모니터링, 알림 및 다른 AWS 서비스와의 통합을 수행할 수 있습니다. CloudWatch는 사용자 지정 대시보드와 경고 기반 자동화를 활성화합니다. 자세한 내용은 [the section called “CloudWatch를 사용하여 모니터링”](#) 단원을 참조하십시오.
- PromQL 쿼리 기능을 사용하여 고성능 지표 수집 및 저장을 위해 [Amazon Managed Service for Prometheus](#)를 사용합니다. Prometheus는 서버 원격 측정 지표를 위한 확장 가능한 시계열 데이터 스토리지를 제공합니다.
- 고급 시각화 및 완전 사용자 지정 가능한 대시보드 작성을 위해 [Amazon Managed Grafana](#)를 사용합니다. Grafana는 사용자 지정 및 확장할 수 있는 사전 구축된 GameLift 대시보드를 제공하며, 여러 데이터 소스로 완전히 사용자 지정 대시보드 생성을 지원합니다. 사용자 지정 쿼리 및 시각화를 구축하여 게임에 중요한 지표를 추적할 수 있습니다.
- 포괄적인 모니터링을 위해 여러 소스를 사용합니다. 게임 호스팅 인프라에 대한 완전한 가시성을 확보를 위해 콘솔 감독, 서버 원격 측정 지표, 자동 알림을 결합합니다.

추가 모니터링 기능을 위해 다음을 사용할 수도 있습니다.

- [the section called “API 직접 호출 로깅”](#) - 감사 및 규정 준수를 위해 API 호출 및 관련 이벤트를 추적합니다.
- [the section called “서버 메시지 로깅”](#) - 사용자 지정 서버 메시지 및 게임 세션 로그를 캡처합니다.

주제

- [the section called “게임 호스팅 리소스 관리”](#)
- [the section called “CloudWatch를 사용하여 모니터링”](#)
- [the section called “서버 원격 측정 지표로 모니터링”](#)
- [the section called “API 직접 호출 로깅”](#)
- [the section called “서버 메시지 로깅”](#)

Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 AWS 서비스인 Amazon CloudWatch를 Amazon GameLift Servers 사용하여 모니터링할 수 있습니다. 이러한 통계는 Amazon GameLift Servers를 통해 호스팅하는 게임 서버가 어떻게 실행되고 있는지에 대한 기록 데이터를 제공하기 위해 15개월간 보관됩니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하

나 조치를 취하도록 경보를 설정할 수 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하세요.

자세한 원격 측정 지표로 서버 측 관찰성을 높이기 위해 Amazon GameLift Servers는 [Amazon Managed Service for Prometheus](#) 및 [Amazon Managed Grafana](#)를 통한 모니터링도 지원합니다. 이러한 원격 측정 지표는 다음을 포함한 자세한 서버 성능 데이터를 제공합니다.

- 서버 타이밍: 서버 델타 시간(서버 틱 속도의 일관성), 서버 틱 속도(초당 업데이트), 서버 틱 시간(틱 당 처리 시간), 서버 월드 틱 시간(게임 월드 업데이트 시간)
- 네트워크 지표: 연결(총 네트워크 연결), 바이트 및 패킷 단위의 네트워크 I/O(데이터 전송 통계), 패킷 손실(전송 실패 백분율)
- 메모리 지표: 단위의 메모리 사용량(총 메모리 사용량) 및 백분율의 물리적 메모리 사용량(메모리 사용률)
- CPU 사용량: CPU 사용량 백분율(게임 서버 프로세스별 리소스 사용률)

SDK 또는 플러그인과 관련된 구현 가이드는 [the section called “서버 원격 측정 지표로 모니터링”](#) 섹션을 참조하세요.

다음 표는 Amazon GameLift Servers에서 사용 가능한 지표와 차원을 열거한 것입니다. CloudWatch에서 사용할 수 있는 모든 지표는 Amazon GameLift Servers 콘솔에서 사용할 수 있으며, 이를 통해 데이터를 여러 개의 사용자 지정할 수 있는 그래프로 제공합니다. 게임에 대한 CloudWatch 지표 AWS Management Console에 액세스하려면 AWS CLI, 또는 CloudWatch API를 사용합니다.

지표에 위치가 없는 경우 홈 위치를 사용합니다.

Amazon GameLift Servers 지표 차원

Amazon GameLift Servers는 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.

차원	설명
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.
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 지표에 사용됩니다.
GameServerGroup	게임 서버 그룹에 대한 FleetIQ 지표를 필터링합니다.
ContainerGroupType	컨테이너 플릿 지표를 필터링하여 플릿 위치의 GameServer ContainerGroup 지표와 플릿 위치의 PerInstance ContainerGroup 지표를 구분합니다. 이 차원은 특정 컨테이너 플릿 지표(예: ContainerNetworkIn, ContainerStorageReadBytes)에 포함되어 있으므로 다양한 컨테이너 그룹 유형에 대한 지표를 구분할 수 있습니다.

플릿에 대한 Amazon GameLift Servers 지표

Amazon GameLift 네임스페이스에는 플릿 또는 플릿 그룹에서 다음과 같은 활동 관련 지표가 포함되어 있습니다. 플릿은 관리형 Amazon GameLift Servers 솔루션과 함께 사용됩니다. Amazon GameLift Servers 서비스는 매분 CloudWatch로 지표를 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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
DesiredInstances	Amazon GameLift Servers가 플릿에서 유지 관리하기 위해 작업 중인 목표 활성 인스턴스 수. Auto Scaling 기능을 사용할 경우 이 값은 현재 적용 중인 조정 정책에 따라 결정됩니다. Auto Scaling 기능을 사용하지 않을 때는 수동으로 값을 설정해야 합니다. 플릿 지표 그룹의 데이터를 확인할 때는 이 지표를 사용하지 못합니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
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: 개 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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
PercentIdleInstances	유휴 상태인 모든 활성 인스턴스의 비율($\text{IdleInstances} / \text{ActiveInstances}$로 계산). 이 지표는 Auto Scaling에 사용할 수 있습니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
RecycledInstances	재활용 및 교체된 스팟 인스턴스 수입입니다. Amazon GameLift Servers는 현재 게임 세션을 호스팅하지 않고 중단 가능성이 높은 스팟 인스턴스를 재활용합니다. 이 지표는 관리형 EC2 플릿에만 사용할 수 있습니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
RecycledSpotInstances	재활용 및 교체된 스팟 인스턴스 수입입니다. Amazon GameLift Servers는 현재 게임 세션을 호스팅하지 않고 중단 가능성이 높은 스팟 인스턴스를 재활용합니다. 이 지표는 컨테이너 플릿에서만 사용할 수 있습니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

지표	설명
InstanceSpotInterruptions	중단된 스팟 인스턴스의 수. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
CPUUtilization	EC2 지표. Amazon GameLift Servers의 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 GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스가 모든 네트워크 인터페이스에서 받은 바이트 수입입니다. 이 측정치는 단일 인스턴스에서 애플리케이션으로 들어오는 네트워크 트래픽의 볼륨을 식별합니다. 단위: 바이트

지표	설명
NetworkOut	EC2 지표. Amazon GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스가 모든 네트워크 인터페이스에서 보낸 바이트 수입입니다. 이 측정치는 단일 인스턴스에서 애플리케이션으로 나가는 네트워크 트래픽의 볼륨을 식별합니다. 단위: 바이트
DiskReadBytes	EC2 지표. Amazon GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스에 사용할 수 있는 모든 인스턴스 저장소 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 GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 인스턴스에 사용할 수 있는 모든 인스턴스 저장소 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 GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 지정된 시간 내에 인스턴스에 사용할 수 있는 모든 인스턴스 저장소 볼륨에서 읽기 작업 완료. 기간의 평균 IOPS(초당 I/O 작업 수)를 계산하려면 기간의 총 작업 수를 해당 기간의 초 수로 나누세요. 단위: 개
DiskWriteOps	EC2 지표. Amazon GameLift Servers의 경우 이 지표는 플릿 위치에 있는 모든 활성 인스턴스의 하드웨어 성능을 나타냅니다. 지정된 시간 내에 인스턴스에 사용할 수 있는 모든 인스턴스 저장소 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: 개

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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum

지표	설명
	차원: 위치
HealthyServerProcesses	정상 상태를 보고하고 있는 활성 서버 프로세스 수. 이 지표는 플릿의 전반적인 게임 서버 상태를 추적하는데 유용합니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
PercentHealthyServerProcesses	정상 상태를 보고하고 있는 모든 활성 서버 프로세스의 비율($\text{HealthyServerProcesses} / \text{ActiveServerProcesses}$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 Servers 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: 개 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: 개 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: 개 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: 개

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: 개 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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
ConcurrentActivatableGameSessions	현재 게임 세션을 호스팅하는 데 사용되지 않는 활성화 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 않는다는 점입니다. (플릿 RuntimeConfiguration 옵션 설정 MaxConcurrentGameSessionActivations 참조). 게임 세션 활성화를 제한하지 않는 플릿의 경우 이 지표는 AvailableGameSessions 과 동일합니다. 단위: 개 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($\text{AvailableGameSessions} / [\text{ActiveGameSessions} + \text{AvailableGameSessions} + \text{unhealthy server processes}]$로 계산). Auto Scaling에 사용할 수 있습니다. 단위: 백분율 관련 CloudWatch 통계: Average 차원: 위치

지표	설명
GameSessionInterruptions	중단된 스폿 인스턴스의 게임 세션 수 단위: 개 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: 개 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: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum

컨테이너 플릿 지표

컨테이너 플릿 지표는 컨테이너 그룹 상태, 리소스 사용률, 성능 지표를 포함한 컨테이너 기반 게임 호스팅에 대한 모니터링 데이터를 제공합니다.

Note

원격 측정 지표 솔루션을 사용할 때 [Amazon Managed Grafana](#)를 통해 컨테이너 플릿 모니터링을 위한 향상된 대시보드를 사용할 수 있습니다. 설정 지침은 [the section called “서버 원격 측정 지표로 모니터링”](#) 섹션을 참조하세요.

지표	설명
ActiveGameServerContainerGroups	현재 활성 상태이고 게임 세션을 호스팅할 준비가 된 컨테이너 그룹입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
IdleGameServerContainerGroups	활성 상태이지만 현재 게임 세션을 호스팅하지 않는 컨테이너 그룹입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
PendingGameServerContainerGroups	시작 중인 컨테이너 그룹입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum

지표	설명
	차원: 위치
TerminatingGameServerContainerGroups	종료 중인 컨테이너 그룹입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
UnhealthyGameServerContainerGroupsReplaced	상태 확인 실패로 인해 교체된 컨테이너 그룹입니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
ContainerCPUUtilizationPerInstance	인스턴스의 모든 컨테이너에 대한 평균 CPU 사용률입니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
ContainerCPUReservation	컨테이너에서 예약한 CPU 리소스의 백분율입니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치

지표	설명
ContainerMemoryReservation	컨테이너에서 예약한 메모리 리소스의 백분율입니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
ContainerMemoryUtilization	컨테이너에서 사용 중인 예약한 메모리의 백분율입니다. 단위: 백분율 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: 위치
ContainerNetworkIn	컨테이너에서 수신한 네트워크 바이트입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
ContainerNetworkOut	컨테이너에서 전송한 네트워크 바이트입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

지표	설명
ContainerStorageReadBytes	컨테이너가 스토리지에서 읽은 바이트입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치
ContainerStorageWriteBytes	컨테이너가 스토리지에 작성한 바이트입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: 위치

DDoS 보호(플레이어 게이트웨이) 지표

Amazon GameLift Servers 플레이어 게이트웨이 지표 `PlayerGatewayMode`는 이 `ENABLED` 또는 로 설정된 경우 DDoS 보호 계층을 통해 흐르는 UDP 트래픽을 추적합니다 `REQUIRED`. 이러한 지표는 트래픽 패턴을 모니터링하고, 잠재적 DDoS 공격을 식별하고, 릴레이 성능을 추적하는 데 도움이 됩니다.

지표	설명
PlayerGatewayPacketsIn	플레이어로부터 수신되어 DDoS 보호 계층을 통해 게임 서버로 전달된 UDP 패킷 수입니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치
PlayerGatewayBytesIn	플레이어로부터 수신되어 DDoS 보호 계층을 통해 게임 서버로 전달된 바이트 수입니다.

지표	설명
	단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치
PlayerGatewayPacketsOut	게임 서버에서 수신되어 DDoS 보호 계층을 통해 플레이어에게 전달된 UDP 패킷 수입니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치
PlayerGatewayBytesOut	게임 서버에서 수신되어 DDoS 보호 계층을 통해 플레이어에게 전달된 바이트 수입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치
PlayerGatewayPacketsThrottled	DDoS 보호 계층의 속도 제한으로 인해 삭제된 인바운드 패킷 수입니다. 단위: 개 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치

지표	설명
PlayerGatewayBytesThrottled	DDoS 보호 계층의 속도 제한으로 인해 삭제된 인바운드 바이트 수입니다. 단위: 바이트 상대적 CloudWatch 통계: Sum, Average, Minimum, Maximum 차원: FleetId, 위치
PlayerGatewayPlayerSessions	현재 DDoS 보호 계층으로 보호되는 플레이어 세션입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum 차원: FleetId, 위치

대기열에 대한 Amazon GameLift Servers 지표

Amazon GameLift 네임스페이스에는 게임 세션 배치 대기열에서 다음과 같은 활동 관련 지표가 포함되어 있습니다. 대기열은 관리형 Amazon GameLift Servers 솔루션과 함께 사용됩니다. Amazon GameLift Servers 서비스는 매분 CloudWatch로 지표를 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

지표	설명
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: 개 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. 단위: 개 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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum

지표	설명
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: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
Placement <region name>	게임 세션은 특정 리전에 위치한 플릿에 성공적으로 배치됩니다. 이 지표는 리전에 따라 PlacementSucceeded 지표를 나눕니다. 단위: 개 관련 CloudWatch 통계: Sum
PlacementsCanceled	마지막 보고 이후 시간 초과 이전에 취소된 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsFailed	마지막 보고 이후 모든 이유에 따라 실패한 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum

지표	설명
PlacementsStarted	마지막 보고 이후 대기열에 새롭게 추가된 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsSucceeded	마지막 보고 이후 새로운 게임 세션까지 이어진 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
PlacementsTimedOut	마지막 보고 이후 대기열의 시간 제한에 걸려 실행되지 못한 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
QueueDepth	대기열에서 상태가 PENDING인 게임 세션 배치 요청 수 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum 차원: 위치

매치메이킹에 대한 Amazon GameLift Servers 지표

Amazon GameLift 네임스페이스에는 매치메이킹 구성 및 매치메이킹 규칙에 대한 FlexMatch 활동과 관련한 지표가 포함되어 있습니다. FlexMatch 매치메이킹은 관리형 Amazon GameLift Servers 솔루션과 함께 사용됩니다. Amazon GameLift Servers 서비스는 매분 CloudWatch로 지표를 전송합니다.

매치메이킹 작업의 시퀀스에 대한 자세한 내용은 [Amazon GameLift Servers FlexMatch 작동 방식](#)을 참조하세요.

매치메이킹 구성

지표	설명
CurrentTickets	현재 처리되고 있거나 처리 대기 중인 매치메이킹 요청입니다. 단위: 개 상대적 CloudWatch 통계: Average, Minimum, Maximum, Sum
MatchAcceptancesTimedOut	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 수락 과정에서 시간을 초과한 잠재적 매치입니다. 단위: 개 관련 CloudWatch 통계: Sum
MatchesAccepted	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 수락된 잠재적 매치입니다. 단위: 개 관련 CloudWatch 통계: Sum
MatchesCreated	마지막 보고 이후 생성된 잠재적 매치입니다. 단위: 개 관련 CloudWatch 통계: Sum

지표	설명
MatchesPlaced	마지막 보고 이후 게임 세션에 성공적으로 배치된 매치입니다. 단위: 개 관련 CloudWatch 통계: Sum
MatchesRejected	수락을 요구하는 매치메이킹 구성의 경우, 마지막 보고 이후 적어도 한 명의 플레이어에 의해 거부된 잠재적 매치입니다. 단위: 개 관련 CloudWatch 통계: Sum
PlayersStarted	마지막 보고 이후 추가된 매치메이킹 티켓의 플레이어입니다. 단위: 개 관련 CloudWatch 통계: Sum
TicketsFailed	마지막 보고 이후 매치 실패로 이어진 매치메이킹 요청입니다. 단위: 개 관련 CloudWatch 통계: Sum
TicketsStarted	마지막 보고 이후 생성된 신규 매치메이킹 요청입니다. 단위: 개 관련 CloudWatch 통계: Sum

지표	설명
TicketsTimedOut	마지막 보고 이후 타임아웃 한도에 도달한 매치메이킹 요청입니다. 단위: 개 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
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: 개 관련 CloudWatch 통계: Sum
RuleEvaluationsFailed	마지막 보고 이후 실패한 매치메이킹 동안의 규칙 평가입니다. 이 지표는 상위 50개 규칙으로 제한됩니다. 단위: 개 관련 CloudWatch 통계: Sum

Amazon GameLift ServersFleetIQ의 측정치

Amazon GameLift 네임스페이스에는 게임 호스팅을 위한 FleetIQ 독립형 솔루션의 일부로서 FleetIQ게임 서버 그룹 및 게임 서버 활동에 대한 지표가 포함되어 있습니다. Amazon GameLift Servers 서비스는 매분 CloudWatch로 지표를 전송합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 [Amazon CloudWatch를 사용하여 오토 스케일링 및 인스턴스 모니터링](#)을 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: 개 관련 Amazon CloudWatch 통계: Sum 차원: GameServerGroup

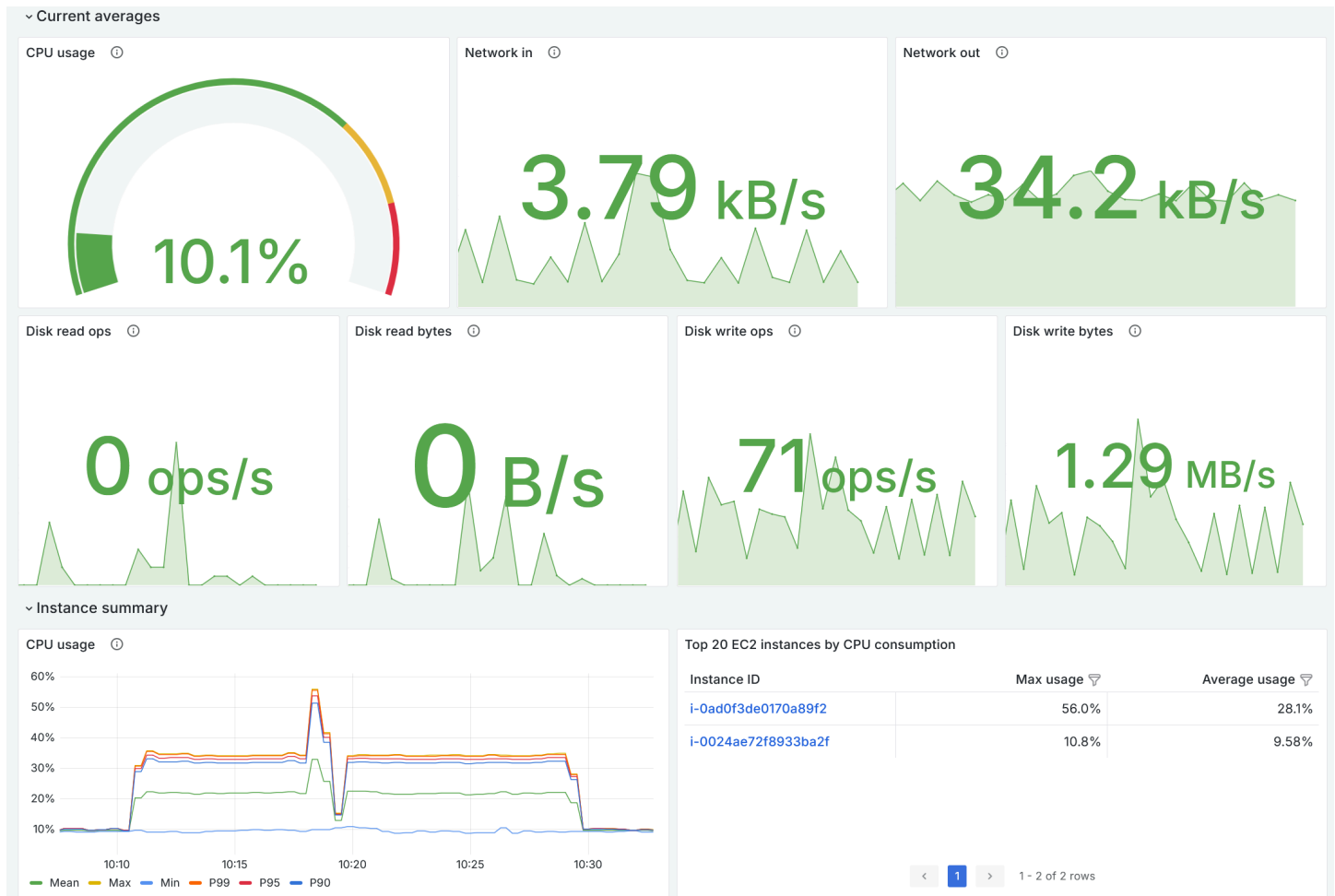
지표	설명
UtilizedGameServers	현재 게임 플레이로 점유된 게임 서버 수입니다. 이 숫자에는 UTILIZED 상태인 게임 서버가 포함됩니다. 단위: 개 관련 Amazon CloudWatch 통계: Sum 차원: GameServerGroup
DrainingAvailableGameServers	현재 게임 플레이를 지원하지 않으며 종료 예정된 인스턴스의 게임 서버 수입니다. 이러한 게임 서버는 새 클레임 요청에 대한 응답으로 클레임되는 우선 순위가 가장 낮습니다. 단위: 개 관련 Amazon CloudWatch 통계: Sum 차원: GameServerGroup
DrainingUtilizedGameServers	현재 게임 플레이를 지원하며 종료 예정된 인스턴스의 게임 서버 수입니다. 단위: 개 관련 Amazon 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 Amazon CloudWatch 통계: Average, Minimum, Maximum 차원: GameServerGroup
GameServerInterruptions	제한된 스팟 가용성으로 인해 중단된 스팟 인스턴스의 게임 서버입니다. 단위: 개 관련 Amazon CloudWatch 통계: Sum 차원: GameServerGroup, InstanceType
InstanceInterruptions	제한된 가용성으로 인해 중단된 스팟 인스턴스입니다. 단위: 개 관련 Amazon CloudWatch 통계: Sum 차원: GameServerGroup, InstanceType

서버 원격 측정 지표로 모니터링

Amazon GameLift Servers는 관리형 Amazon EC2 및 컨테이너 플릿에서 실행되는 게임 서버의 원격 측정 지표를 수집하고 게시하도록 구성할 수 있습니다. 이러한 지표는 서버 빌드와 함께 원격 측정 수집기를 배포한 후 사용할 수 있습니다. 지표 시스템은 모든 SDK(C++, C#, Go), 모든 플러그인(Unreal, Unity), Amazon GameLift Servers 게임 서버 래퍼를 지원합니다. [Amazon Managed Service](#)

for Prometheus, the section called “CloudWatch를 사용하여 모니터링”, [Amazon Managed Grafana](#) 대시보드로의 지표 데이터 흐름(시각화에 권장됨).



Amazon Managed Grafana 또는 Amazon CloudWatch에서 사전 구축된 대시보드를 사용하여 게임 서버 전반의 성능을 모니터링합니다.

원격 측정 지표의 이점

원격 측정 지표 시스템은 5가지 주요 이점을 제공합니다.

- **게임 엔진별 지표** - 게임 엔진 플러그인(Unreal, Unity)은 게임 성능 최적화에 중요한 서버 틱 시간, 프레임 속도, 엔진 수준 리소스 사용률과 같은 엔진별 성능 지표와 네이티브 통합을 제공합니다.
- **사용자 지정 지표 지원** - 서버 SDK 함수 호출을 사용하여 자체 게임별 지표를 정의하고 추적하여 게임에 중요한 사용자 지정 게임 플레이 이벤트, 비즈니스 로직 성능, 애플리케이션별 데이터 포인트를 모니터링합니다.
- **자동 수집** - 지표는 추가 계측이 필요 없고 [Amazon Managed Service for Prometheus](#) 및 Amazon CloudWatch와의 직접 통합 없이 원격 측정 수집기 배포 후 자동으로 흐름됩니다.

- 다중 수준 모니터링 - 용량 및 규모 조정에 대한 플릿 수준 지표, 리소스 사용률에 대한 인스턴스 수준 지표, 성능 추적을 위한 게임 세션 지표입니다.
- 범용 호환성 - 모든 Amazon GameLift Servers 지원되는 개발 환경과 호환되고, 모든 서버 SDK와 통합되며, 게임 엔진 플러그인에서 기본적으로 지원됩니다.

Note

원격 측정 지표는 Amazon Linux 2023 또는 Windows를 실행하는 Amazon GameLift Servers 관리형 Amazon EC2 또는 컨테이너 플릿에 사용할 수 있습니다.

시작하기 전 준비 사항

필요한 AWS 리소스

- AWS 에 대해 구성된 계정입니다 Amazon GameLift Servers.
- 다음에서 실행 중인 관리형 플릿:
 - 지원되는 운영 체제가 있는 Amazon EC2 또는
 - Amazon Linux 2023가 있는 컨테이너
- 적절한 IAM 권한

IAM 요구 사항

다음 IAM 권한은 해당 서비스를 사용할 계획인 경우에만 필요합니다.

- [Amazon Managed Service for Prometheus](#)(Prometheus에 지표를 게시하는 경우에만 필요)
 - `aps:RemoteWrite` 권한
 - Prometheus 워크스페이스에 대한 액세스
- Amazon CloudWatch(Amazon CloudWatch에 지표를 게시하는 경우에만 필요)
 - `cloudwatch:PutMetricData` 권한
 - 지표 네임스페이스에 대한 액세스
- [Amazon Managed Grafana](#)(Grafana 대시보드를 사용하는 경우에만 필요)
 - `grafana:Read` 권한
 - 대시보드 액세스를 위한 SSO 구성

구현

개발 환경에 따라 구현 경로를 선택합니다.

SDK 구현

SDK 유형	SDK 설정	사용자 지정 지표	API 참조
Go SDK	전체 설정 가이드	Go 지표 API	Go 작업 및 데이터 유형
C# SDK	전체 설정 가이드	C# 지표 API	C# 작업 및 데이터 유형
C++ SDK	전체 설정 가이드	C++ 지표 API	C++ 작업 및 데이터 유형

플러그인 구현

플러그인	플러그인 설정	사용자 지정 지표	API 참조
비현실적	전체 설정 가이드	Unreal 지표 API	Unreal 작업 및 데이터 유형
Unity	전체 설정 가이드	Unity 지표 API	C# 작업 및 데이터 유형

워크플로 구현

각 구현은 2단계 프로세스를 따릅니다.

1. 전체 설정 가이드(METRICS.md) - 인프라 배포, AWS 인프라 구성, 플릿 설정 및 Grafana 대시보드 구성.
2. API 구현 가이드(CUSTOM_METRICS.md) - 언어별 SDK 사용, 지표 유형, 사용자 지정 지표 생성, 고급 구성.

Verification(확인)

1. [Amazon Managed Service for Prometheus](#) 워크스페이스 또는 Amazon CloudWatch 콘솔에서 수신되는 원격 측정 데이터를 확인하여 지표 흐름을 검증합니다.

2. 사전 구축된 대시보드를 사용하여 [Amazon Managed Grafana](#)의 대시보드 가시성을 확인합니다.
3. 모니터링 대시보드에 표시되는지 확인하여 사용자 지정 지표를 테스트합니다.

Note

구현을 완료한 후 이 페이지로 돌아가서 [사용 가능한 지표](#) 페이지로 이동합니다.

사용 가능한 지표

지표는 세 가지 범주로 구분됩니다.

- 자동으로 수집된 지표
- SDK 제공 지표
- 사용자 지정 지표

자동 지표 수집

이러한 지표에는 코드 변경이 필요하지 않습니다.

인스턴스 지표

지표 유형	설명	사용 사례
CPU	인스턴스당 사용률	리소스 모니터링
Memory	물리적 메모리 사용량 및 백분율	용량 계획
네트워크 I/O	전송/수신된 바이트 및 패킷	연결 상태
디스크 I/O	읽기/쓰기 작업 및 처리량	스토리지 성능

플릿 지표

지표 유형	설명	사용 사례
활성 인스턴스	실행 중인 인스턴스 수	플릿 조정

지표 유형	설명	사용 사례
게임 세션	활성 및 사용 가능한 세션	용량 관리
충돌된 게임 세션	충돌된 게임 세션	오류 모니터링

SDK 제공 지표

다음 코드에는 SDK 함수 호출이 필요합니다.

서버 타이밍 지표

지표	설명	구현
서버 델타 시간	현재 서버 틱과 이전 서버 틱 간의 시간 차이입니다. 서버 틱 속도의 일관성을 측정합니다.	GetDeltaTime() 호출
서버 틱 속도	서버가 업데이트를 처리하는 초당 횟수를 표시합니다.	자동으로 계산됨
서버 틱 시간	서버가 단일 틱 또는 업데이트를 처리하는 데 걸리는 시간	GetTickTime() 호출
서버 월드 틱 시간	서버가 각 틱으로 게임 월드를 업데이트하는 데 걸리는 시간	GetWorldUpdateTime() 호출

구현: 엔진에 구애받지 않는 SDK(C++, C#, Go)의 경우 계산된 타이밍 값으로 게임 루프에서 SDK 함수를 호출하여 이러한 지표를 구현합니다. 엔진 플러그인(Unreal, Unity)의 경우 이러한 지표는 엔진 통합을 통해 자동으로 캡처됩니다.

네트워크 지표

지표	설명	구현
연결	서버가 설정한 총 네트워크 연결 수	InitMetrics() 이후 자동

지표	설명	구현
네트워크 I/O(바이트)	서버가 네트워크를 통해 송신 및 수신하는 총 바이트 수	InitMetrics() 이후 자동
네트워크 I/O(패킷)	서버가 송신 및 수신하는 총 네트워크 패킷 수	InitMetrics() 이후 자동
패킷 손실	전송 중에 손실되는 네트워크 패킷의 백분율	InitMetrics() 이후 자동

구현: SDK 함수 호출을 네트워킹 라이브러리와 통합합니다. SDK는 다양한 네트워크 구현에 대한 지침을 제공합니다.

프로세스 지표

지표	설명	구현
CPU 사용량(%)	게임 서버 프로세스에서 사용 중인 CPU 리소스의 백분율	InitMetrics() 이후 자동
메모리 사용량(단위)	서버 프로세스에서 사용 중인 총 메모리 양	InitMetrics() 이후 자동
물리적 메모리 사용량(%)	현재 사용 중인 서버의 총 물리적 메모리 백분율	InitMetrics() 이후 자동
서버 상태	게임 서버 상태	InitMetrics() 이후 자동

구현: 이러한 지표는 각 게임 세션 프로세스에 대해 SDK에서 자동으로 수집됩니다.

프로세스별 대시보드 조직

프로세스별 지표는 두 가지 특수 대시보드에서 사용할 수 있습니다.

- 서버 성능 대시보드 - 개별 게임 세션에 대한 서버 타이밍(델타 시간, 틱 속도, 틱 시간, 월드 틱 시간), 네트워크 지표(연결, I/O 바이트/패킷, 패킷 손실), 메모리 사용량, CPU 사용량을 포함합니다.

- 인스턴스 성능 대시보드 - 인스턴스 리소스 소비에 가장 많이 기여하는 프로세스를 식별하는 데 도움이 되는 '상위 N개 메모리 소비 게임 세션' 및 '상위 N개 CPU 소비 게임 세션' 표를 제공합니다. 게임 세션 링크를 클릭하면 세부 지표를 심층적으로 조사할 수 있습니다.

프로세스별 지표 사용 사례

프로세스별/게임 세션별 지표는 다음 모니터링 시나리오를 지원합니다.

- 심층 성능 조사 - 특정 프로세스 또는 게임 세션으로 인해 호스트/인스턴스의 성능이 저하된 경우 프로세스별 지표는 상위 CPU 및 메모리 소비 게임 세션 표를 통해 문제가 발생한 프로세스를 식별하는 데 도움이 됩니다.
- 게임 서버 충돌 조사 - 게임 세션이 충돌할 때 이러한 지표는 충돌이 메모리 부족, CPU 과부하 또는 네트워크 대역폭 문제로 인한 것인지 확인하는 데 도움이 됩니다.
- 플레이어 보고 문제 조사 - 플레이어가 게임 플레이 중에 지연 또는 중단을 보고하면 프로세스별 지표는 CPU, 메모리, 네트워크, 틱 시간 또는 월드 업데이트 시간의 병목 현상을 식별하는 데 도움이 됩니다.
- 다양한 빌드의 성능 변화 식별 - 틱 시간, 틱 속도, 월드 업데이트 시간 지표를 통해 개발자는 다양한 서버 빌드에서 게임 성능이 어떻게 변경되는지 측정할 수 있습니다.
- 게임 플레이의 지연 및 느려짐 감지 - 틱 시간, 틱 속도, 월드 업데이트 시간 지표는 서버가 게임을 업데이트하는 속도를 반영하므로 플레이어 경험에 직접적인 영향을 미칩니다.
- 벤치마킹 - 플레이어 수, 게임 모드, 기타 변수와 같은 요인을 기반으로 다양한 게임 시나리오가 서버 성능에 미치는 영향을 식별합니다.

대시보드 조직

지표는 다양한 모니터링 시나리오를 위해 [Amazon Managed Grafana](#)의 특수 대시보드로 구성됩니다. 사용 가능한 대시보드는 플릿 유형에 따라 다릅니다.

EC2 플릿 대시보드

- EC2 플릿 개요 대시보드 - 상위 수준 플릿 용량, 조정 인사이트, 동시 플레이어(CCU), 인스턴스, 플레이어 용량, 충돌된 게임 세션.
- 인스턴스 개요 대시보드 - 평균 CPU, 메모리, 네트워크, 디스크 사용률을 포함한 모든 인스턴스의 집계된 호스트 수준 지표.
- 인스턴스 성능 대시보드 - 리소스 집약적인 프로세스를 식별하기 위한 '상위 N개 메모리 소비 게임 세션' 및 '상위 N개 CPU 소비 게임 세션' 표가 있는 개별 인스턴스에 대한 세부 지표.

- 서버 성능 대시보드(EC2) - EC2 인스턴스의 개별 게임 세션에 대한 게임 루프 타이밍, 네트워크 성능, 메모리, CPU 지표.

컨테이너 플릿 대시보드

- 컨테이너 플릿 개요 대시보드 - CPU 예약, 메모리 사용률, 컨테이너 그룹 상태를 포함한 컨테이너 플릿 리소스 사용률에 대한 상위 수준 개요.
- 컨테이너 성능 대시보드 - CPU 사용률, 메모리 사용량, 네트워크 I/O, 스토리지 성능을 포함하여 특정 ECS 작업 내의 개별 컨테이너에 대한 세부 지표.
- 서버 성능 대시보드(컨테이너) - 컨테이너의 개별 게임 세션에 대한 게임 루프 타이밍, 네트워크 성능, 메모리, CPU 지표.

자세한 대시보드 정보와 사용 지침은 [the section called “대시보드 조직 및 사용량”](#) 섹션을 참조하세요.

작동 방식

원격 측정 지표 시스템은 게임 서버에서 시각화 대시보드로의 간단한 4단계 데이터 흐름을 따릅니다.

컬렉션: GameLift Server SDK 또는 플러그인과 통합된 게임 서버는 동일한 인스턴스에서 실행되는 로컬 원격 측정 수집기에 지표를 자동으로 내보냅니다. SDK는 코드에서 정의한 자동 지표(서버 수명 주기, 리소스 사용량) 및 사용자 지정 지표를 모두 캡처합니다.

처리: 원격 측정 수집기는 게임 서버의 지표를 집계하여 인스턴스 수준 성능 데이터(CPU, 메모리, 네트워크, 디스크 사용량)와 결합합니다. 이를 통해 게임 성능과 기본 인프라를 모두 전체적으로 파악할 수 있습니다.

스토리지: 처리된 지표는 선택한 지표 웨어하우스로 내보내집니다. 고성능 시계열 스토리지를 위한 [Amazon Managed Service for Prometheus](#), AWS 서비스 통합을 위한 Amazon CloudWatch 또는 둘 다. 모든 데이터 전송은 인증되고 암호화됩니다.

시각화: [Amazon Managed Grafana](#)는 지표 웨어하우스에 연결하여 사전 구축된 GameLift 대시보드를 표시합니다. 이러한 대시보드는 게임 호스팅 인프라를 모니터링하고 문제를 해결하는 데 도움이 되는 플릿 개요, 서버 성능 세부 정보, 컨테이너 모니터링 보기를 제공합니다.

Note

게임 서버와 원격 측정 수집기 간의 모든 지표 전송은 보안을 위해 인스턴스에서 로컬로 이루어집니다. 수집기만 적절한 인증을 사용하여 AWS 서비스와 통신합니다.

대시보드 조직 및 사용량

[Amazon Managed Grafana](#)의 포괄적인 대시보드에서 지표를 봅니다. 사용 가능한 대시보드는 플릿 유형에 따라 다릅니다.

플릿 유형별 대시보드 가용성

다음 표에는 각 플릿 유형에 사용할 수 있는 대시보드를 보여 줍니다.

대시보드	플릿 유형	설명
EC2 플릿 개요	EC2 Fleet	동시 플레이어(CCU), 인스턴스, 플레이어 용량에 대한 정보를 표시합니다.
인스턴스 개요	EC2 Fleet	모든 플릿 인스턴스의 평균 CPU, 메모리, 네트워크 사용률을 표시합니다.
인스턴스 성능	EC2 Fleet	개별 인스턴스에 대한 세부 지표(CPU, 메모리, 디스크, 네트워크)를 표시합니다.
컨테이너 플릿 개요	컨테이너 플릿	관리형 컨테이너 플릿에 있는 모든 컨테이너의 평균 리소스 사용률을 표시합니다.
컨테이너 성능	컨테이너 플릿	특정 ECS 작업 내의 개별 컨테이너에 대한 세부 지표를 표시합니다.
서버 성능	둘 다	지정된 게임 서버 프로세스의 네트워크, 메모리, 런타임 성능을 표시합니다 (EC2 및 컨테이너 플릿의 별도 버전).

관리형 EC2 플릿:

- EC2 플릿 개요는 높은 수준의 플릿 용량과 규모 조정 인사이트를 제공합니다.
- 호스트 수준 모니터링을 위해 인스턴스 개요 및 인스턴스 성능 대시보드를 사용합니다.
- 시스템 수준 가시성을 위해 호스트 지표 수신기를 통해 수집된 지표입니다.
- EC2 인스턴스 리소스 사용률 및 성능에 중점을 둡니다.

- 서버 성능(EC2)은 기본 인프라와 관계없이 게임 서버 애플리케이션 지표를 모니터링합니다.

관리형 컨테이너 플릿:

- ECS 작업 및 컨테이너 수준 모니터링을 위해 컨테이너 플릿 개요 및 컨테이너 성능 대시보드를 사용합니다.
- 컨테이너화된 워크로드 가시성을 위해 ECS 컨테이너 수신기를 통해 수집된 지표입니다.
- 작업 수준 집계 및 컨테이너 리소스 격리에 중점을 둡니다.
- 서버 성능(컨테이너)은 기본 인프라와 관계없이 게임 서버 애플리케이션 지표를 모니터링합니다.

EC2 플릿 개요 대시보드

이 대시보드는 전 세계 및 위치별로 플릿의 사용률 및 용량에 대한 개략적인 개요를 제공합니다. 게임 서버 중지, 시작 및 충돌수뿐만 아니라 정상 게임 서버의 비율을 보여주는 그래프가 특징입니다. FleetID 및 위치를 기준으로 필터링할 수 있습니다.

플릿 개요 지표

다음 표에는 플릿 개요 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
글로벌 CCU 및 용량	CCU	모든 위치의 모든 인스턴스에 대해 모든 게임 서버의 동시 사용자 수를 표시합니다.
	글로벌 CCU	모든 인스턴스에 대해 전역적으로 모든 게임 서버의 동시 사용자 수를 표시합니다.
	활성 인스턴스	활성 상태인 플릿의 총 인스턴스 수를 표시합니다.
	활성 프로세스	게임 세션을 호스팅할 준비가 된 활성 게임 서버 프로세스 수를 표시합니다.
	게임 서버 시작	플릿 전체에서 시작된 게임 세션 수를 표시합니다.

대시보드	지표 이름	정의
글로벌 CCU 및 용량	정상 게임 서버	플릿 전체에서 Amazon GameLift Servers에 정상으로 보고되는 게임 서버의 평균 백분율을 표시합니다.
	충돌된 게임 세션	충돌된 게임 세션의 게임 세션 ID를 표시합니다. 게임 세션 ARN의 링크를 클릭하여 서버 성능 대시보드로 이동합니다.
위치 CCU	위치 플레이어	해당 위치의 모든 인스턴스에 걸친 모든 게임 서버 내의 동시 접속 플레이어 수를 보여줍니다
	위치 용량	위치의 용량 사용률(%)과 해당 위치에서 사용 중인 게임 서버의 비율을 표시합니다.

Note

CCU 지표는 게임 서버 코드에서 구현해야 합니다. 이러한 지표는 자동으로 수집되지 않으므로 애플리케이션에서 구현하고 보고해야 합니다.

인스턴스 개요 대시보드

이 대시보드는 플릿의 모든 인스턴스에서 집계된 호스트 수준 지표를 제공합니다. 현재 평균은 인스턴스의 전체 상태를 보여줍니다. 성능이 저하되면 CPU 사용량, 메모리 사용량, 네트워크, 디스크 사용량에 병목 현상이 있는지 확인합니다. FleetID 및 위치를 기준으로 필터링할 수 있습니다.

인스턴스 개요 지표

다음 표에는 인스턴스 개요 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
인스턴스 요약	평균 CPU 사용량	인스턴스 요약은 인스턴스별 평균 리소스 사용량을 추적합니다. 모든 플릿 인스턴스에서 사용 중인 CPU 리소스의 평균 백분율입니다. 높은 사

대시보드	지표 이름	정의
		용률 패턴을 사용하려면 개별 인스턴스 지표 및 리소스 배포를 확인해야 합니다.
	피크 CPU 사용량	해당 기간 동안 플릿의 모든 인스턴스에서 기록된 가장 높은 CPU 사용률
	인스턴스별 CPU 사용량	플릿의 각 인스턴스에 대한 개별 CPU 사용률로 리소스 핫스팟 식별 가능
인스턴스 요약	평균 메모리 사용량	모든 플릿 인스턴스에서 사용 중인 RAM의 평균 백분율입니다. 높은 사용률 패턴을 사용하려면 메모리 소비와 같은 개별 인스턴스 지표 및 리소스 배포를 확인해야 합니다.
	최대 메모리 사용량	해당 기간 동안 플릿의 모든 인스턴스에서 기록된 가장 높은 메모리 사용률
	인스턴스별 메모리 사용량	플릿의 각 인스턴스에 대한 개별 메모리 사용률을 통해 메모리 집약적인 워크로드 식별
네트워크 요약	네트워크 I/O(바이트)	네트워크 요약은 평균 인스턴스 연결을 반영합니다. 모든 플릿 인스턴스의 평균 네트워크 트래픽 볼륨(송신 및 수신). 지연 시간 또는 연결 문제의 경우 네트워크 I/O 및 패킷 속도와 같은 개별 인스턴스 지표를 확인합니다.
	네트워크 I/O(패킷)	모든 플릿 인스턴스의 평균 네트워크 패킷 전송 속도(송신 및 수신).
	인스턴스별 네트워크 사용량	네트워크 병목 현상을 식별하는 데 유용한 각 인스턴스의 개별 네트워크 사용률
디스크 요약	디스크 I/O 작업	디스크 요약은 인스턴스의 평균 디스크 성능을 나타냅니다. 모든 플릿 인스턴스의 평균 디스크 읽기/쓰기 작업입니다. 응답 시간이 느리면 개별 인스턴스 수준에서 디스크 작업, I/O 대기 시간, 대기 중인 작업을 검사하는 것이 좋습니다.

대시보드	지표 이름	정의
	디스크 I/O 처리량	모든 플릿 인스턴스의 평균 디스크 읽기/쓰기 처리량입니다.
	인스턴스별 디스크 사용량	각 인스턴스의 개별 디스크 사용률로 스토리지 성능 문제 식별

Note

인스턴스 수준 지표는 호스트 지표 수신기를 통해 수집되며 플릿의 인프라 성능에 대한 시스템 수준 가시성을 제공합니다. 이 대시보드를 사용하여 전체 플릿 상태 추세를 식별하고 성능 문제가 감지되면 개별 인스턴스로 드릴다운합니다.

인스턴스 성능 대시보드

이 대시보드는 개별 인스턴스에 대한 자세한 성능 지표를 제공합니다. 현재 평균은 인스턴스의 전체 상태를 보여 줍니다. 성능이 저하되면 CPU 사용량, 메모리 사용량, 파일 시스템 사용량에 병목 현상이 있는지 확인합니다. 인스턴스 리소스 소비에 가장 많이 기여하는 프로세스를 식별하는 데 도움이 되는 '상위 N개 메모리 소비 게임 세션' 및 '상위 N개 CPU 소비 게임 세션' 표를 제공합니다. 게임 세션 링크를 클릭하면 세부 지표를 심층적으로 조사할 수 있습니다. 특정 인스턴스 ID를 기준으로 필터링할 수 있습니다.

플릿 성능 지표

다음 표에는 인스턴스 성능 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
로드 요약	상위 N개 메모리 소비 게임 세션	로드 요약은 게임 서버의 리소스 사용량을 추적합니다. 이 인스턴스에서 가장 많은 메모리 리소스를 소비하는 게임 세션의 순위 목록입니다. 높은 사용률 패턴을 사용하려면 개별 세션 지표 및 리소스 배포를 확인해야 합니다. 게임 세션 링크를 클릭하면 세부 지표를 심층적으로 조사할 수 있습니다.

대시보드	지표 이름	정의
현재 평균	상위 N개 CPU 소비 게임 세션	이 인스턴스에서 가장 많은 CPU 리소스를 소비하는 게임 세션의 순위 목록입니다. 높은 사용률 패턴을 사용하려면 개별 세션 지표 및 CPU 및 메모리와 같은 리소스 배포를 확인하도록 요구합니다. 게임 세션 링크를 클릭하면 세부 지표를 심층적으로 조사할 수 있습니다.
	게임 세션당 CPU 사용량	각 활성 게임 세션의 리소스 소비를 보여주는 개별 CPU 사용률 분석
	게임 세션당 메모리 사용량	각 활성 게임 세션의 리소스 소비를 보여주는 개별 메모리 사용률 분석
	인스턴스 CPU 사용량	시간 경과에 따른 선택한 인스턴스의 전체 CPU 사용률
	인스턴스 메모리 사용량	시간 경과에 따른 선택한 인스턴스의 전체 메모리 사용률
	인스턴스 파일 시스템 사용량	스토리지 용량 문제를 식별하는 데 유용한 선택된 인스턴스의 파일 시스템 사용량
Network	인스턴스 네트워크 I/O	네트워크는 인스턴스 연결을 반영합니다. 선택한 인스턴스의 네트워크 트래픽 볼륨 및 패킷 속도입니다. 지연 시간 또는 연결 문제의 경우 네트워크 I/O, 패킷 속도, 오류 수 조사
	네트워크 연결	선택한 인스턴스의 활성 네트워크 연결 수
	네트워크 오류 수	연결 문제를 식별하기 위한 네트워크 오류 통계
디스크	디스크 I/O 작업	디스크는 디스크 성능을 나타냅니다. 선택한 인스턴스에 대한 디스크 읽기/쓰기 활동, 사용률입니다. 응답 시간이 느리면 디스크 작업, I/O 대기 시간, 대기 중인 작업을 검사하는 것이 좋습니다.

대시보드	지표 이름	정의
	디스크 I/O 대기 시간	디스크 작업이 완료될 때까지 대기하는 데 소요된 시간
	대기 중 디스크 작업	처리 대기 중인 디스크 작업 수

Note

상위 N개 메모리 및 CPU 소비 게임 세션 테이블은 전체 인스턴스 성능에 영향을 미칠 수 있는 성능 병목 현상과 리소스 집약적인 프로세스를 식별하는 데 필수적입니다. 이러한 순위를 통해 추가 조사를 위해 문제가 있는 게임 세션을 빠르게 식별할 수 있습니다.

컨테이너 플릿 개요 대시보드

이 대시보드는 컨테이너 플릿의 리소스 사용률 및 용량에 대한 개략적인 개요를 제공합니다. CPU 예약, 메모리 사용률, 컨테이너 그룹 상태를 포함한 관리형 컨테이너 플릿에 있는 모든 컨테이너의 평균 리소스 사용률을 표시합니다. FleetID 및 위치를 기준으로 필터링할 수 있습니다.

컨테이너 플릿 개요 지표

다음 표에는 컨테이너 플릿 개요 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
컨테이너 그룹 상태	활성 게임 서버 컨테이너 그룹	현재 활성 상태이고 게임 세션을 호스팅할 준비가 된 컨테이너 그룹
	유휴 게임 서버 컨테이너 그룹	활성 상태이지만 현재 게임 세션을 호스팅하지 않는 컨테이너 그룹
	보류 중인 게임 서버 컨테이너 그룹	시작 중인 컨테이너 그룹입니다.
	종료 중인 게임 서버 컨테이너 그룹	종료 중인 컨테이너 그룹

대시보드	지표 이름	정의
리소스 사용률	컨테이너 CPU 사용률	플릿의 모든 컨테이너에 대한 평균 CPU 사용률
	컨테이너 메모리 사용률	플릿의 모든 컨테이너에 대한 평균 메모리 사용률
	컨테이너 CPU 예약	플릿 전체에서 컨테이너가 예약한 CPU 리소스의 백분율
네트워크 활동	컨테이너 네트워크 입력	플릿 전체에서 컨테이너가 수신한 평균 네트워크 바이트
	컨테이너 네트워크 출력	플릿 전체에서 컨테이너가 보낸 평균 네트워크 바이트

Note

컨테이너 플릿 지표는 ECS 컨테이너 수신기를 통해 수집되며 작업 수준 집계 및 컨테이너 리소스 격리에 중점을 두고 컨테이너화된 워크로드 가시성을 제공합니다.

컨테이너 성능 대시보드

이 대시보드는 특정 ECS 작업 내의 개별 컨테이너에 대한 자세한 성능 지표를 제공합니다. CPU 사용률, 메모리 사용량, 네트워크 I/O, 스토리지 성능 등 개별 컨테이너에 대한 자세한 지표를 표시합니다. 특정 컨테이너 ID 또는 ECS 작업을 기준으로 필터링할 수 있습니다.

컨테이너 성능 지표

다음 표에는 컨테이너 성능 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
리소스 성능	컨테이너 CPU 사용률	시간 경과에 따른 선택한 컨테이너의 CPU 사용률

대시보드	지표 이름	정의
	컨테이너 메모리 사용률	시간 경과에 따른 선택한 컨테이너의 메모리 사용률
	컨테이너 메모리 예약	선택한 컨테이너에서 예약한 메모리 리소스의 백분율
네트워크 성능	컨테이너 네트워크 I/O	선택한 컨테이너의 네트워크 트래픽 볼륨(바이트 입/출력)
	네트워크 연결	선택한 컨테이너의 활성 네트워크 연결 수
스토리지 성능	컨테이너 스토리지 읽기/쓰기	선택한 컨테이너의 스토리지 읽기 및 쓰기 활동
	컨테이너 스토리지 I/O 작업	선택한 컨테이너에서 수행한 스토리지 I/O 작업 수

Note

컨테이너 성능 지표는 ECS 작업 내의 개별 컨테이너 리소스 소비 및 성능 특성에 대한 자세한 가시성을 제공합니다.

서버 성능 대시보드

서버 성능 대시보드에는 개별 게임 세션의 서버 타이밍, 네트워크 활동, 메모리, CPU 사용량과 관련된 지표가 표시됩니다. 게임 세션 ID를 기준으로 필터링하고 지표를 Amazon CloudWatch 또는 [Amazon Managed Grafana](#)로 직접 내보낼 수 있습니다.

서버 성능 지표

다음 표에는 서버 성능 대시보드에서 사용할 수 있는 지표를 보여 줍니다.

대시보드	지표 이름	정의
서버 타이밍	서버 델타 시간	현재 서버 틱과 이전 서버 틱 간의 시간 차이입니다. 서버 틱 속도의 일관성을 측정한 값입니다.
	서버 틱 속도	서버가 업데이트를 처리하는 초당 횟수를 표시합니다.
	서버 틱 시간	서버가 단일 틱 또는 업데이트를 처리하는 데 걸리는 시간
	서버 월드 틱 시간	서버가 각 틱으로 게임 월드를 업데이트하는 데 걸리는 시간
Network	연결	서버가 설정한 총 네트워크 연결 수
	네트워크 I/O(바이트)	서버가 네트워크를 통해 송신 및 수신하는 총 바이트 수
	네트워크 I/O(패킷)	서버가 송신 및 수신하는 총 네트워크 패킷 수
	패킷 손실	전송 중에 손실되는 네트워크 패킷의 백분율
Memory	메모리 사용량(단위)	서버 프로세스에서 사용 중인 총 메모리 양
	물리적 메모리 사용량(%)	현재 사용 중인 서버의 총 물리적 메모리 백분율
CPU 사용량	CPU 사용량(%)	게임 서버 프로세스에서 사용 중인 CPU 리소스의 백분율

일반적인 모니터링 시나리오

심층 성능 조사 분석

시나리오: 특정 프로세스 또는 게임 세션으로 인해 호스트/인스턴스 성능이 저하되고 있음

조사 단계:

- 인스턴스 성능 대시보드에 액세스합니다.
- '상위 N개 메모리 소비 게임 세션' 테이블을 검토하여 인스턴스 메모리 소비에 가장 많이 기여하는 프로세스를 식별합니다.
- '상위 N개 CPU 소비 게임 세션' 테이블을 검토하여 인스턴스 CPU 사용률에 가장 많이 기여하는 프로세스를 식별합니다.
- 게임 세션 링크를 클릭하여 세부 지표를 심층적으로 조사할 수 있습니다.
- 서버 타이밍(서버 델타 시간, 서버 틱 속도, 서버 틱 시간, 서버 월드 틱 시간)을 분석하여 성능 병목 현상을 식별합니다.

게임 서버 충돌 조사

시나리오: 게임 세션이 중단되어 근본 원인을 파악해야 함

조사 단계:

- 충돌한 게임 세션에 대한 서버 성능 대시보드에 액세스합니다.
- 메모리 사용량(단위) 및 물리적 메모리 사용량(%)을 확인하여 충돌이 메모리 부족으로 인한 것인지 확인합니다.
- CPU 사용량(%)을 검토하여 CPU 과부하로 인해 충돌이 발생했는지 확인합니다.
- 네트워크 I/O(바이트) 및 네트워크 I/O(패킷)를 분석하여 네트워크 대역폭 문제가 충돌에 기여했는지 확인합니다.
- 패킷 손실 비율을 검사하여 네트워크 관련 문제를 식별합니다.

플레이어 보고 문제 조사

시나리오: 플레이어가 게임 플레이 중 지연 또는 중단을 보고합니다.

조사 단계:

- 영향을 받은 게임 세션에 대한 서버 성능 대시보드에 액세스합니다.
- 서버 틱 시간 및 서버 월드 틱 시간을 검토하여 게임 업데이트 지연을 식별합니다.
- 서버 틱 속도를 확인하여 서버 업데이트 빈도가 일정한지 확인합니다.
- CPU 사용량(%)을 분석하여 처리 병목 현상을 식별합니다.
- 메모리 사용량 지표를 검토하여 메모리 관련 성능 문제를 식별합니다.
- 네트워크 I/O 지표 및 패킷 손실을 확인하여 네트워크 병목 현상을 식별합니다.

다양한 게임 서버 빌드의 성능 변경 사항 식별

시나리오: 다양한 서버 빌드에서 게임 성능이 어떻게 변하는지 측정하려고 합니다.

조사 단계:

- 서버 틱 시간 지표를 서로 다른 빌드 간에 비교하여 처리 효율성 변화를 측정합니다.
- 빌드 간 서버 틱 속도 일관성을 분석하여 성능 역행을 식별합니다.
- 서버 월드 틱 시간을 검토하여 게임 월드 업데이트 성능 변경 사항을 측정합니다.
- 빌드 간 메모리 사용량 패턴을 비교하여 메모리 최적화 개선 또는 역행을 식별합니다.
- CPU 사용량 추세를 모니터링하여 계산 효율성 변화를 평가합니다.

게임 플레이어의 지연 및 느려짐 감지

시나리오: 서버 응답성과 게임 업데이트 속도를 모니터링해야 함

조사 단계:

- 서버 틱 시간을 모니터링하여 서버가 각 업데이트 주기를 처리하는 속도를 측정합니다.
- 서버 틱 속도를 추적하여 초당 일관된 게임 상태 업데이트를 보장합니다.
- 서버 월드 틱 시간을 분석하여 게임 월드 업데이트 속도를 측정하면 고객 경험에 직접적인 영향을 미칩니다.
- 서버 델타 시간 변형에 대한 알림을 설정하여 일관되지 않은 서버 성능을 감지합니다.

다양한 게임 시나리오 벤치마킹

시나리오: 다양한 게임 시나리오가 서버 성능에 미치는 영향을 파악하고자 합니다.

조사 단계:

- 다양한 플레이어 수의 서버 성능 지표를 비교하여 조정 영향을 이해합니다.
- 서버 틱 시간 및 CPU 사용량 지표를 사용하여 게임 모드 간의 성능 차이를 분석합니다.
- 다양한 게임 시나리오에서 메모리 사용량 패턴을 모니터링하여 리소스 집약적인 기능을 식별합니다.
- 네트워크 I/O 지표를 추적하여 다양한 게임 플레이 시나리오의 대역폭 요구 사항을 이해합니다.
- 인스턴스 성능 대시보드를 사용하여 리소스 소비가 가장 높은 게임 세션을 생성하는 게임 시나리오를 식별합니다.

높은 리소스 사용률 응답

시나리오: 비정상적인 리소스 스파이크(CPU 85% 초과, 메모리 90% 초과)

조사 단계:

영향을 받는 리소스 식별

- DescribeGameSessionDetails API를 사용합니다.
- 필요한 경우 상태별로 필터링합니다.
- 영향을 받는 인스턴스를 문서화합니다.

리소스 사용량 분석

- 인스턴스 개요 대시보드를 검토합니다.
- 플릿의 사용률을 비교합니다.
- 과거 패턴을 확인합니다.

게임 서버 영향 모니터링

- 서버 성능 지표를 확인합니다.
- 틱 시간 및 패킷 손실을 검토합니다.
- 메모리 누수를 모니터링합니다.

해결 단계

- 세션 로그를 다운로드합니다.
- 빌드 문제를 해결합니다.
- 개선 사항을 모니터링합니다.

게임 서버 충돌 분석

시나리오: 플릿 전반의 여러 오류 상태 게임 세션

조사 단계:

초기 평가

- 플릿 개요 대시보드에 액세스합니다.
- 충돌한 세션 테이블을 검토합니다.
- 타이밍/위치의 패턴을 기록해 둡니다.

성능 분석

- 서버 타이밍 지표를 확인합니다.
- 리소스 사용률을 검토합니다.
- 네트워크 성능을 모니터링합니다.

인프라 검토

- 플릿 용량을 확인합니다.
- 인스턴스 상태를 확인합니다.
- 조정 정책을 검토합니다.

해결 경로

- 서버 로그를 분석합니다.
- 코드 최적화를 검토합니다.
- 수정 사항을 구현합니다.

플릿 용량 최적화

시나리오: 게임 시작 또는 벤치마크 연구

분석 단계:

리소스 사용률

- 위치별로 필터링합니다.
- P50/P95/P99 지표를 검토합니다.
- 사용 패턴을 분석합니다.

인스턴스 유형 분석

- 유형별로 성능을 비교합니다.
- 확장 대상을 식별합니다.
- 사용률 패턴을 문서화합니다.

최적화 작업

- 조정 정책을 조정합니다.
- 인스턴스 유형을 수정합니다.
- 플릿 구성을 업데이트합니다.

문제 해결 가이드

일반적인 문제 및 해결 단계

누락되거나 불완전한 지표

증상

- 대시보드에 지표가 표시되지 않음.
- 부분적인 지표 수집.
- 지연된 지표 업데이트.

해결 단계

A. 수집기 상태 확인

systemd 서비스 확인:

```
sudo systemctl status gamelift-telemetry-collector
```

수집기 로그 검토:

```
sudo journalctl -u gamelift-telemetry-collector
```

- 수집기 구성 확인.

B. IAM 권한 확인

- 인스턴스 역할 권한 확인.
- 필요한 정책 확인.
 - `aps:RemoteWrite`
 - `cloudwatch:PutMetricData`
- 역할 신뢰 관계 검증.

C. 네트워크 연결

- 엔드포인트 액세스 확인.
- 보안 그룹 규칙 확인.
- 네트워크 ACL 검토.

인증 오류

증상

- SigV4 인증 실패.
- 액세스 거부됨 메시지 수신.
- 자격 증명 새로 고침 문제.

해결 단계

A. SigV4 인증

- 임시 자격 증명 확인.
- 자격 증명 교체 확인.
- 인스턴스 프로파일 검증.

B. AMP 액세스

- 워크스페이스 구성 검토.
- 원격 쓰기 URL 확인.
- IAM 역할 바인딩 확인.

대시보드 문제

증상

- 비어 있는 대시보드.
- 누락된 데이터 포인트.
- 인증 실패.

해결 단계

A. 데이터 소스 구성

- Prometheus 연결 확인.
- Amazon CloudWatch 통합 확인.
- 데이터 소스 권한 테스트.

B. Grafana 액세스

- SSO 구성 확인.
- 필요한 경우 2FA 설정 확인.
- 사용자 권한 확인.

Windows 관련 문제

증상

- 서비스 시작에 실패.
- 지표 수집 격차.
- 권한 오류.

해결 단계

- Windows 서비스 상태 확인.
- Windows 이벤트 로그 확인.
- 수집기 구성 검토.
- Windows별 경로 검증.

를 사용하여 Amazon GameLift Servers API 호출 로깅 AWS CloudTrail

Amazon GameLift Servers는에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다Amazon GameLift Servers. CloudTrail은 Amazon GameLift Servers에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 직접 호출에는 Amazon GameLift Servers 콘솔로부터의 직접 호출과 Amazon GameLift Servers API 작업에 대한 코드 직접 호출이 포함됩니다. 추적을 생성하면 Amazon GameLift Servers 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon GameLift Servers에 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 설명서](#)를 참조하세요.

CloudTrail의 Amazon GameLift Servers 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. 에서 활동이 발생하면 Amazon GameLift Servers해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 내용은 [CloudTrail 이벤트 기록을 사용하여 이벤트 보기](#)를 참조하세요.

Amazon GameLift Servers에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 AWS 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 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 리전에서 CloudTrail 로그 파일 받기](#) 및 [여러 계정에서 CloudTrail 로그 파일 받기](#)

모든 Amazon GameLift Servers 작업은 CloudTrail에서 로깅되고 Amazon GameLift Servers API 참조<https://docs.aws.amazon.com/gamelift/latest/apireference/>에 기록됩니다. 예를 들어

CreateGameSession, CreatePlayerSession, UpdateGameSession 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 내용은 [CloudTrail userIdentity 요소](#)를 참조하세요.

Amazon GameLift Servers 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 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다.

CloudTrail 로그 파일은 퍼블릭 API 직접 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 GameLift Servers의 서버 메시지 로깅

Amazon GameLift Servers 서버의 사용자 지정 서버 메시지를 로그 파일로 캡처할 수 있습니다. 로깅을 구성하는 방법은 사용자 지정 서버 또는 Amazon GameLift Servers Realtime을 사용하는지 여부, 그리고 관리형 EC2 플릿을 사용하는지 또는 컨테이너 플릿을 사용하는지 여부에 따라 달라집니다(이 장의 해당 하위 섹션들 참조).

주제

- [로깅 차이: 관리형 EC2 플릿 대 컨테이너 플릿 비교](#)
- [서버 메시지 로깅\(사용자 지정 서버\)](#)

로깅 차이: 관리형 EC2 플릿 대 컨테이너 플릿 비교

로깅 동작은 관리형 EC2 플릿과 컨테이너 플릿 간에 크게 다릅니다. 이러한 차이를 이해하는 것은 EC2에서 컨테이너로 마이그레이션하거나 로깅 요구 사항에 적합한 플릿 유형을 선택할 때 매우 중요합니다.

주요 차이점 개요

기능	관리형 EC2 플릿	컨테이너 플릿
로그 스토리지	게임 세션이 종료된 후 Amazon S3에 자동으로 업로드됨	Amazon CloudWatch Logs 또는 사용자 지정 로깅 솔루션을 구성해야 함
GetGameSessionLogUrl API	사용 가능 - 로그 다운로드를 위해 Amazon S3 URL 반환	사용할 수 없음 - Amazon S3에 로그가 자동으로 저장되지 않음
로그 보존	Amazon S3에서 14일(자동)	로깅 구성에 따라 다름
실시간 모니터링	제한 - 게임 세션이 종료된 후에만 로그 사용 가능	Amazon CloudWatch Logs 통합으로 사용 가능
설정 복잡성	자동 - 추가 구성 필요 없음	명시적 로깅 구성 필요

관리형 EC2 플릿 로깅

관리형 EC2 플릿의 경우 Amazon GameLift Servers는 자동 로그 관리를 제공합니다.

- 자동 업로드: 게임 세션이 종료되면 서버 로그가 Amazon S3에 자동 업로드
- GetGameSessionLogUrl API: 이 API를 사용하여 Amazon S3에서 로그를 다운로드하기 위해 미리 서명된 URL 검색
- 14일 보존: 로그는 자동 삭제 전 14일 동안 Amazon S3에 보존
- 크기 제한: 로그 파일에는 게임 세션당 크기 제한이 있음([Amazon GameLift Servers 엔드포인트 및 할당량 참조](#))

자세한 내용은 [the section called “사용자 지정 서버 로깅”](#) 및 [GetGameSessionLogUrl API 참조](#)를 참조하세요.

컨테이너 플릿 로깅

컨테이너 플릿에서는 로깅을 명시적으로 구성해야 합니다.

- 자동 Amazon S3 업로드 없음: 로그가 Amazon S3에 자동으로 업로드되지 않음
- GetGameSessionLogUrl 사용할 수 없음: 이 API는 컨테이너 플릿에서 작동하지 않음
- Amazon CloudWatch Logs 통합: 중앙 집중식 로깅을 위해 Amazon CloudWatch Logs로 로그를 전송하도록 컨테이너 구성
- 사용자 지정 로깅 솔루션: 로그 드라이버 또는 사이드카 컨테이너를 사용하여 자체 로깅 인프라 구현
- 실시간 액세스: 적절한 구성을 사용하면 게임 세션 중에 로그에 실시간으로 액세스할 수 있음

자세한 컨테이너 로깅 옵션은 [컨테이너 플릿 작동 방식](#)을 참조하세요.

마이그레이션 고려 사항

관리형 EC2에서 컨테이너 플릿으로 마이그레이션할 때는 다음 로깅 변경 사항을 고려하세요.

- 로그 검색 코드 업데이트: GetGameSessionLogUrl API 호출을 Amazon CloudWatch Logs 쿼리 또는 사용자 지정 로깅 솔루션으로 대체
- 로그 보존 구성: Amazon CloudWatch Logs 또는 로깅 시스템에서 적절한 보존 정책 설정
- 실시간 모니터링 구현: 실시간 로그 액세스를 활용하여 관찰성 향상
- 로그 볼륨 및 비용 검토: Amazon CloudWatch Logs 요금은 관리형 EC2 플릿에 포함된 Amazon S3 스토리지와 다름Use this API to retrieve a pre-signed URL for downloading logs from Amazon S3

권장 컨테이너 로깅 설정

컨테이너 플릿의 경우 다음을 권장합니다.

1. Amazon CloudWatch Logs 통합: awslogs 로그 드라이버를 사용하도록 컨테이너 정의 구성
2. 구조화된 로깅: 더 나은 검색 및 분석을 위해 구조화된 로그 형식(JSON) 사용
3. 로그 수준: 세부 정보 및 비용을 제어하는 적절한 로그 수준 구현
4. 보존 정책: 규정 준수 및 디버깅 요구 사항에 따라 보존 기간 설정

서버 메시지 로깅(사용자 지정 서버)

Amazon GameLift Servers 서버의 사용자 지정 서버 메시지를 로그 파일로 캡처할 수 있습니다.

⚠ Important

게임 세션당 로그 파일 크기에는 제한이 있습니다(AWS 일반 참조에서 [Amazon GameLift Servers 엔드포인트 및 할당량](#) 참조). 게임 세션이 종료되면 Amazon GameLift Servers는 서버 로그를 Amazon Simple Storage Service(Amazon S3)에 업로드합니다. Amazon GameLift Servers는 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 Servers 사용자 지정 서버를 사용하면 서버 프로세스 구성의 일부로 구성하는 로깅을 수행할 자체 코드를 작성합니다. Amazon GameLift Servers는 로깅 구성을 사용하여 각 게임 세션이 끝날 때 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 [ProcessParameters](#) 객체의 [LogParameters](#)로 제공합니다.

```
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 [ProcessParameters](#) 객체를 제공합니다.

```
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 [ProcessParameters](#) 객체의 [LogParameters](#)로 제공합니다.

```
var processReadyParameter = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(logPaths));
```

3. [ProcessReady\(\)](#)를 호출할 때 [ProcessParameters](#) 객체를 제공합니다.

```
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

로그 액세스는 플릿 유형에 따라 다릅니다.

관리형 EC2 플릿

게임 세션이 종료되면 Amazon GameLift Servers가 자동으로 로그를 S3 버킷에 저장하고 14일간 보관합니다. 게임 세션에 대한 로그의 위치를 가져오려면 [GetGameSessionLogUrl](#) API 작업을 사용할 수 있습니다. 로그를 다운로드하려면 작업에서 반환하는 URL을 사용합니다.

또는 원하는 로깅 서비스 또는 스토리지 위치로 직접 로그를 전송하도록 게임 서버를 구성하여 자체 로깅 솔루션을 설정할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

컨테이너 플릿

컨테이너 플릿은 모든 컨테이너에서 표준 출력 및 오류 스트림을 캡처합니다. 다음 옵션 중 하나를 사용하여 로깅을 구성할 수 있습니다.

- 컨테이너 출력을 지정된 로그 그룹의 CloudWatch 로그 스트림으로 저장
- S3 스토리지 버킷에 컨테이너 출력 저장
- 로깅 끄기(컨테이너 출력이 저장되지 않음)

컨테이너 플릿을 생성할 때 로깅 옵션을 구성하는 방법에 대한 자세한 내용은 Amazon GameLift Servers API 참조의 [LogConfiguration](#)을 참조하세요.

Amazon GameLift Servers의 보안

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 빌드된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 사용자 간의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS는 클라우드에서 AWS AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 규정 [AWS 준수 프로그램](#) 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 [AWS 프로그램 제공 범위 내 서비스](#) 규정 준수 프로그램 제공 범위 내 서비스를 Amazon GameLift Servers참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 회사의 요구 사항, 관련AWS 및 규정을 비롯한 기타 요인에 대해서도 책임이 있습니다.

이 설명서는 Amazon GameLift Servers 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Amazon GameLift Servers를 구성하는 방법을 보여줍니다. 또한 Amazon GameLift Servers 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- [Amazon GameLift Servers의 데이터 보호](#)
- [Amazon GameLift Servers의 자격 증명 및 액세스 관리](#)
- [Amazon GameLift Servers를 사용하여 로깅 및 모니터링](#)
- [의 규정 준수 검증Amazon GameLift Servers](#)
- [Amazon GameLift Servers의 복원성](#)
- [Amazon GameLift Servers에서 인프라 보안](#)
- [Amazon GameLift Servers의 구성 및 취약성 분석](#)
- [Amazon GameLift Servers의 보안 모범 사례](#)

Amazon GameLift Servers의 데이터 보호

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

AWS [공동 책임 모델](#)의 데이터 보호에 적용됩니다 Amazon GameLift Servers. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 [데이터 프라이버시 FAQ](#)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 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 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 [CloudTrail 추적 작업을](#) 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [연방 정보 처리 표준\(FIPS\) 140-3](#)을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon GameLift Servers 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 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 자격 증명을 URL에 포함해서는 안 됩니다.

Amazon GameLift Servers 고유 데이터는 다음과 같이 처리됩니다.

- Amazon GameLift Servers에 업로드하는 게임 서버 빌드 및 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. 스크립트 및 빌드를 삭제하려면 Amazon GameLift Servers 콘솔 또는 서비스 API를 사용합니다.
- 게임 세션 로그 데이터는 게임 세션이 완료된 후 일정 기간 동안 Amazon S3에 저장됩니다. 권한 있는 사용자는 Amazon GameLift Servers 콘솔의 링크를 통해 다운로드하거나 서비스 API를 호출하여 로그 데이터에 액세스할 수 있습니다.
- 지표 및 이벤트 데이터는 Amazon GameLift Servers에 저장되며 Amazon GameLift Servers 콘솔을 통해 또는 서비스 API를 호출하여 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 Amazon GameLift Servers에 저장됩니다. 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 Servers 고유 데이터의 유희 데이터 암호화는 다음과 같이 처리됩니다.

- 게임 서버 빌드 및 스크립트는 서버 측 암호화를 사용하여 Amazon S3 버킷에 저장됩니다.
- 고객이 제공한 데이터는 암호화된 형식으로 Amazon GameLift Servers에 저장됩니다.

전송 중 암호화

Amazon GameLift Servers APIs에 대한 연결은 보안(SSL) 연결을 통해 이루어지며 [AWS 서명 버전 4](#)를 사용하여 인증됩니다(AWS CLI 또는 AWS SDK를 통해 연결할 경우 서명이 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 Servers 리소스에서 호스팅되는 사용자 지정 게임 서버의 경우 통신에 Amazon GameLift Servers 서비스가 관련되지 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 지원 플릿에 대한 인증서는 플릿과 동시에 생성되며 만료 날짜는 플릿 생성 날짜를 기준으로 합니다.
- TLS 인증서 생성이 활성화된 Amazon GameLift Servers Realtime의 경우 Realtime용 클라이언트 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.

Note

TLS 인증서에 대한 인증 기관 요구 사항을 충족하기 위해 Amazon GameLift Servers는 인증서를 생성하도록 구성된 플릿의 최대 인증서 수명을 조정합니다. 인증서 수명은 플릿 생성 시 시작되며 다음 일정에 따라 변경됩니다.

- 2026년 3월 11일까지 발급된 TLS 인증서의 최대 수명은 398일입니다.
- 2026년 3월 1일부터 발급된 TLS 인증서의 최대 수명은 200일입니다.
- 2027년 3월 1일부터 발급된 TLS 인증서의 최대 수명은 100일입니다.
- 2029년 3월 1일부터 발급된 TLS 인증서의 최대 수명은 47일입니다.

인증서를 갱신하고 up-to-date 게임 서버 런타임 환경을 유지하기 위해에서는 게임 서버 플릿을 정기적으로 교체할 것을 Amazon GameLift Servers 권장합니다.

인터넷워크 트래픽 개인 정보

Amazon GameLift Servers 인스턴스에 안전하게 원격으로 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. 를 사용하면 AWS Systems Manager Session Manager 및 Run Command를 사용하여 인스턴스에 대한 Amazon GameLift Servers FleetIQ원격 액세스가 TLS 1.2를 사용하여 암호화되고 연결 생성 요청은 SigV4를 사용하여 서명됩니다. 관리형 Amazon GameLift Servers 인스턴스 연결에 대한 도움말은 [플릿 인스턴스에 연결](#) 단원을 참조하십시오.

Amazon GameLift Servers의 I자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 Amazon GameLift Servers 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- [대상](#)
- [ID를 통한 인증](#)
- [정책을 사용하여 액세스 관리](#)
- [Amazon GameLift Servers에서 IAM을 사용하는 방식](#)
- [Amazon GameLift Servers에 대한 ID 기반 정책 예시](#)
- [Amazon GameLift Servers ID 및 액세스 문제 해결](#)
- [AWS에 대한 관리형 정책 Amazon GameLift Servers](#)

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청([참조 Amazon GameLift Servers ID 및 액세스 문제 해결](#))
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출([Amazon GameLift Servers에서 IAM을 사용하는 방식](#) 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성([Amazon GameLift Servers에 대한 ID 기반 정책 예시](#) 참조)

ID를 통한 인증

인증은 AWS 자격 증명으로 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다.

로그인하는 방법에 대한 자세한 내용은 AWS Sign-In 사용 설명서의 [AWS 계정에 로그인하는 방법](#) 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해서는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 [API 요청용 AWS Signature Version 4](#) 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 [루트 사용자 자격 증명이 필요한 작업](#) 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 [What is IAM Identity Center?](#)를 참조하세요.

IAM 사용자 및 그룹

[IAM 사용자](#)는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명에 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 [자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여 액세스하도록 인간 사용자에게 요구](#)하기를 참조하세요.

[IAM 그룹](#)은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 [IAM 사용자 사용 사례](#) 섹션을 참조하세요.

IAM 역할

[IAM 역할](#)은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. [사용자에서 IAM 역할\(콘솔\)로 전환하거나 또는 API 작업을 호출하여 역할을](#) 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 [역할 수임 방법](#)을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 [교차 계정 리소스 액세스](#)를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다. 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 [JSON 정책 개요](#) 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 [고객 관리형 정책으로 사용자 지정 IAM 권한 정의](#)를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 [관리형 정책 및 인라인 정책 중에서 선택](#) 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 [보안 주체를 지정](#)해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 [IAM 엔터티의 권한 범위](#)를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 [서비스 제어 정책](#)을 참조하세요.
- 리소스 제어 정책(RCP) - 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 [리소스 제어 정책\(RCP\)](#)을 참조하세요.
- 세션 정책 - 역할 또는 페더레이션 사용자에게 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 [세션 정책](#)을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하세요.

Amazon GameLift Servers에서 IAM을 사용하는 방식

IAM을 사용하여 Amazon GameLift Servers에 대한 액세스를 관리하기 전에 Amazon GameLift Servers와 함께 사용할 수 있는 IAM 기능을 알아보세요.

Amazon GameLift Servers를 통해 사용할 수 있는 IAM 기능

IAM 기능	Amazon GameLift Servers 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예

IAM 기능	Amazon GameLift Servers 지원
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	예
서비스 연결 역할	아니요

Amazon GameLift Servers 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 [AWS IAM으로 작업하는 서비스를](#) 참조하세요.

Amazon GameLift Servers에 대한 ID 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 [고객 관리형 정책으로 사용자 지정 IAM 권한 정의](#)를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 [IAM JSON 정책 요소 참조](#)를 참조하세요.

Amazon GameLift Servers 자격 증명 기반 정책 예시

Amazon GameLift Servers ID 기반 정책의 예를 보려면 [Amazon GameLift Servers에 대한 ID 기반 정책 예시](#)를 참조하세요.

Amazon GameLift Servers 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자

는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 [보안 주체를 지정](#)해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 이 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [IAM에서 교차 계정 리소스 액세스](#)를 참조하세요.

Amazon GameLift Servers 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Amazon GameLift Servers 작업 목록의 경우 서비스 권한 부여 참조의 [Amazon GameLift Servers에서 정의한 작업](#)을 참조하세요.

Amazon GameLift Servers의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
gamelift
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
    "gamelift:action1",
    "gamelift:action2"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "gamelift:Describe*"
```

Amazon GameLift Servers ID 기반 정책의 예를 보려면 [Amazon GameLift Servers에 대한 ID 기반 정책 예시](#)를 참조하세요.

Amazon GameLift Servers 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 [Amazon 리소스 이름\(ARN\)](#)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*" 
```

Amazon GameLift Servers 리소스 유형 및 해당 ARN의 목록의 경우 서비스 권한 부여 참조의 [Amazon GameLift Servers에서 정의한 리소스](#)를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 [Amazon GameLift Servers에서 정의한 작업](#)을 참조하세요.

일부 Amazon GameLift Servers 리소스에는 ARN 값이 있어 IAM 정책을 사용하여 리소스에 대한 액세스를 관리할 수 있습니다. Amazon GameLift Servers 플릿 리소스에는 다음 구문을 사용하는 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 Servers 자격 증명 기반 정책의 예를 보려면 [Amazon GameLift Servers에 대한 ID 기반 정책 예시](#) 섹션을 참조하세요.

Amazon GameLift Servers 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 [조건 연산자](#)를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 [AWS 전역 조건 컨텍스트 키](#)를 참조하세요.

Amazon GameLift Servers 조건 키 목록의 경우 서비스 권한 부여 참조의 [Amazon GameLift Servers를 위한 조건 키](#)를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 [Amazon GameLift Servers가 정의한 작업](#) 섹션을 참조하세요.

Amazon GameLift Servers 자격 증명 기반 정책의 예를 보려면 [Amazon GameLift Servers에 대한 ID 기반 정책 예시](#) 섹션을 참조하세요.

Amazon GameLift Servers의 ACL

ACL 지원: 아니요

액세스 제어 목록(ACL)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.

Amazon GameLift Servers를 사용한 ABAC

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 [조건 요소](#)에 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 [ABAC 권한 부여를 통한 권한 정의](#)를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 [속성 기반 액세스 제어\(ABAC\) 사용](#)을 참조하세요.

해당 리소스 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예는 [태그를 기준으로 Amazon GameLift Servers 플릿 보기](#) 섹션을 참조하세요.

Amazon GameLift Servers에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 [IAM의 임시 보안 자격 증명](#) 및 [IAM으로 작업하는 AWS 서비스](#) 섹션을 참조하세요.

Amazon GameLift Servers의 서비스 간 엔터티 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스 함께 사용합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.

Amazon GameLift Servers의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [AWS 서비스 AWS에 권한을 위임할 역할 생성](#)을 참조하세요.

Warning

서비스 역할에 대한 권한을 변경하면 Amazon GameLift Servers 기능이 중단될 수 있습니다. Amazon GameLift Servers이 그 일을 하라는 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon GameLift Servers 호스팅 게임 서버가 AWS Lambda 함수 또는 Amazon DynamoDB 데이터베이스와 같은 다른 AWS 리소스에 액세스하도록 허용합니다. 게임 서버는 Amazon GameLift Servers 관리하는 플릿에서 호스팅되므로 다른 AWS 리소스에 Amazon GameLift Servers 대한 제한된 액세스를 제공하는 서비스 역할이 필요합니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

Amazon GameLift Servers에 대한 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 연결 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.

Amazon GameLift Servers에 대한 ID 기반 정책 예시

기본적으로 사용자 및 역할에는 Amazon GameLift Servers 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [IAM 정책 생성\(콘솔\)](#)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 Amazon GameLift Servers에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조에서 [Amazon GameLift Servers에 대한 작업, 리소스 및 조건 키](#)를 참조하세요.

주제

- [정책 모범 사례](#)
- [Amazon GameLift Servers 콘솔 사용](#)
- [사용자가 자신의 고유한 권한을 볼 수 있도록 허용](#)
- [게임 세션을 위한 플레이어 액세스 허용](#)
- [하나의 Amazon GameLift Servers 대기열에 대한 액세스 허용](#)
- [태그를 기준으로 Amazon GameLift Servers 플릿 보기](#)
- [Amazon S3의 게임 빌드 파일에 액세스](#)

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 Amazon GameLift Servers 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 [AWS 관리형 정책](#) 또는 [AWS 직무에 대한 관리형 정책](#)을 참조하세요.
- 최소 권한 적용 - IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 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 권한](#)을 참조하세요.
- 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 작성할 수 있습니다. AWS 서비스와 같은 특정을 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 [IAM JSON 정책 요소: 조건](#)을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 [IAM Access Analyzer에서 정책 검증](#)을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정입니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 [MFA를 통한 보안 API 액세스](#)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 [IAM의 보안 모범 사례](#)를 참조하세요.

Amazon GameLift Servers 콘솔 사용

Amazon GameLift Servers 콘솔에 액세스하려면 최소 권한 세트가 있어야 합니다. 이러한 권한은의 Amazon GameLift Servers 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

이러한 엔터티가 Amazon GameLift Servers 콘솔을 계속 사용할 수 있도록 하려면 다음 예제 및 [관리 권한 예제](#)의 구문을 사용하여 사용자 및 그룹에 권한을 추가합니다. 자세한 내용은 [Amazon GameLift Servers의 사용자 권한 설정](#) 단원을 참조하십시오.

AWS CLI 또는 AWS API 작업을 Amazon GameLift Servers 통해 로 작업하는 사용자는 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.

Amazon GameLift Servers 콘솔 기능을 사용하는 데 필요한 권한에 대한 자세한 내용은 [관리 권한 예제](#)의 관리자에 대한 권한 구문을 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 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.

하나의 Amazon GameLift Servers 대기열에 대한 액세스 허용

다음 예제는 사용자에게 특정 Amazon GameLift Servers 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.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewSpecificQueueInfo",
      "Effect": "Allow",
      "Action": [
        "gamelift:DescribeGameSessionQueues"
      ],
      "Resource": "arn:aws:gamelift:us-east-1:555555555555:gamesessionqueue/examplequeue123"
    },
    {
      "Sid": "ManageSpecificQueue",
      "Effect": "Allow",
      "Action": [

```

```

        "gamelift:UpdateGameSessionQueue",
        "gamelift>DeleteGameSessionQueue"
    ],
    "Resource": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/
examplequeue123"
    }
]
}

```

태그를 기준으로 Amazon GameLift Servers 플릿 보기

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 Amazon GameLift Servers 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.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListFleetsInConsole",
      "Effect": "Allow",
      "Action": "gamelift:ListFleets",
      "Resource": "*"
    },
    {
      "Sid": "ViewFleetIfOwner",
      "Effect": "Allow",
      "Action": "gamelift:DescribeFleetAttributes",
      "Resource": "arn:aws:gamelift:*:*:fleet/*",
      "Condition": {
        "StringEquals": {"aws:ResourceAccount": "${aws:username}"}
      }
    }
  ]
}

```

Amazon S3의 게임 빌드 파일에 액세스

게임 서버를 Amazon GameLift Servers와 통합한 후 빌드 파일을 Amazon S3에 업로드합니다. Amazon GameLift Servers가 빌드 파일에 액세스할 수 있도록 하려면 다음 정책을 사용합니다.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::bucket-name/object-name"
    }
  ]
}
```

Amazon GameLift Servers 게임 파일 업로드에 대한 자세한 내용은 [Amazon GameLift Servers의 게임 서버 빌드 생성](#) 섹션을 참조하세요.

Amazon GameLift Servers ID 및 액세스 문제 해결

다음 정보를 사용하여 Amazon GameLift Servers 및 AWS Identity and Access Management (IAM)로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- [Amazon GameLift Servers에서 작업을 수행할 권한이 없음](#)
- [iam:PassRole을 수행하도록 인증되지 않음](#)
- [내 외부의 사람이 내 Amazon GameLift Servers 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.](#)

Amazon GameLift Servers에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 AWS 계정 관리자에게 문의하여 지원을 받으세요. 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.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon GameLift Servers에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 Amazon GameLift Servers에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
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 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Amazon GameLift Servers 리소스에 액세스 AWS 계정 하도록 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 제

어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon GameLift Servers에서 이러한 기능을 지원하는지 여부를 알아보려면 [Amazon GameLift Servers에서 IAM을 사용하는 방식](#)을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 [IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요](#).
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [타사가 AWS 계정 소유한에 대한 액세스 권한 제공을](#) AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [외부에서 인증된 사용자에게 액세스 권한 제공\(ID 페더레이션\)](#)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 [IAM의 크로스 계정 리소스 액세스](#)를 참조하세요.

AWS 에 대한 관리형 정책 Amazon GameLift Servers

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 [고객 관리형 정책](#)을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 줍니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 [AWS 관리형 정책](#)을 참조하세요.

AWS 관리형 정책: GameLiftContainerFleetPolicy

GameLiftContainerFleetPolicy를 IAM 역할에 연결할 수 있습니다.

이 정책은 Amazon GameLift Servers 컨테이너 플릿의 컴퓨팅 작업에 대한 권한을 부여합니다. 컨테이너 플릿은가 자동으로 Amazon GameLift Servers 관리하는 호스팅 리소스 세트입니다.는 사용자를

대신하여 Amazon GameLift Servers 서비스 및 기타 AWS 서비스에 연결할 수 있는 권한이 Amazon GameLift Servers 필요합니다.

Amazon GameLift Servers를 사용하여 컨테이너 플릿을 생성할 때 GameLiftContainerFleetPolicy 관리형 정책이 연결된 IAM 서비스 역할을 제공합니다. 이 서비스 역할을 생성하는 방법에 대한 지침은 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#) 섹션을 참조하세요.

자세한 내용은 [GameLiftContainerFleetPolicy](#)를 참조하세요.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `cloudwatch` - Amazon GameLift Servers가 AWS 계정의 Amazon CloudWatch Events 로그 스트림에 게임 세션 로그를 쓸 수 있도록 허용합니다.
- `cloudwatch` - Amazon GameLift Servers가 CloudWatch 로그 그룹을 생성하여 로그 스트림에서 게임 세션 데이터를 구성할 수 있도록 허용합니다.
- `s3` - Amazon GameLift Servers가 AWS 계정의 Amazon Simple Storage Service 버킷에 게임 세션 로그를 쓸 수 있도록 허용합니다.
- `s3` - API 작업을 사용하여 지정된 Amazon S3 버킷 AWS 리전 이 있는 Amazon GameLift Servers 검색할 수 있습니다 `s3:GetBucketLocation`.
- `gamelift` - Amazon GameLift Servers가 호스팅 게임 서버가 AWS 계정을 통해 Amazon GameLift Servers 서비스와 통신할 수 있도록 하는 인증 토큰을 검색할 수 있도록 허용합니다.

Amazon GameLift Servers AWS 관리형 정책에 대한 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 Amazon GameLift Servers 이후부터의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 [Amazon GameLift Servers 릴리스 정보 페이지](#)의 RSS 피드를 구독하세요.

변경	설명	Date
GameLiftContainerFleetPolicy - 변경	Amazon GameLift Servers는 Amazon S3 버킷 AWS 리전 의 를 검색할 수 있는 새 권한을 추가했습니다.	2024년 2월 5일

변경	설명	Date
GameLiftContainerFleetPolicy – 새 정책	Amazon GameLift Servers는 Amazon GameLift Servers 관리형 플릿에서 게임 서버 컨테이너를 실행할 수 있도록 새로운 권한을 추가했습니다.	2024년 11월 12일
Amazon GameLift Servers에서 변경 사항 추적 시작	Amazon GameLift Servers가 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2024년 11월 12일

Amazon GameLift Servers를 사용하여 로깅 및 모니터링

모니터링은 Amazon GameLift Servers와 사용자 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 중요한 역할을 합니다. 발생하는 다중 지점 실패를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.

AWS 및 Amazon GameLift Servers는 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 지정한 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책으로 알림이 전송됩니다. CloudWatch 경보는 특정 상태가 아니라 상태 변경에 의해 트리거되고 지정한 수의 기간 동안 유지됩니다. 자세한 내용은 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#) 섹션을 참조하세요.

AWS CloudTrail 로그

CloudTrail은 AWS에서 사용자, 역할 또는 Amazon GameLift Servers 서비스가 수행한 작업 기록을 제공합니다. CloudTrail에서 수집한 정보를 사용하여 Amazon GameLift Servers에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 [를 사용하여 Amazon GameLift Servers API 호출 로깅 AWS CloudTrail](#) 섹션을 참조하세요.

의 규정 준수 검증Amazon GameLift Servers

Amazon GameLift Servers는 AWS 규정 준수 프로그램의 범위에 속하지 않습니다.

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 [규정 준수 프로그램 제공 범위 내 AWS 서비스](#)를 참조하고 관심 있는 규정 준수 프로그램을 선택하십시오. 일반적인 정보는 [AWS 규정 준수 프로그램](#)을 참조하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 [AWS Artifact에서 보고서 다운로드](#)를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. AWS 서비스 사용 시 규정 준수 책임에 대한 자세한 내용은 [AWS 보안 설명서](#)를 참조하세요.

Amazon GameLift Servers의 복원성

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

AWS 글로벌 인프라는 AWS리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 리전을 제공하며 이러한 가용 리전은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 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 리전 및 가용 영역에 대한 자세한 내용은 [AWS 글로벌 인프라](#)를 참조하세요.

AWS 글로벌 인프라뿐만 아니라 Amazon GameLift Servers도 데이터 복원성 요구 사항을 지원하기 위해 다음과 같은 기능을 제공합니다.

- 다중 리전 대기열 - Amazon GameLift Servers 게임 세션 대기열은 사용 가능한 호스팅 리소스로 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 사용자 지정](#) 섹션을 참조하세요.
- 자동 용량 조정 - Amazon GameLift Servers 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 Servers를 사용하여 게임 호스팅 용량 확장](#) 섹션을 참조하세요.
- 인스턴스 간 분산 - Amazon GameLift Servers는 플릿 크기에 따라 수신 트래픽을 여러 인스턴스로 분산합니다. 인스턴스가 비정상이거나 응답하지 않는 경우 프로덕션 게임의 가용성을 유지하기 위해 인스턴스가 여러 개 있어야 합니다.

- Amazon S3 스토리지 - 게임 서버 빌드 및 스크립트는 Amazon GameLift Servers에 업로드되고 여러 데이터 센터 복제를 사용하여 복원성을 높이는 스탠다드 스토리지 클래스를 사용하여 Amazon S3에 저장됩니다. 게임 세션 로그도 스탠다드 스토리지 클래스를 사용하여 Amazon S3에 저장됩니다.

Amazon GameLift Servers에서 인프라 보안

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

관리형 서비스인 Amazon GameLift Servers은 [Amazon Web Services: 보안 프로세스 개요](#) 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 Amazon GameLift Servers에 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 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 [AWS Security Token Service](#)(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이 Amazon GameLift Servers 서비스는 각 플릿이 AWS 클라우드에서 논리적으로 격리된 영역에 존재하도록 모든 플릿을 Amazon Virtual Private Cloud(VPC)에 배치합니다. 또한 Amazon GameLift Servers 정책을 사용하여 특정 VPC 엔드포인트 또는 특정 VPC로부터의 액세스를 제어할 수 있습니다. 그러면 AWS 네트워크 내에 있는 특정 VPC에서만 특정 Amazon GameLift Servers 리소스에 대한 네트워크 액세스가 효과적으로 격리됩니다. 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.

Amazon GameLift Servers의 구성 및 취약성 분석

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

구성 및 IT 컨트롤은 AWS와 고객 간의 공동 책임입니다. 자세한 내용은 AWS [공동 책임 모델](#)을 참조하세요. AWS는 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 사례에서도 Amazon GameLift Servers의 구성 및 취약성 분석을 다룹니다.

- 고객은 게임 호스팅을 위해 Amazon GameLift Servers 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 Amazon GameLift Servers에 업로드하고, 새 컨테이너 이미지에 이를 제공하며, 새 빌드에 대한 새 플릿을 만들고, 트래픽을 새 플릿으로 리디렉션합니다.
 - 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 대체합니다. 최소한 1년에 한 번은 프로덕션 플릿을 교체해야 합니다. 자세한 내용은 [플릿 런타임 환경을 최신 상태로 유지](#) 섹션을 참조하세요.
- 고객은 AWS SDK, Amazon GameLift Servers 서버 SDK 및 Realtime 서버용 Amazon GameLift Servers 클라이언트 SDK를 비롯한 최신 SDK 버전으로 게임을 정기적으로 업데이트하는 것을 고려해야 합니다.

Amazon GameLift Servers의 보안 모범 사례

Amazon EC2의 독립형 기능으로 Amazon GameLift Servers FleetIQ를 사용하는 경우 Amazon EC2 사용 설명서에서 [Amazon EC2의 보안](#)을 참조하세요.

Amazon GameLift Servers는 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 사용하세요.

플릿 런타임 환경을 최신 상태로 유지

Amazon GameLift Servers에서는 관리형 플릿(관리형 EC2 및 관리형 컨테이너 플릿 포함)을 정기적으로 교체하여 게임 서버의 안전한 런타임 환경을 유지할 것을 적극 권장합니다. 런타임 업데이트 없이 장기간 실행되는 플릿에는 오래된 종속성과 게임 서버를 손상시킬 수 있는 보안 취약성이 포함될 수 있습니다. Amazon GameLift Servers 플릿에 배포된 소프트웨어에 대한 책임을 공유하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers의 구성 및 취약성 분석](#) 섹션을 참조하세요.

관리형 플릿의 런타임 환경은 Amazon Machine Image(AMI) 버전에 따라 결정됩니다. 새 플릿이 생성되면 Amazon GameLift Servers가 플릿에 사용 가능한 최신 AMI 버전을 할당하고 해당 플릿의 모든 컴퓨팅 인스턴스가 해당 버전과 함께 배포됩니다. AMI 버전을 업데이트하려면 새 플릿을 생성해야 합니다. 현재 AMI 버전에 대한 자세한 내용은 [Amazon GameLift Servers AMI 버전](#) 섹션을 참조하세요.

권장 모범 사례:

- 플릿 수명 모니터링 및 30일 이상 된 플릿 교체 - Amazon GameLift Servers 콘솔에서 플릿 생성 날짜를 추적하거나 CLI를 사용하여 플릿 속성을 검색할 수 있습니다. Amazon GameLift Servers는 90일 이상 된 플릿에 대해 콘솔에 경고를 표시하고 1년 이상 된 플릿에 대해 이메일로 계정 소유자에게 알립니다.

Note

플릿을 업데이트해도(예: [UpdateFleetAttributes](#) 또는 [UpdateContainerFleet](#) 사용) AMI 버전은 변경되지 않습니다. 새 플릿을 생성해야 합니다.

- 보안 상태에 따라 정기적으로 플릿 교체 - 새 플릿을 생성하고 이전 플릿을 사용 중지하도록 정기적인 일정을 설정합니다. [Amazon Q](#)와 같은 서비스를 사용하여 현재 AMI 버전으로 게임 코드를 검토하고, 보안 문제를 감지하며, 문제 해결 단계를 제안하는 것이 좋습니다.
- 배포 전에 최신 AMI 버전으로 서버 빌드 테스트 - 새 플릿을 생성하기 전에 서버 빌드를 수정하고 Amazon GameLift Servers에 이를 업로드해야 할 수 있습니다.
- AWS 계정의 플릿 할당량 관리 - 대체 플릿을 생성하는 데 필요한 경우 한도 증가를 요청할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 엔드포인트 및 할당량](#) 단원을 참조하십시오.
- 플릿 교체 자동화 고려 - 프로세스를 자동화하여 새 플릿을 생성하고 이전 플릿에서 플레이어 트래픽을 마이그레이션할 수 있습니다. 예제:
 - AWS CloudFormation 를 사용하여 플릿 생성 및 관리를 자동화합니다. 플릿 구성을 CloudFormation 템플릿으로 유지하고 이를 사용하여 리소스 스택을 시작합니다.
 - Amazon GameLift Servers 별칭 기능을 활용하여 특정 플릿 ID를 추상화합니다. 플릿 별칭을 사용하면 진행 중인 게임 세션을 중단하지 않고 플레이어 트래픽을 기존 플릿에서 새 플릿으로 쉽게 전환할 수 있습니다. 자세한 내용은 [별칭을 사용하여 Amazon GameLift Servers 플릿 지정 추상화](#)를 참조하세요.
 - 블루/그린 배포 전략을 사용하여 마이그레이션 위험을 줄이고 가동 중지 시간을 0으로 유지합니다. 두 개의 동일한 프로덕션 환경을 사용하면 프로덕션과 유사한 전체 테스트 환경을 활용하고, 마이그레이션 프로세스를 더 잘 제어하며, 즉각적인 롤백을 보장할 수 있습니다.

포트 구성 보호

보안 위험이 있으므로 인터넷에 포트를 열지 않는 것이 좋습니다. 예를 들어 다음 구성은 인터넷상의 모든 사용자가 인스턴스에 액세스할 수 있도록 허용하는 원격 데스크톱 포트를 엽니다.

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "0.0.0.0/0",
      "Protocol": "RDP",
      "ToPort": 3389
    }
  ]
}
```

대신 다음 예제와 같이 [UpdateFleetPortSettings](#)를 사용하여 특정 IP 주소 또는 주소 범위가 있는 포트를 엽니다.

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "54.186.139.221/32",
      "Protocol": "TCP",
      "ToPort": 3389
    }
  ]
}
```

추가 보안 리소스

Amazon GameLift Servers를 보다 안전하게 사용할 수 있는 방법에 대한 자세한 내용은 [AWS Well-Architected Tool 보안 원칙](#)을 참조하세요.

Amazon GameLift Servers 요금 및 비용 계획

게임 인프라 예산을 최적화하려면 Amazon GameLift Servers 요금 구조를 이해하고 게임 호스팅 솔루션에서 비용 효율적인 전략을 구현해야 합니다. 요금 시나리오 및 요금 도구 예제를 포함한 자세한 서비스 정보는 [Amazon GameLift Servers 요금](#)을 참조하세요.

신규 AWS 고객은 프리 티어에서 최대 12개월 동안 요금을 부과Amazon GameLift Servers하지 않고 사용할 수 있습니다. 요금이 부과되지 않도록 하려면 프리 티어 사용 한도 내에서 사용하세요.

요금 모델

Amazon GameLift Servers는 다양한 사용 시나리오와 비즈니스 요구 사항을 수용하도록 설계된 여러 요금 모델을 제공합니다. 모델마다 비용 기준이 다릅니다. 각 모델의 비용 요소를 조정하여 비용을 제어할 수 있습니다.

관리형 호스팅

관리형 EC2 및 관리형 컨테이너 플릿은 Amazon GameLift Servers가 관리하는 Amazon EC2 인스턴스를 사용합니다. 게임 호스팅을 설정할 때 사용할 Amazon EC2 인스턴스 유형을 선택합니다.

- 컴퓨팅 비용 - 게임 세션 호스팅을 위해 시간당 EC2 인스턴스 사용량에 대한 비용을 지불합니다. 인스턴스의 비용 요인은 다음과 같습니다.
 - AWS 리전
 - 인스턴스 패밀리, 유형, 크기
 - 온디맨드 인스턴스 대 스팟 인스턴스 사용
 - 운영 체제(관리형 컨테이너 플릿은 Linux에서만 실행됨)
- 데이터 전송 - 게임 클라이언트와 호스팅된 게임 서버 간의 트래픽에 대한 요금 지불

Anywhere 호스팅

Anywhere 호스팅은 사용자가 제공하고 관리하는 게임 호스팅 리소스에서 실행되므로 컴퓨팅 비용이 들지 않습니다. 대신 다음을 기반으로 비용이 책정됩니다.

- 게임 세션 - Anywhere 컴퓨팅에 배치된 게임 세션 수를 기준으로 지불
- 연결 시간(분) - 서버 프로세스 연결 시간(분)에 대해 지불

Amazon GameLift Servers용 요금 계산기를 사용하여 다양한 Amazon GameLift Servers 구성에 대한 비용을 추정합니다. 게임 호스팅 솔루션을 설계하거나 구성 변경 비용을 예측할 때 이 계산기를 사용합니다. [예상 Amazon GameLift Servers 요금 산출](#)을 참조하세요.

Amazon GameLift Servers 예상 요금 산출

를 사용하면 [에 대한 요금 견적을 생성할 Amazon GameLift Servers](#) AWS Pricing Calculator 수 있습니다. 계산기를 사용하기 AWS 위해 AWS 계정 또는에 대한 심층 지식이 필요하지 않습니다.

AWS Pricing Calculator 계산기는 서비스 비용에 영향을 미치는 결정을 안내하여 게임 프로젝트의 비용이 얼마나 될 Amazon GameLift Servers 수 있는지 알려줍니다. Amazon GameLift Servers를 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.

Amazon GameLift Servers 관리형 호스팅 예상 요금

이 옵션은 서버 인스턴스 사용량 및 데이터 전송 비용을 포함하여 Amazon GameLift Servers 관리형 서버에서 게임을 호스팅하는 데 드는 추정 비용을 제공합니다. Amazon GameLift Servers 관리형 호스팅 서비스를 사용하면 FlexMatch 매치메이킹에 대한 추가 요금은 발생하지 않습니다.

둘 이상의 AWS 리전 또는 둘 이상의 인스턴스 유형에서 게임 서버를 호스팅하거나 호스팅할 계획인 경우 각 리전 및 인스턴스 유형에 대한 추정치를 생성합니다.

Amazon GameLift Servers 인스턴스

이 섹션에서는 플레이어를 위한 게임 세션을 호스팅하는 데 필요한 컴퓨팅 리소스의 유형과 수를 추정하는 데 도움이 됩니다. Amazon GameLift Servers는 [Amazon Elastic Compute Cloud\(Amazon EC2\) 인스턴스](#)를 사용하여 게임 서버를 관리합니다. Amazon GameLift Servers에서는 특정 인스턴스 유형 및 운영 체제를 갖춘 인스턴스 플릿을 배포합니다. 플릿을 여러 개 보유하고 있거나 보유할 계획이라면 각 플릿에 대한 추정치를 산출합니다.

시작하려면 [구성 Amazon GameLift Servers 페이지](#)를 엽니다 AWS Pricing Calculator. 설명을 추가하고 리전을 선택한 다음 Amazon GameLift Servers 호스팅 추정(인스턴스+데이터 전송)을 선택합니다. Amazon GameLift Servers 인스턴스에서 다음 필드를 작성합니다.

- 최대 동시 플레이어 수(최대 CCU)

게임 서버에 동시에 연결할 수 있는 최대 플레이어 수입니다. 이 필드는 Amazon GameLift Servers가 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 GameLift Servers가 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 Servers 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.

다. Amazon GameLift Servers가 스팟 인스턴스를 사용하는 방법에 대한 자세한 정보는 [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 GameLift Servers는 60개의 온디맨드 인스턴스 및 40개의 스팟 인스턴스를 유지합니다.

- 인스턴스 유형

Amazon GameLift Servers 플릿은 컴퓨팅 성능, 메모리, 스토리지 및 네트워킹 기능이 서로 다른 다양한 Amazon EC2 인스턴스 유형을 사용할 수 있습니다. Amazon GameLift Servers 플릿을 구성할 때 게임의 요구 사항에 가장 적합한 인스턴스 유형을 선택합니다. Amazon GameLift Servers에서 인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 [관리형 플릿의 컴퓨팅 리소스 선택](#) 섹션을 참조하세요.

Amazon GameLift Servers 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.

의 [구성 Amazon GameLift Servers 페이지에서](#) 데이터 전송(DTO)을 AWS Pricing Calculator 확장한 다음 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 4입니다.

Amazon GameLift Servers 예상 요금을 구성했으면 내 예상 비용에 추가를 선택합니다. 에서 견적을 생성하고 관리하는 방법에 대한 자세한 내용은 AWS Pricing Calculator 사용 설명서의 [견적 생성, 서비스 구성 및 서비스 추가](#)를 AWS Pricing Calculator참조하세요.

비용 최적화 전략

이러한 전략을 관리형 호스팅 솔루션과 함께 사용하면 고성능 및 플레이어 경험을 유지하면서 클라우드 호스팅 비용을 줄일 수 있습니다.

모범 사례

다음 팁을 따라 게임 호스팅에 대한 비용 최적화 사례를 통합하세요. 최소한 Amazon GameLift Servers 비용 및 사용 패턴을 정기적으로 검토하는 것이 좋습니다. 요금 관련 질문이 있는 경우 담당팀에 문의하세요.

플릿 구성

- 요금 계산기 사용 - 게임 호스팅 데이터와 함께 이 도구를 사용하여 옵션을 탐색하고 비용 절감을 위한 잠재적 구성 시나리오를 테스트합니다. [Amazon GameLift Servers용AWS 요금 계산기](#)를 참조하세요.
- 개발 비용 절감 - 개발 및 테스트의 경우 더 작고 저렴한 인스턴스 유형을 사용하고 필요할 때만 이를 실행합니다.
- 게임과 인스턴스 유형 매치 - 게임 요구 사항에 가장 적합한 플릿의 Amazon EC2 인스턴스 패밀리, 유형, 크기를 선택합니다. [관리형 플릿의 컴퓨팅 리소스 선택](#)에서 인스턴스 유형 섹션을 참조하세요.

- Graviton 인스턴스 유형 사용 - Graviton 인스턴스는 ARM 기반 프로세서로 구동됩니다. 온디맨드 요금 모델을 사용하면 동급 x86 기반 인스턴스보다 가격 대비 성능, 뛰어난 에너지 효율성, 저렴한 비용을 제공합니다.
- 스팟 인스턴스 사용 - 스팟 인스턴스는 일반적으로 온디맨드보다 비용이 저렴하지만 항상 사용할 수 있는 것은 아닙니다. 스팟 플릿과 온디맨드 플릿을 결합하여 저비용과 고가용성의 균형을 맞춥니다(스팟 사용량 50~85% 권장). 중요하지 않은 게임 모드 또는 사용량이 적은 시간에 스팟 인스턴스를 사용합니다. 스팟 인스턴스 실행 가능성을 모니터링하려면 InstanceInterruptions 및 GameServerInterruptions와 같은 지표를 추적합니다. [온디맨드 인스턴스 및 스팟 인스턴스 비교](#)을(를) 참조하세요.
- Linux에서 서버 실행 - Linux용 게임 서버 런타임을 구축합니다. Linux와 함께 배포되는 인스턴스는 일반적으로 Windows보다 비용 효율적입니다.
- 리소스 사용을 최적화 - 성능을 유지하면서 가능한 한 많은 동시 게임 서버 프로세스를 실행하여 효율성을 극대화하도록 플릿을 구성합니다. [관리형 Amazon GameLift Servers에서 게임 서버 런타임 구성 최적화](#)을(를) 참조하세요.
- 다중 리전 전략 생성 - 게임 호스팅을 여러 위치에 배포하면 플레이어의 지연 시간을 줄이고 백업 옵션을 개선할 수 있습니다. 가장 큰 플레이어 마켓에 호스팅을 먼저 배포하고 피크 시간 동안 오버플로 용량에 보조 위치를 사용하여 리전 적용 범위와 비용 효율성의 균형을 맞춥니다. [다중 위치 대기열 빌드](#)을(를) 참조하세요.

플릿 용량 조정

- 0으로 조정 - 플릿을 사용하지 않는 경우 불필요한 요금이 발생하지 않도록 플릿 용량을 0으로 수동으로 설정합니다. [Amazon GameLift Servers 플릿의 용량을 수동으로 설정](#)을(를) 참조하세요.
- Auto Scaling 추가 - Auto Scaling을 사용하여 게임 호스팅 용량을 조정하여 호스팅 리소스의 과다 프로비저닝을 방지합니다. 용량을 변동하는 플레이어 수요 및 기타 주요 지표와 일치시킵니다. [Amazon GameLift Servers를 통해 플릿 용량 Auto-Scale](#)을(를) 참조하세요.
- 버퍼 유지 관리 - 플레이어를 기다리지 않고 플레이어 수요의 갑작스러운 급증을 처리하려면 대상 추적을 사용하여 유휴 게임 서버의 버퍼를 유지합니다. 플레이어 베이스의 크기와 사용 패턴에 따라 버퍼를 사용자 지정합니다. [대상 기반 Auto Scaling](#)을(를) 참조하세요.

게임 세션 배치

- 배치 대기열 사용 - Amazon GameLift Servers 대기열은 알고리즘을 사용하여 호스팅 리소스 비용 및 기타 요인을 기반으로 게임 세션의 '최상'의 호스팅 위치를 결정합니다. [게임 세션 배치 구성](#)을(를) 참조하세요.

- 대기열 우선 순위 사용자 지정 - 게임 세션을 배치할 때 대기열이 호스팅 비용의 우선 순위를 지정하는 방법을 변경할 수 있습니다. [게임 세션 배치 우선순위를\(를\)](#) 참조하세요.
- 배치 지표 모니터링 - 대기열이 게임 세션의 위치를 선택하는 방법을 추적하여 최적화 기회를 식별합니다. [대기열에 대한 Amazon GameLift Servers 지표](#)을(를) 참조하세요.

데이터 전송

- 데이터 전송 추적 - 게임 클라이언트와 서버 간에 데이터가 전송되는 방식을 모니터링하고 활동을 최적화하기 위한 조치를 취합니다.
- 데이터 압축 사용 - 게임 클라이언트와 서버 간의 데이터 전송을 위한 데이터 압축 기술을 구현하는 것이 좋습니다. 압축은 대역폭 사용량을 줄이고 게임 플레이 성능과 지연 시간을 개선하는 데 중요한 역할을 합니다.

리소스 비용 및 사용률 도구

를 사용하여 AWS 도구를 사용하여 게임 호스팅 비용을 모니터링하고 최적화하는 방법을 알아봅니다 Amazon GameLift Servers. 추가 도구에 대한 자세한 내용은 [AWS 결제 및 비용 관리](#) 섹션을 참조하세요.

AWS 비용 관리 도구

- 결제 콘솔 - AWS 청구서 및 사용량을 검토합니다. [과금 정보 설정](#)을 참조하세요.
- 프리 티어 사용 알림 - 프리 티어 한도에 가까워지면 알림을 설정합니다. [AWS 프리 티어 사용량 추적을 참조하세요](#).
- Amazon CloudWatch 결제 알림 - 사용량이 사용자 지정 임계값에 도달하면 알림을 구성합니다. [결제 경보를 생성하여 예상 AWS 요금 모니터링](#)을 참조하세요.

비용 할당 및 추적

- 비용 할당 태그 - 플릿 및 기타 리소스에 태그를 지정하여 호스팅 비용을 구성하고 추적합니다. [비용 할당 태그를 사용하여 AWS 비용 구성 및 추적](#)을 참조하세요.
- 비용 보고서 - 할당된 태그별로 분류된 보고서를 생성합니다. [월별 비용 할당 보고서 사용](#)을 참조하세요.
- AWS Cost Explorer - 필터링 및 사용자 지정 가능한 보기를 사용하여 비용, 추세, 예측을 분석합니다. [를 사용한 비용 및 사용량 분석을 참조하세요 AWS Cost Explorer](#).

- AWS Budgets - AWS 비용 및 사용량을 추적하고 조치를 취합니다. [를 사용한 비용 관리를 참조하세요 AWS Budgets](#).

Amazon GameLift Servers 및 Amazon CloudWatch를 사용한 성능 모니터링

이러한 주요 지표를 모니터링하여 리소스 사용률을 최적화합니다. Amazon GameLift Servers 콘솔에서 지표를 보거나 Amazon CloudWatch 대시보드를 사용합니다. Amazon GameLift Servers에 사용 가능한 모든 지표에 대한 자세한 내용은 [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#) 섹션을 참조하세요.

- 인스턴스 지표 - ActiveInstances, IdleInstances, PercentIdleInstances
- 서버 프로세스 지표 - ActiveServerProcesses, HealthyServerProcesses
- 게임 세션 지표 - ActiveGameSessions, AvailableGameSessions
- 플레이어 세션 지표 - CurrentPlayerSessions
- 대기열 지표 - AverageWaitTime, QueueDepth
- 매치메이킹 지표 - CurrentTickets, MatchesPlaced
- 하드웨어 성능 - CPUUtilization, NetworkIn/NetworkOut, DiskReadBytes/DiskWriteBytes
- 리소스 사용률 지표
 - PercentIdleInstances를 사용하여 최적의 플릿 크기를 결정합니다.
 - PercentAvailableGameSessions을 추적하여 게임 호스팅 용량이 충분한지 확인합니다.
 - InstanceInterruptions 및 GameServerInterruptions을 모니터링하여 스팟 인스턴스 실행 가능성을 결정합니다.

Amazon GameLift Servers 참조 안내서

이 섹션에는 Amazon GameLift Servers를 사용하기 위한 참조 문서가 포함되어 있습니다.

주제

- [Amazon GameLift Servers의 서비스 API](#)
- [Amazon GameLift Servers용 서버 SDK 5.x](#)
- [Amazon GameLift Servers용 서버 SDK 버전 4 이하](#)
- [게임 세션 배치 이벤트](#)
- [Amazon GameLift Servers AMI 버전](#)
- [Amazon GameLift Servers 서비스 위치](#)
- [Amazon GameLift Servers 엔드포인트 및 할당량](#)
- [Amazon GameLift Servers API 제한](#)
- [Amazon GameLift Servers SDK API 제한](#)
- [Amazon GameLift Servers UDP 핑 비콘](#)

Amazon GameLift Servers의 서비스 API

이 작업 기반 목록을 사용하여 Amazon GameLift Servers 게임 호스팅 솔루션 및 기타 기능을 구축할 때 API 작업을 찾을 수 있습니다. AWS SDK는 `aws.gamelift` 네임스페이스에 이러한 작업을 포함합니다. [AWS SDK를 다운로드](#)하거나 [Amazon GameLift Servers API 참조 설명서를 확인합니다](#). AWS 명령 [AWS CLI 참조](#)에 설명된 대로 API를 명령줄 인터페이스(AWS CLI)와 함께 사용할 수도 있습니다.

API에는 관리형 게임 호스팅을 위한 두 가지 작업 세트가 포함되어 있습니다.

- [Amazon GameLift Servers 호스팅 리소스 관리](#)
- [게임 세션 시작 및 플레이어 참여](#)

Amazon GameLift Servers 서비스 API에는 다른 Amazon GameLift Servers 도구 및 솔루션과 함께 사용할 수 있는 작업도 포함되어 있습니다. FleetIQ API 작업 목록은 [FleetIQ API 작업](#)을 참조하세요. 매치메이킹을 위한 FlexMatch API 목록은 [FlexMatch API 작업](#)을 참조하세요.

Amazon GameLift Servers 호스팅 리소스 관리

이러한 작업을 호출하여 게임 서버의 호스팅 리소스 구성, 플레이어 수요에 맞게 용량 조정, 성능 및 사용량 지표 등을 구성할 수 있습니다. 이러한 API 작업은 Amazon GameLift Servers Realtime을 포함하여 Amazon GameLift Servers에서 호스팅되는 게임 서버에서 사용됩니다. 대부분의 리소스 관리 작업은 [Amazon GameLift Servers 콘솔](#)에서도 작업할 수 있으며 또는 AWS Command Line Interface (AWS CLI) 도구로 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 Servers 빌드 리소스를 생성합니다. AWS CLI 명령으로 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 Servers 리전에 업로드된 모든 빌드 목록을 가져옵니다.
- [DescribeBuild](#) - 빌드와 관련된 정보를 검색합니다.
- [UpdateBuild](#) - 빌드 이름 및 버전을 포함하여 빌드 메타데이터를 변경합니다.
- [DeleteBuild](#) - Amazon GameLift Servers에서 빌드를 제거합니다.

Amazon GameLift Servers Realtime 구성 스크립트 관리

- [CreateScript](#) - JavaScript 파일을 업로드하고 새 Amazon GameLift Servers 스크립트 리소스를 생성합니다.
- [ListScripts](#) Realtime - Amazon GameLift Servers 리전에 업로드된 모든 스크립트 목록을 가져옵니다.
- [DescribeScript](#) - Realtime 스크립트와 관련된 정보를 검색합니다.
- [UpdateScript](#) - 스크립트 메타데이터를 변경하고 수정된 스크립트 콘텐츠를 업로드합니다.
- [DeleteScript](#) - Amazon GameLift Servers에서 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 Servers 컴퓨팅 리소스 플릿을 구성하고 배포합니다. 일단 배포되면 게임 서버가 구성된 대로 자동으로 시작되고 게임 세션을 호스팅할 준비가 됩니다.
- [ListFleets](#) - Amazon GameLift Servers 리전의 모든 플릿 목록을 가져옵니다.
- [DeleteFleet](#) - 더 이상 게임 서버나 호스팅 플레이어를 실행하지 않는 플릿을 제거합니다.
- 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 계정 및 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.
- 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.

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 Servers 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 Servers 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.

관리형 호스팅 인스턴스에 연결

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 Servers 호스팅 리소스와 기타 AWS 리소스 간에 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 Servers 플릿의 VPC와 VPC 중 하나 간에 피어링 연결을 설정합니다 VPCs.
- [DescribeVpcPeeringConnections](#) - Amazon GameLift Servers 플릿과의 활성 또는 보류 중인 VPC 피어링 연결에 대한 정보를 검색합니다.
- [DeleteVpcPeeringConnection](#) - Amazon GameLift Servers 플릿과의 VPC 피어링 연결을 삭제합니다.

게임 세션 시작 및 플레이어 참여

백엔드 서비스에서 이러한 작업을 호출하여 새 게임 세션을 시작하고, 기존 게임 세션에 대한 정보를 얻으며, 플레이어를 게임 세션에 참여시킵니다. 이러한 작업은 Amazon GameLift Servers에서 호스팅되는 사용자 지정 게임 서버로 사용하기 위한 것입니다. Amazon GameLift Servers Realtime을 사용하는 경우 [Amazon GameLift Servers Realtime 클라이언트 API\(C#\)](#)를 사용하여 게임 세션을 관리합니다.

- 하나 이상의 플레이어를 위한 새로운 게임 세션 시작.
 - [StartGameSessionPlacement](#) - Amazon GameLift Servers에 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 알고리

좁을 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 Servers용 서버 SDK 5.x

이 섹션에서는 Amazon GameLift Servers용 서버 SDK 5.x에 대한 참조 문서를 제공합니다. 서버 SDK는 게임 서버가 Amazon GameLift Servers 서비스와 상호 작용하는 데 사용하는 핵심 기능을 제공합니다. 예를 들어 게임 서버는 서비스로부터 게임 세션을 시작 및 중지하라는 프롬프트를 수신하고 서비스에 대한 정기적인 게임 세션 상태 업데이트를 제공합니다. 호스팅을 위해 배포하기 전에 게임 서버를 서버 SDK와 통합합니다.

서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 사용할 사용자 지정 멀티플레이어 게임 서버를 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 페이지를 참조하세요.

Amazon GameLift Servers용 서버 SDK의 최신 메이저 버전은 5.x입니다. 다음 호스팅 기능을 사용하려면 버전 5.x를 사용해야 합니다.

- Amazon GameLift Servers Anywhere
- Unreal Engine 및 Unity용 Amazon GameLift Servers 플러그인

 Note

서버 SDK 버전 4.x 이하를 사용해야 하는 경우 설명서 및 다운로드 정보는 [Amazon GameLift Servers용 서버 SDK 버전 4 이하](#) 섹션을 참조하세요.

주제

- [Amazon GameLift Servers용 서버 SDK 5의 업데이트](#)
- [Amazon GameLift Servers용 서버 SDK 5.x로 마이그레이션](#)
- [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)
- [Amazon GameLift Servers용 Go 서버 SDK - 작업](#)
- [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)

Amazon GameLift Servers용 서버 SDK 5의 업데이트

호스팅 중인 게임 서버가 Amazon GameLift Servers용 서버 SDK를 사용하여 Amazon GameLift Servers 서비스와 통신하고 플레이어를 위해 게임 세션을 시작 및 관리할 수 있습니다. 최신 버전인 Amazon GameLift Servers 서버 SDK 5는 새로운 Amazon GameLift Servers 기능에 대한 다양한 개선 사항과 지원을 제공합니다. 게임 서버 빌드가 현재 Amazon GameLift Servers 서버 SDK 4 이하를 사용하는 경우 이 주제의 지침에 따라 게임을 업데이트합니다.

Amazon GameLift Servers 서버 SDK 버전 5.0.0 이상에는 다음과 같은 업데이트가 포함되어 있습니다.

- 확장된 언어 - 라이브러리는 C++, C#, Go 언어로 제공됩니다. Unreal Engine과 함께 사용할 C++ 라이브러리를 빌드할 수 있습니다.
- 게임 엔진 플러그인 지원 - Unreal Engine 및 Unity용 Amazon GameLift Servers 독립 실행형 플러그인에는 Amazon GameLift Servers 서버 SDK 5 라이브러리가 필요합니다. 이러한 플러그인은 호스

팅을 위해 Amazon GameLift Servers에 게임을 통합, 테스트 및 배포하기 위한 안내 워크플로를 제공합니다. [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 및 [Unreal Engine용 Amazon GameLift Servers 플러그인](#) 설명서를 참조하세요.

- Amazon GameLift Servers Anywhere 지원 - Anywhere 플릿을 사용하면 Amazon GameLift Servers 기능(매치메이킹 포함)을 사용하도록 자체 호스팅 리소스를 설정할 수 있습니다. Amazon GameLift Servers 에이전트를 추가하여 게임 세션 수명 주기 관리를 자동화합니다. 온프레미스 하드웨어를 사용하여 프로덕션 호스팅에 Anywhere 플릿을 사용하거나 빠른 반복 게임 개발을 위해 테스트 환경을 설정합니다. [Anywhere 호스팅](#) 및 [Amazon GameLift Servers 에이전트](#)를 참조하세요.
- 업데이트된 테스트 도구 - Amazon GameLift Servers Anywhere 기능으로 게임에 대한 로컬 또는 클라우드 기반 테스트 환경을 설정할 수 있습니다. Amazon GameLift Servers 에이전트 사용 여부에 상관없이 테스트를 설정합니다. 이러한 도구는 Amazon GameLift Servers Local을 대체합니다. [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#)(를) 참조하세요.
- C#용 통합 .NET 솔루션 - C# 서버 SDK 5.1+는 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.
- 새 Compute 리소스 - 이 새 리소스는 다양한 유형의 호스팅 리소스를 결합합니다. 여기에는 클라우드 기반 호스팅 리소스(관리형 EC2 또는 컨테이너 플릿) 및 고객 제어 호스팅 리소스(Anywhere 플릿)가 포함됩니다. 다음과 같은 업데이트가 포함되어 있습니다.
 - Compute 리소스에 대한 새 API 호출: [ListCompute\(\)](#), [DescribeCompute\(\)](#) 및 [GetComputeAccess\(\)](#) 이러한 작업은 모든 유형의 Amazon GameLift Servers 플릿에 대한 호스팅 리소스 정보를 반환합니다. 일반적으로 서버 SDK 5.x를 사용하는 게임 서버가 있는 플릿의 경우 컴퓨팅별 작업을 사용하여 인스턴스별 작업을 대체합니다. 또한 Amazon GameLift Servers 에이전트가 없는 Anywhere 플릿에는 [RegisterCompute\(\)](#), [DeregisterCompute\(\)](#), [GetComputeAuthToken\(\)](#) 작업이 사용됩니다.
 - CloudWatch 영역 FleetId, Location, ComputeType를 사용하는 신규 지표 ActiveCompute: 이 지표는 이전 지표 ActiveInstances를 대체합니다.
- 원격 액세스를 위한 Amazon EC2 Systems Manager(SSM) - 보안을 강화하려면 Amazon GameLift Servers 관리형 플릿의 인스턴스에 연결할 때 SSH 대신 SSM을 사용합니다. [플릿 인스턴스에 연결](#)(를) 참조하세요.

Amazon GameLift Servers용 서버 SDK 5.x로 마이그레이션

Server SDK 버전 5.x를 사용하도록 게임 프로젝트를 업데이트하려면 다음과 같이 변경합니다.

Note

서버 SDK 버전 4.x 이하를 계속 사용해야 하는 경우 설명서 및 다운로드 정보는 [Amazon GameLift Servers용 서버 SDK 버전 4 이하](#) 섹션을 참조하세요. SDK 버전 4.0.2에 한하여 [공식 GitHub 릴리스](#)에서 다운로드할 수 있습니다.

1. 개발 환경에 대한 최신 Amazon GameLift Servers용 서버 SDK 패키지[[다운로드 사이트](#)]를 다운로드합니다. 다운로드한 패키지 및 버전에 대한 Readme 파일의 설치 지침을 따릅니다. 게임 프로젝트에서 서버 SDK를 사용하려면 다음 지침을 참조하세요.

- [C++, C# 또는 Go를 사용하는 개발 환경](#)
- [Unreal Engine 프로젝트\(Unreal 라이브러리를 C++ 서버 SDK만 해당\)](#)
- [Unity 프로젝트의 경우\(Unity 라이브러리를 C# 서버 SDK만 해당\)](#)
- [Unreal Engine용 Amazon GameLift Servers 플러그인과 함께 사용](#)
- [Unity용 Amazon GameLift Servers 플러그인과 함께 사용](#)

2. 다음과 같이 서버 코드를 업데이트합니다.

- 서버 코드 콜백 함수 onCreateGameSession()을 onStartGameSession()으로 변경합니다.
- 필요에 따라 InitSDK() 입력을 업데이트합니다.

- Amazon GameLift Servers 관리형 EC2 플릿 또는 Amazon GameLift Servers 에이전트를 사용하는 Anywhere 플릿에 게임 서버 빌드를 배포하려는 경우:

([C++](#)) ([C#](#)) ([Unreal](#)) 파라미터 없이 InitSDK()를 호출합니다. 호출 후 컴퓨팅 환경과 Amazon GameLift Servers 서비스에 대한 WebSocket 연결이 설정됩니다.

- Amazon GameLift Servers 에이전트 없이 게임 서버 빌드를 Anywhere 플릿에 배포하려는 경우:

([C++](#)) ([C#](#)) ([Unreal](#)) 서버 파라미터로 InitSDK()를 호출합니다. 게임 서버 프로세스가 이러한 파라미터를 사용하여 Amazon GameLift Servers 서비스와의 연결을 설정합니다.

3. 게임 서버 빌드 또는 기타 호스팅 애플리케이션이 실행 중에 다른 AWS 리소스와 통신하는 경우 애플리케이션이 해당 리소스에 액세스하는 방법을 변경해야 합니다. AssumeRoleCredentials의 사용을 새 서버 SDK 작업 GetFleetRoleCredentials()(게임 서버용)로 바꾸거나 공유 자격 증명(다른 애플리케이션의 경우)을 사용합니다. 이 변경 사항을 구

현하는 방법에 대한 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 페이지를 참조하세요.

4. 프로젝트가 서버 SDK 작업 `GetInstanceCertificate()`을 호출하여 TLS 인증서를 검색한 경우 대신 새 `GetComputeCertificate()`([C++](#)) ([C#](#)) ([Unreal](#))를 사용하도록 코드를 수정합니다.
5. Amazon GameLift Servers에 게임 빌드를 업로드할 때(예: [upload-build](#) 또는 [CreateBuild\(\)](#)) `ServerSdkVersion` 파라미터를 사용 중인 5.x 버전으로 설정합니다(이 파라미터는 현재 기본값이 4.0.2임). 이 파라미터는 게임 서버 빌드의 실제 서버 SDK 라이브러리와 일치해야 합니다. 업로드된 게임 서버 빌드에 잘못된 버전을 지정하면 해당 빌드로 생성한 모든 플릿이 실패합니다. [Amazon GameLift Servers의 게임 서버 빌드 생성을\(를\)](#) 참조하세요.

다음 예제에서는 서버 SDK 버전을 지정하는 방법을 보여줍니다.

```
aws gamelift upload-build \
  --operating-system AMAZON_LINUX_2023 \
  --server-sdk-version "5.0.0" \
  --build-root "~/mygame" \
  --name "My Game Nightly Build" \
  --build-version "build 255" \
  --region us-west-2
```

6. 스크립트를 사용하여 관리형 플릿에 원격으로 연결하는 경우 [플릿 인스턴스에 연결](#)에 설명된 대로 새 프로세스를 사용하도록 스크립트를 업데이트합니다.

Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업

해당 서버 SDK 5.x 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 페이지를 참조하세요.

Note

이 주제에서는 C++ Standard Library(std)로 빌드할 때 사용할 수 있는 Amazon GameLift Servers C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DDGAMELIFT_USE_STD=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

Amazon GameLift Servers용 C++ 서버 SDK 5.x - 데이터 유형

Amazon GameLift Servers C++ 서버 SDK 5.x 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

이 주제에서는 C++ Standard Library(std)로 빌드할 때 사용할 수 있는 Amazon GameLift Servers C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DDGAMELIFT_USE_STD=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

[Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#)

데이터 타입

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [AttributeValue](#)
- [GetFleetRoleCredentialsRequest](#)
- [AwsLongOutcome](#)
- [AwsStringOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [GenericOutcome](#)
- [GenericOutcomeCallable](#)

- [PlayerSessions](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [InitSDKOutcome](#)
- [GameLiftError](#)
- [Enums](#)

LogParameters

Amazon GameLift Servers가 게임 세션 종료 후 업로드하고 저장하기를 원하는 게임 세션 중에 생성된 파일을 식별하는 객체입니다. 게임 서버는 [ProcessReady\(\)](#) 호출 시 ProcessParameters 객체의 일부로 LogParameters를 Amazon GameLift Servers에 제공합니다.

속성	설명
LogPaths	Amazon GameLift Servers가 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: std::vector<std::string> 필수 항목 여부: 아니요

MetricsParameters

이 데이터 유형을 사용하여 게임 서버에 대한 지표 수집 및 충돌 보고를 구성합니다. 게임 서버는 [InitMetrics\(\)](#) 호출을 통해 MetricsParameters를 Amazon GameLift Servers에 전달합니다.

속성	설명
StatsdHost	지표 수집을 위한 StatsD 서버의 호스트 이름 또는 IP 주소. 유형: std::string 필수 항목 여부: 아니요
StatsdPort	지표 수집을 위한 StatsD 서버의 포트 번호. 유형: int 필수 항목 여부: 아니요
CrashReporterHost	충돌 리포터 서버의 호스트 이름 또는 IP 주소. 유형: std::string 필수 항목 여부: 아니요
CrashReporterPort	충돌 리포터 서버의 포트 번호. 유형: int 필수 항목 여부: 아니요
FlushIntervalMs	지표 데이터 플러시에 대한 밀리초 단위의 간격. 유형: int 필수 항목 여부: 아니요
MaxPacketSize	서버로 전송되는 지표 패킷의 최대 크기(바이트 단위). 유형: int

필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 [ProcessReady\(\)](#)에서 Amazon GameLift Servers로 보낸 파라미터 집합이 포함됩니다.

속성	설명
LogParameters	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon GameLift Servers는 나중에 액세스할 수 있도록 파일을 복사하고 저장합니다. 유형: <code>Aws::GameLift::Server::LogParameters</code> 필수 항목 여부: 아니요
OnHealthCheck	Amazon GameLift Servers가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출하고 60초 동안 응답을 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon GameLift Servers는 서버 프로세스를 정상이 아닌 것으로 기록합니다. 유형: <code>std::function<bool()> onHealthCheck</code> 필수 항목 여부: 아니요
OnProcessTerminate	Amazon GameLift Servers가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분

을 기다렸다가 [ProcessEnding\(\)](#) 호출로 응답한 후 서버 프로세스를 종료합니다.

유형: `std::function<void()>`
`onProcessTerminate`

필수 항목 여부: 예

OnStartGameSession

Amazon GameLift Servers가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응답으로 이 함수를 호출합니다. 콜백 함수가 [GameSession](#) 객체를 전달합니다.

유형: `const std::function<void (Aws::GameLift::Model::GameSession)>` `onStartGameSession`

필수 항목 여부: 예

OnUpdateGameSession

Amazon GameLift Servers가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수입니다. 매치 채우기 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 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

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	GameSession 객체입니다. GameSession 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 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

속성	설명
	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. <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> 유형: <code>std::string</code> 필수 항목 여부: 아니요
이름	게임 세션을 설명하는 레이블입니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
FleetId	게임 세션이 실행 중인 플릿의 고유 식별자입니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
MaximumPlayerSessionCount	게임 세션에 연결된 최대 플레이어 수입니다. 유형: <code>int</code> 필수 항목 여부: 아니요

속성	설명
Port	게임 세션의 포트 번호입니다. Amazon GameLift Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: <code>in</code> 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
GameSessionData	단일 문자열 값으로 포맷된 사용자 지정 게임 세션 속성의 집합입니다. 유형: <code>std::string</code> 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: <code>std::string</code> 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: <code>std :: vector < GameProperty ></code> 필수 항목 여부: 아니요
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. - TLS 지원 플릿: <code><unique identifier>.<region identifier>.amazon gamelift.com</code> - TLS 미지원 플릿: <code>ec2-<unique identifier>.compute.amazonaws.com</code> TLS 지원 플릿에서 실행되는 게임 세션에 연결할 때는 IP 주소가 아닌 DNS 이름을 사용해야 합니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요

ServerParameters

게임 서버 프로세스가 Amazon GameLift Servers 서비스와의 연결을 설정하는 데 사용하는 정보입니다. Amazon GameLift Servers 에이전트 없이 게임 서버 빌드가 Anywhere 플릿 또는 컨테이너 플릿에 배포되는 경우에만 [InitSDK\(\)](#)를 호출할 때 이러한 파라미터를 포함합니다. 다른 모든 배포 시나리오에서는 파라미터 없이 [InitSDK\(\)](#)를 호출합니다.

속성	설명
webSocketUrl	Amazon GameLift Servers Anywhere 컴퓨팅 리소스에 대해 RegisterCompute 를 수행하면 GameLiftServerSdkEndpoint Amazon GameLift Servers가 반환됩니다. 유형: std::string 필수 항목 여부: 예
processId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: std::string 필수 항목 여부: 예
hostId	HostID는 컴퓨터를 등록할 때 사용되는 ComputeName 입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: std::string 필수 항목 여부: 예
fleetId	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: std::string 필수 항목 여부: 예
authToken	Amazon GameLift Servers에서 생성한 인증 토큰으로, Amazon GameLift Servers에 서버를 인증합니다. 자세한 내용은 GetComputeAuthToken을 참조하세요. 유형: std::string

속성	설명
	필수 항목 여부: 예

StartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 [StartMatchBackfill\(\)](#) 호출을 통해 이 정보를 Amazon GameLift Servers에 전달합니다.

속성	설명
GameSessionArn	고유한 게임 세션 식별자입니다. API 작업 GetGameSessionId 는 ARN 형식의 식별자를 반환합니다. 유형: <code>std::string</code> 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: <code>std::string</code> 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: <code>std::vector<Player></code> 필수 항목 여부: 예

속성	설명
TicketId	매치메이킹 또는 매치 채우기 요청 티켓의 고유 식별자입니다. 값을 제공하지 않으면 Amazon GameLift Servers에서 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다. 유형: <code>std::string</code> 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. Amazon GameLift Servers는 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: <code>Dictionary<string,int></code> 필수 항목 여부: 아니요
PlayerAttributes	매치메이킹에 사용할 플레이어 정보가 포함된 키:값 페어의 모음입니다. 플레이어 속성 키는 매치메이킹 규칙 세트에 사용되는 <code>PlayerAttributes</code>와 일치해야 합니다.

속성	설명
	플레이어 속성에 대한 자세한 내용은 Attribute Value를 참조하세요. 유형: <code>std::map<std::string, AttributeValue></code> 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요

DescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon GameLift Servers로 [DescribePlayerSessions\(\)](#) 호출을 통해 이 정보를 제공합니다.

속성	설명
GameSessionId	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> 입니다. GameSessionID 는 다음입니다. 사용자 지정 ID 문자열 또는

속성	설명
	유형: <code>std::string</code> 필수 항목 여부: 아니요
PlayerSessionId	플레이어 세션의 고유 식별자입니다. 이 파라미터를 사용하여 단일 특정 플레이어 세션을 요청합니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 이 파라미터를 사용하여 특정 플레이어에 대한 모든 플레이어 세션을 요청합니다. 플레이어 ID 생성 (를) 참조하세요. 유형: <code>std::string</code> 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	결과와 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\(\)](#) 호출을 통해 이 정보를 Amazon GameLift Servers 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 카-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: <code>std::string</code> 필수 항목 여부: 아니요
N	숫자 속성 값을 나타냅니다. 유형: <code>double</code> 필수 항목 여부: 아니요
SL	문자열 속성 값의 배열을 나타냅니다. 유형: <code>std::vector<std::string></code> 필수 항목 여부: 아니요
SDM	문자열 키와 이중 값의 사전을 나타냅니다. 유형: <code>std::map<std::string, double></code> 필수 항목 여부: 아니요

GetFleetRoleCredentialsRequest

이 데이터 유형은 게임 서버에 다른 AWS 리소스에 대한 제한된 액세스 권한을 부여합니다. 자세한 내용은 단원을 참조하십시오 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#).

속성	설명
RoleArn	리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon AWS 리소스 이름(ARN)입니다. 유형: std::string 필수 항목 여부: 아니요
RoleSessionName	AWS Security Token Service AssumeRole 세션을 고유하게 식별하는 데 사용할 수 있는 역할 세션 이름입니다. 이 이름은 CloudTrail의 감사 로그와 같은 감사 로그에 표시됩니다. 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: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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: <code>std::string</code> 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue 로 변환합니다. 유형: <code>long&&</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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 rvalue 로 변환합니다. 유형: <code>Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

DescribePlayerSessionsResult

요청과 일치하는 각 플레이어 세션의 속성을 포함하는 객체 모음입니다.

속성	설명
NextToken	결과와 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: <code>std::string</code> 필수 항목 여부: 예
PlayerSessions	유형: <code>IList<the section called "PlayerSessions"></code> 필수 항목 여부:
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다. 유형: <code>std::string&&</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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 발생하는 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 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

PlayerSessions

이 데이터 유형은 Amazon GameLift Servers가 게임 서버로 전달하는 플레이어 세션을 나타냅니다. 자세한 내용은 [PlayerSession](#)을 참조하세요.

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
Status	결과를 필터링하는 기준이 되는 플레이어 세션 상태입니다. PlayerSessionId 또는 PlayerId가 제공되면 PlayerSessionStatusFilter는 응답에 영향을 주지 않습니다. 유형: PlayerSessionStatus 열거형. 가능한 값은 다음을 포함합니다. ACTIVE

속성	설명
	• 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 rvalue 로 변환합니다. 유형: StartMatchBackfillResult&& 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool

속성	설명
	필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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 ID가 지정되지 않으면 Amazon GameLift Servers는 UUID 형식으로 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하고 매치 결과를 검색할 수 있습니다. 유형: <code>std::string</code> 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” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue로 변환합니다.

속성	설명
	유형: <code>Aws::GameLift::Server::Model::GetComputeCertificateResult</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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 Servers 관리형 플릿을 사용하는 경우 이 경로에는 다음이 포함됩니다. <code>certificate.pem</code> : 최종 사용자 인증서입니다. 전체 인증서 체인은 이 인증서에 추가된 <code>certificateChain.pem</code> 의 조합입니다. <code>certificateChain.pem</code> : 루트 인증서와 중간 인증서를 포함하는 인증서 체인입니다. <code>rootCertificate.pem</code> : 루트 인증서입니다. <code>privateKey.pem</code> : 최종 사용자 인증서의 프라이빗 키입니다.

속성	설명
	유형: <code>std::string</code> 필수 항목 여부: 아니요
ComputeName	컴퓨팅 리소스의 이름입니다. 유형: <code>std::string</code> 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 rvalue 로 변환합니다. 유형: <code>Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError”

속성	설명
	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 식별하기 위한 토큰입니다.

속성	설명
	유형: <code>string</code> 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

InitSDKOutcome

 Note

InitSDKOutcome은 `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: <code>bool</code> 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: GameLiftErrorType 열거형 . 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: std::string 필수 항목 여부: 아니요
ErrorName	오류 유형 이름입니다. 유형: std::string 필수 항목 여부: 아니요

Enums

Amazon GameLift Servers용 서버 SDK(C++)에 대해 정의된 Enums는 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 - Amazon GameLift Servers에 대한 로컬 연결이 실패했습니다.
- NETWORK_NOT_INITIALIZED - 네트워크가 초기화되지 않았습니다.
- SERVICE_CALL_FAILED - AWS 서비스에 대한 호출이 실패했습니다.
- WEBSOCKET_CONNECT_FAILURE
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL

- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- ALREADY_INITIALIZED - Amazon GameLift Servers 서버 또는 클라이언트는 이미 Initialize()를 사용하여 초기화되었습니다.
- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALIZED - Amazon GameLift Servers 클라이언트가 초기화되지 않았습니다.
- GAMELIFT_SERVER_NOT_INITIALIZED - Amazon GameLift Servers 서버가 초기화되지 않았습니다.
- GAME_SESSION_ENDED_FAILED - Amazon GameLift Servers용 서버 SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY - Amazon GameLift Servers 서버 게임 세션이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED - Amazon GameLift Servers용 서버 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 Servers 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 Servers용 서버 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 Servers용 서버 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 보내는데 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.

Amazon GameLift Servers용 C++ 서버 SDK 5.x - 데이터 유형

주제

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [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.

예제

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitMetrics()

서버 성능 데이터를 수집하고 보고하기 위한 지표 시스템을 초기화합니다. 최상의 결과를 얻으려면 [InitSDK\(\)](#) 전에 이 메서드를 호출하여 SDK 초기화 중에 지표 수집을 활성화합니다.

구문

```
Aws::GameLift::GenericOutcome InitMetrics();
```



```
Aws::GameLift::GenericOutcome InitMetrics(const
    Aws::GameLift::Server::MetricsParameters &metricsParameters);
```

파라미터

[MetricsParameters](#) (선택 사항)

지표 수집을 위한 구성 파라미터입니다. 제공되지 않은 경우 환경 변수로 재정의할 수 있는 기본값을 사용합니다.

반환 값

성공하면 성공을 나타내는 [the section called “GenericOutcome”](#) 객체를 반환합니다. 실패하면 오류 메시지를 반환합니다.

예제

```
// Initialize with default parameters (uses environment variables if available)
// Defaults: localhost:8125 for StatsD, localhost:8126 for crash reporter
// FlushInterval: 10000ms, MaxPacketSize: 512 bytes
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::InitMetrics();
if (outcome.IsSuccess()) {
    // Metrics system initialized successfully
}

// Initialize with custom parameters
Aws::GameLift::Server::MetricsParameters metricsParams("localhost", 8125, "crash-host",
    8126, 5000, 1024);
Aws::GameLift::GenericOutcome customOutcome =
    Aws::GameLift::Server::InitMetrics(metricsParams);
if (customOutcome.IsSuccess()) {
    // Metrics system initialized with custom parameters
}
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화 단계가 발생하기 전에 출시 시점에서 이 메서드를 호출합니다. 이 작업은 호스트 환경에서 서버 파라미터를 읽어 게임 서버 프로세스와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다. 맥등성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

Amazon GameLift Servers 에이전트 없이 게임 서버 빌드가 Amazon GameLift Servers Anywhere 플릿 또는 컨테이너 플릿에 배포되는 경우 [InitSDK\(\)](#)를 호출하고 서버 파라미터 세트를 지정합니다.

구문

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

반환 값

서버 프로세스가 [ProcessReady\(\)](#)를 호출할 준비가 되었는지 여부를 나타내는 [the section called “InitSDKOutcome”](#) 객체를 반환합니다.

예제

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK();
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화 단계가 발생하기 전에 출시 시점에서 이 메서드를 호출합니다. 이 작업에는 게임 서버 프로세스와 Amazon GameLift Servers 서비스 간의 통신을 설정하기 위한 서버 파라미터 세트가 요구됩니다. 맥등성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

게임 서버 빌드가 Amazon GameLift Servers 관리형 EC2 플릿 또는 Amazon GameLift Servers 에이전트를 사용하는 Amazon GameLift Servers Anywhere 플릿 또는 컨테이너 플릿에 배포되는 경우 서버 파라미터 없이 [InitSDK\(\)](#)를 호출합니다.

구문

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

파라미터

[ServerParameters](#)

Amazon GameLift Servers Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 구성합니다.

- 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Servers Anywhere 컴퓨팅이 포함된 Amazon GameLift Servers 플릿의 ID입니다.
- Amazon GameLift Servers 작업을 통해 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.

예제

Amazon GameLift Servers Anywhere 예제

```
//Define the server parameters
std::string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
std::string processId = "PID1234";
std::string fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
std::string hostId = "HardwareAnywhere";
std::string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
Aws::GameLift::Server::Model::ServerParameters serverParameters =
    Aws::GameLift::Server::Model::ServerParameters(websocketUrl, authToken, fleetId,
        hostId, processId);

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. [InitSDK\(\)](#) 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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()

Amazon GameLift Servers 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 Servers 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.

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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 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.

예제

```
// 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 종료 중임을 Amazon GameLift Servers에 알립니다. 활성 게임 세션의 종료 등 다른 모든 정리 작업 이후 및 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.

구문

```
Aws::GameLift::GenericOutcome processEndingOutcome =
    Aws::GameLift::Server::ProcessEnding();
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화 후 onStartGameSession() 콜백 함수의 일부로 호출되어야 합니다.

구문

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =
    Aws::GameLift::Server::ActivateGameSession();
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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.

구문

```
GenericOutcome
UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy
newPlayerSessionPolicy);
```

파라미터

playerCreationSessionPolicy

형식: PlayerSessionCreationPolicy [열거형](#) 값.

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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.

구문

```
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 유틸 프로세스의 경우 호출은 Success=True 및 GameSessionId=""를 반환합니다.

예제

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =
    Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift Servers에서 onProcessTerminate() 콜백을 수신한 후 조치를 취합니다. Amazon GameLift Servers는 다음과 같은 이유로 onProcessTerminate()를 호출합니다.

- 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- [스팟 인스턴스 중단](#)으로 인해 인스턴스가 종료되는 경우

구문

```
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.

예제

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =
    Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 유효한지 확인합니다. 플레이어 세션이 검증된 후 Amazon GameLift Servers는 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(String playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

이 예제는 유효하지 않은 플레이어 세션 ID의 검증 및 거부를 포함하는 연결 요청을 처리합니다.

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
    playerId)
{
    Aws::GameLift::GenericOutcome connectOutcome =
    Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
}
```

```
else
{
    connection.Reject(connectOutcome.GetError().GetMessage());
}
}
```

RemovePlayerSession()

플레이어가 서버 프로세스와의 연결이 끊겼음을 Amazon GameLift Servers에 알립니다. 이에 따라 Amazon GameLift Servers는 플레이어 슬롯을 사용 가능한 것으로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(String playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
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

구문

```
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.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. NextToken을 생략하고 Limit 값을 10으로 설정하면, Amazon GameLift Servers가 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);
```

StartMatchBackfill()

FlexMatch를 통해 생성된 게임 세션에서 열린 슬롯에 참여할 새로운 플레이어를 찾는 요청을 보냅니다. 자세한 내용은 [FlexMatch 채우기 기능](#)을 참조하세요.

이 작업은 비동기식입니다. 새로운 플레이어가 매치되면 Amazon GameLift Servers는 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.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift Servers가 자동으로 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.

예제

```
// 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 [FlexMatch 채우기 기능](#)을 참조하세요.

구문

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

파라미터

[StopMatchBackfillRequest](#)

다음은 취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체입니다.

- 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.

예제

```
// 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 Servers Anywhere 컴퓨팅 리소스 및 Amazon GameLift Servers 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. Amazon GameLift Servers Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은 [RegisterCompute](#)을 참조하세요.

구문

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

반환 값

[the section called “GetComputeCertificateOutcome”](#)을 반환합니다.

예제

```
Aws::GameLift::GetComputeCertificateOutcome certificate =
    Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

Amazon GameLift Servers가 다른 AWS 서비스와 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

구문

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest
    request);
```


파라미터

[GetFleetRoleCredentialsRequest](#)

반환 값

[the section called “GetFleetRoleCredentialsOutcome”](#) 객체를 반환합니다.

예제

```
// 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 Servers Game Server SDK를 메모리에서 분리합니다. ProcessEnding() 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 게임 세션 이후마다 서버 프로세스를 종료하지 않는 경우, Destroy() 및 InitSDK()를 차례로 호출하여 Amazon GameLift Servers가 게임 세션을 ProcessReady()로 호스팅할 준비가 되었음을 알리기 전에 다시 초기화합니다.

구문

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

파라미터

파라미터는 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
Aws::GameLift::GenericOutcome processEndingOutcome =
    Aws::GameLift::Server::ProcessEnding();
Aws::GameLift::Server::Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess()) {
    exit(0);
}
else {
    cout << "ProcessEnding() failed. Error: " <<
        processEndingOutcome.GetError().GetErrorMessage();
    exit(-1);
}
```

Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업

해당 서버 SDK 5.x 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 페이지를 참조하세요. 또한 Unity용 Amazon GameLift Servers 플러그인을 사용하는 경우 [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 페이지를 참조하세요.

Amazon GameLift Servers용 C# 서버 SDK 5.x - 데이터 유형

Amazon GameLift Servers C# 서버 SDK 5.x 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 페이지를 참조하세요. 또한 Unity용 Amazon GameLift

Servers 플러그인을 사용하는 경우 [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#) 페이지를 참조하세요.

[Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)

데이터 타입

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)
- [AttributeValue](#)
- [AwsStringOutcome](#)
- [GenericOutcome](#)
- [MetricsManagerOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [PlayerSessions](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [AwsDateTimeOutcome](#)

- [GameLiftError](#)
- [Enums](#)

LogParameters

이 데이터 유형을 사용하여 게임 세션이 종료된 후 게임 서버가 Amazon GameLift Servers에 업로드할 게임 세션 중에 생성된 파일을 식별합니다. 게임 서버는 [ProcessReady\(\)](#) 호출을 통해 LogParameters to Amazon GameLift Servers에 전달합니다.

속성	설명
LogPaths	Amazon GameLift Servers가 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

MetricsParameters

이 데이터 유형을 사용하여 게임 서버에 대한 지표 수집 및 충돌 보고를 구성합니다. 게임 서버는 [InitMetrics\(\)](#) 호출을 통해 MetricsParameters를 Amazon GameLift Servers에 전달합니다.

속성	설명
StatsdHost	지표 수집을 위한 StatsD 서버의 호스트 이름 또는 IP 주소.

	유형: String 필수 항목 여부: 아니요
StatsdPort	지표 수집을 위한 StatsD 서버의 포트 번호. 유형: Integer 필수 항목 여부: 아니요
CrashReporterHost	충돌 리포터 서버의 호스트 이름 또는 IP 주소. 유형: String 필수 항목 여부: 아니요
CrashReporterPort	충돌 리포터 서버의 포트 번호. 유형: Integer 필수 항목 여부: 아니요
FlushIntervalMs	지표 데이터 플러시에 대한 밀리초 단위의 간격. 유형: Integer 필수 항목 여부: 아니요
MaxPacketSize	서버로 전송되는 지표 패킷의 최대 크기(바이트 단위). 유형: Integer 필수 항목 여부: 아니요

ProcessParameters

이 데이터 유형에는 [ProcessReady\(\)](#) 호출에서 Amazon GameLift Servers로 보낸 파라미터 집합이 포함됩니다.

속성	설명

LogParameters	게임 세션 로그 파일의 디렉터리 경로 목록을 포함하는 객체입니다. 유형: <code>Aws::GameLift::Server::LogParameters</code> 필수 항목 여부: 예
OnHealthCheck	Amazon GameLift Servers가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출합니다. 이 함수를 호출한 후 Amazon GameLift Servers는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 Amazon GameLift Servers는 프로세스를 비정상 상태로 기록합니다. 유형: <code>void OnHealthCheckDelegate()</code> 필수 항목 여부: 예
OnProcessTerminate	Amazon GameLift Servers가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수의 이름입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답한 후 서버 프로세스를 종료합니다. 유형: <code>void OnProcessTerminateDelegate()</code> 필수 항목 여부: 예

OnStartGameSession

Amazon GameLift Servers가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응답으로 이 함수를 호출합니다. 콜백 함수가 [GameSession](#) 객체를 가져옵니다.

유형: void OnStartGameSession
Delegate([GameSession](#))

필수 항목 여부: 예

OnUpdateGameSession

Amazon GameLift Servers가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수의 이름입니다. 매치 채우기 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 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	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. <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code>

속성	설명
	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 Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: Integer 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: String 필수 항목 여부: 아니요

속성	설명
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>.amazon.gamelift.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 Servers Anywhere 서버와 Amazon GameLift Servers 서비스 간의 연결을 유지하는데 사용되는 정보입니다. 이 정보는 [InitSDK\(\)](#)에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift Servers 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용합니다.

속성	설명
WebSocketUrl	Amazon GameLift Servers Anywhere의 일부로 RegisterCompute 할 때 GameLiftServerSdkEndpoint 가 반환됩니다. 유형: String 필수 항목 여부: 예
ProcessId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다.

속성	설명
HostId	유형: String 필수 항목 여부: 예
	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
	Amazon GameLift Servers에서 생성한 인증 토큰으로, Amazon GameLift Servers에 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 GameLift Servers에 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 GameLift Servers에서 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 GameLift Servers는 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 Attribute Value를 참조하세요. 유형: Dictionary<string, Attribute Value> 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 유형은 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. <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> . <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 생성을(를) 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	결과와 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 Servers 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 유형은 게임 서버에 다른 AWS 리소스에 대한 제한된 액세스 권한을 부여합니다. 자세한 내용은 단원을 참조하십시오 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#).

속성	설명
RoleArn	리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon AWS 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 발생하는 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 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

MetricsManagerOutcome

[InitMetrics\(\)](#) 호출 결과를 나타냅니다. 성공 시 MetricsManager 인스턴스를 포함하거나 실패 시 오류 정보를 포함합니다.

결과	지표를 수집하고 보고하기 위한 MetricsManager 인스턴스입니다. 유형: MetricsManager 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool

	필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

DescribePlayerSessionsResult

속성	설명
NextToken	결과와 다음 페이지의 시작을 나타내는 토큰입니다. 결과 집합의 시작을 지정하려면 값을 제공하지 않습니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다.

속성	설명
	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 발생하는 오류입니다. 유형: the section called “GameLiftError” 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

속성	설명
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
포트	유형: int 필수 항목 여부: 예
Status	유형: PlayerSessionStatus 열거형 . 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.

속성	설명
	유형: the section called “StartMatchBackfillResult” 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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.

속성	설명
Error	유형: bool
	필수 항목 여부: 예
	작업이 실패한 경우 발생하는 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.

속성	설명
	유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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

속성	설명
SecretAccessKey	인증에 대한 비밀 액세스 키 ID를 지정합니다. 유형: string 필수 항목 여부: 아니요
SessionToken	AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하기 위한 토큰입니다. 유형: string 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 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.

속성	설명
	유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

GameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: GameLiftErrorType 열거형 . 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: string 필수 항목 여부: 아니요
ErrorCode	오류 유형 이름입니다. 유형: string 필수 항목 여부: 아니요

Enums

Amazon GameLift Servers용 서버 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 서비스에 대한 호출이 실패했습니다.
- LOCAL_CONNECTION_FAILED - Amazon GameLift Servers에 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 Servers 서버 또는 클라이언트는 이미 Initialize()를 사용하여 초기화되었습니다.
- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALIZED - Amazon GameLift Servers 클라이언트가 초기화되지 않았습니다.
- GAMELIFT_SERVER_NOT_INITIALIZED - Amazon GameLift Servers 서버가 초기화되지 않았습니다.
- GAME_SESSION_ENDED_FAILED - Amazon GameLift Servers용 서버 SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY - Amazon GameLift Servers 서버 게임 세션이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED - Amazon GameLift Servers용 서버 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 Servers 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 Servers용 서버 SDK가 서비스에 연락하여 프로세스가 종료되었음을 보고할 수 없습니다.
- `PROCESS_NOT_ACTIVE` - 서버 프로세스가 아직 활성화되지 않았고, `GameSession`에 바인딩되지 않았으며, `PlayerSessions`를 수락하거나 처리할 수 없습니다.
- `PROCESS_NOT_READY` - 서버 프로세스를 아직 활성화할 준비가 되지 않았습니다.
- `PROCESS_READY_FAILED` – Amazon GameLift Servers용 서버 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 보내는데 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

Amazon GameLift Servers용 C# 서버 SDK 5.x - 데이터 유형

주제

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [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.

예제

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitMetrics()

서버 성능 데이터를 수집하고 보고하기 위한 지표 시스템을 초기화합니다. 최상의 결과를 얻으려면 [InitSDK\(\)](#) 전에 이 메서드를 호출하여 SDK 초기화 중에 지표 수집을 활성화합니다.

구문

```
MetricsManagerOutcome InitMetrics();  
MetricsManagerOutcome InitMetrics(MetricsParameters metricsParameters);
```

파라미터

[MetricsParameters](#) (선택 사항)

지표 수집을 위한 구성 파라미터입니다. 제공되지 않은 경우 환경 변수로 재정의할 수 있는 기본값을 사용합니다.

반환 값

성공하면 MetricsManager 인스턴스를 포함하는 [the section called “MetricsManagerOutcome”](#) 객체를 반환합니다. 실패하면 오류 메시지를 반환합니다.

예제

```
// Initialize with default parameters (uses environment variables if available)  
var outcome = GameLiftServerAPI.InitMetrics();  
if (outcome.Success) {  
    var metricsManager = outcome.Result;  
} else {  
    Console.WriteLine("Failed to initialize metrics: " + outcome.Error.ErrorMessage);  
}
```



```
// Initialize with custom parameters
var metricsParams = new MetricsParameters("localhost", 8125, "crash-host", 9999, 1000,
    1024);
var customOutcome = GameLiftServerAPI.InitMetrics(metricsParams);
if (customOutcome.Success) {
    var metricsManager = customOutcome.Result;
}
```

InitSDK()

관리형 EC2 플릿의 Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출합니다. 이 메서드는 호스트 환경에서 서버 파라미터를 읽어 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다. 맥등성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

구문

```
GenericOutcome InitSDK();
```

반환 값

성공하는 경우, 서버 프로세스가 [ProcessReady\(\)](#)를 호출할 준비가 되었음을 나타내는 InitSdkOutcome 객체를 반환합니다.

예제

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

Anywhere 플릿용 Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출합니다. 이 메서드는 명시적인 서버 파라미터를 통해 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다. 맥등성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

구문

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

파라미터

ServerParameters

Amazon GameLift Servers Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 구성합니다.

- 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Servers Anywhere 컴퓨팅이 포함된 Amazon GameLift Servers 플릿의 ID입니다.
- Amazon GameLift Servers 작업을 통해 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.

예제

```
//Define the server parameters
string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
string processId = "PID1234";
string fleetId = "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
string hostId = "HardwareAnywhere";
string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
ServerParameters serverParameters =
    new ServerParameters(websocketUrl, processId, hostId, fleetId, authToken);
```

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. [InitSDK\(\)](#) 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

구문

```
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.

예제

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 종료 중임을 Amazon GameLift Servers에 알립니다. 활성 게임 세션의 종료 등 다른 모든 정리 작업 이후 및 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.

구문

```
GenericOutcome ProcessEnding()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화 후 onStartGameSession() 콜백 함수의 일부로 호출되어야 합니다.

구문

```
GenericOutcome ActivateGameSession()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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.

구문

```
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.

예제

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.

구문

```
AwsStringOutcome GetGameSessionId()
```

반환 값

성공하면 게임 세션 ID를 [the section called “AwsStringOutcome”](#) 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

예제

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift Servers에서 onProcessTerminate() 콜백을 수신한 후 이 작업을 수행합니다. 다음과 같은 이유로 Amazon GameLift Servers가 onProcessTerminate()를 호출합니다.

- 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- [스팟 인스턴스 중단](#)으로 인해 인스턴스가 종료되는 경우

구문

```
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 -0000Z는 6373559680000000000 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예제

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 유효한지 확인합니다. 플레이어 세션이 검증된 후 Amazon GameLift Servers는 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(String playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 GameLift가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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 끊겼음을 Amazon GameLift Servers에 알립니다. 이에 따라 Amazon GameLift Servers는 플레이어 슬롯을 사용 가능한 것으로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
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.

구문

```
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.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. NextToken을 생략하고 Limit 값을 10으로 설정하면, Amazon GameLift Servers가 요청과 일치하는 처음 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 [FlexMatch 채우기 기능](#)을 참조하세요.

이 작업은 비동기식입니다. 새로운 플레이어가 매치되면 Amazon GameLift Servers는 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](#)

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.

예제

```
// 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 [FlexMatch 채우기 기능](#)을 참조하세요.

구문

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

파라미터

[StopMatchBackfillRequest](#)

종단하려는 매치메이킹 티켓에 대한 세부 정보를 제공하는 StopMatchBackfillRequest 객체입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
// 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 Servers Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은 [RegisterCompute](#)을 참조하세요.

구문

```
GetComputeCertificateOutcome GetComputeCertificate();
```

반환 값

다음에 포함하는 `GetComputeCertificateResponse` 객체를 반환합니다.

- `CertificatePath`: 컴퓨팅 리소스에 있는 TLS 인증서의 경로입니다. Amazon GameLift Servers 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 이름입니다.

예제

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

Amazon GameLift Servers가 다른 AWS 서비스와 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

구문

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest request);
```

파라미터

[GetFleetRoleCredentialsRequest](#)

AWS 리소스에 대한 제한된 액세스를 게임 서버로 확장하는 역할 자격 증명입니다.

반환 값

[the section called “GetFleetRoleCredentialsOutcome”](#) 객체를 반환합니다.

예제

```
// 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 Servers Game Server SDK를 메모리에서 분리합니다. ProcessEnding() 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 게임 세션 이후마다 서버 프로세스를 종료하지 않는 경우, Destroy() 및 InitSDK()를 차례로 호출하여 Amazon GameLift Servers가 게임 세션을 ProcessReady()로 호스팅할 준비가 되었음을 알리기 전에 다시 초기화합니다.

구문

```
GenericOutcome Destroy()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
// Operations to end game sessions and the server process
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();

// Shut down and destroy the instance of the GameLift Game Server SDK
GenericOutcome destroyOutcome = GameLiftServerAPI.Destroy();

// Exit the process with success or failure
if (processEndingOutcome.Success)
{
    Environment.Exit(0);
}
else
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
        processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
```

Amazon GameLift Servers용 Go 서버 SDK - 작업

해당 서버 SDK 5.x 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

GameLiftServerAPI.go는 Go 서버 SDK 작업을 정의합니다.

Amazon GameLift Servers용 Go 서버 SDK - 데이터 유형

해당 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

GameLiftServerAPI.go는 Go 서버 SDK 작업을 정의합니다.

[Amazon GameLift Servers용 Go 서버 SDK - 작업](#)

데이터 타입

- [LogParameters](#)
- [MetricsParameters](#)

- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Player](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)

LogParameters

Amazon GameLift Servers가 게임 세션 종료 후 업로드하고 저장하기를 원하는 게임 세션 중에 생성된 파일을 식별하는 객체입니다. 게임 서버는 [ProcessReady\(\)](#) 호출 시 ProcessParameters 객체의 일부로 LogParameters를 Amazon GameLift Servers에 제공합니다.

속성	설명
LogPaths	Amazon GameLift Servers가 향후 사용을 위해 저장하도록 하려는 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

MetricsParameters

지표 시스템을 초기화하기 위해 구성 파라미터를 전달하는 객체입니다. 이 구성은 StatsD 보고, 충돌 보고 및 지표 처리 동작을 설정하는 데 사용됩니다. 게임 서버는 [InitMetrics\(\)](#) 호출의 일부로 MetricsParameters를 Amazon GameLift Servers에 제공합니다.

속성	설명
StatsdHost	지표 보고를 위한 StatsD 서버 호스트(예: "localhost"). 유형: string 필수 항목 여부: 예
StatsdPort	지표 보고를 위한 StatsD 서버 포트(예: 8125). 유형: int 필수 항목 여부: 예
CrashReporterHost	충돌 추적 및 프로세스 모니터링을 위한 충돌 리포터 호스트입니다. 유형: string 필수 항목 여부: 예
CrashReporterPort	충돌 추적 및 프로세스 모니터링을 위한 충돌 리포터 포트입니다. 유형: int 필수 항목 여부: 예
FlushIntervalMs	밀리초 단위의 지표 플러시 간격. 지표가 StatsD로 전송되는 빈도를 제어합니다. 유형: int 필수 항목 여부: 예
MaxPacketSize	바이트 단위의 지표에 대한 최대 패킷 크기. StatsD로 전송되는 UDP 패킷의 크기를 제한합니다. 유형: int 필수 항목 여부: 예

ProcessParameters

서버 프로세스와 Amazon GameLift Servers 간의 통신을 설명하는 객체입니다. 서버 프로세스는 [ProcessReady\(\)](#)를 호출하여 이 정보를 Amazon GameLift Servers에 제공합니다.

속성	설명
LogParameters	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon GameLift Servers는 나중에 액세스할 수 있도록 파일을 복사하고 저장합니다. 유형: LogParameters 필수 항목 여부: 아니요
OnHealthCheck	Amazon GameLift Servers가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출하고 60초 동안 응답을 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon GameLift Servers는 서버 프로세스를 정상이 아닌 것으로 기록합니다. 유형: OnHealthCheck func() bool 필수 항목 여부: 아니요
OnProcessTerminate	Amazon GameLift Servers가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 ProcessEnding() 호출로 응답한 후 서버 프로세스를 종료합니다. 유형: OnProcessTerminate func() 필수 항목 여부: 예
OnStartGameSession	Amazon GameLift Servers가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수입니다. 매치 채우기 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 이 함수를 호출합니다. GameSession 객체, 상태 업데이트(updateReason) 및 매치 채우기 티켓 ID를 전달합니다.

유형: `OnStartGameSession func (model.GameSession )`

필수 항목 여부: 예

OnUpdateGameSession

Amazon GameLift Servers가 업데이트된 게임 세션 정보를 전달하기 위해 간접 호출하는 콜백 함수입니다. 매치 채우기 요청이 처리된 후 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 이 함수를 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	GameSession 객체입니다. GameSession 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 유형: <code>GameSession GameSession()</code> 필수 항목 여부: 예
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: <code>UpdateReason UpdateReason()</code>

속성	설명
	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. <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> 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

속성	설명
	필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: Integer 필수 항목 여부: 아니요
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift Servers 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 Servers Anywhere 서버와 Amazon GameLift Servers 서비스 간의 연결을 유지하는데 사용되는 정보입니다. 이 정보는 [InitSDK\(\)](#)에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift Servers 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용합니다.

속성	설명
WebSocket URL	Amazon GameLift Servers Anywhere 컴퓨팅 리소스에 대해 RegisterCompute를 수행하면 GameLiftServerSdkEndpoint Amazon GameLift Servers가 반환됩니다. 유형: string 필수 항목 여부: 예
ProcessID	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: string 필수 항목 여부: 예

속성	설명
HostID	새 서버 프로세스를 호스팅하는 컴퓨팅 리소스의 고유 식별자입니다. HostID는 컴퓨터를 등록할 때 사용되는 ComputeName 입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: string 필수 항목 여부: 예
FleetID	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: string 필수 항목 여부: 예
AuthToken	Amazon GameLift Servers에서 생성한 인증 토큰으로, Amazon GameLift Servers에 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 GameLift Servers에 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 Amazon GameLift Servers에서 값을 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. Amazon GameLift Servers는 매치가 성사된 후 팀 정보를 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: <code>map[string] int</code> 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: <code>map[string] AttributeValue</code> 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: <code>String</code> 필수 항목 여부: 아니요
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: <code>String</code> 필수 항목 여부: 아니요

DescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon GameLift Servers로 [DescribePlayerSessions\(\)](#) 호출을 통해 이 정보를 제공합니다.

속성	설명
GameSessionID	고유한 게임 세션 식별자입니다. 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 이 파라미터를 사용합니다. 게임 세션 ID 형식은 <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> 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(를) 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 연결되어 있습니다. - COMPLETED - 플레이어 연결이 끊어졌습니다. - TIMEDOUT - 플레이어 세션 요청이 수신되었지만 플레이어가 제한 시간(60 초) 이내에 연결하지 않았거나 확인되지 않았습니다. 유형: String

속성	설명
	필수 항목 여부: 아니요
NextToken	결과와 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\(\)](#) 호출을 통해 이 정보를 Amazon GameLift Servers 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 서버로 확장하는 역할 자격 증명입니다. 자세한 내용은 단원을 참조하십시오 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#).

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

[Amazon GameLift Servers용 Go 서버 SDK - 데이터 유형](#)

작업

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitMetricsFromEnvironment\(\)](#)
- [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

```
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.

예제

```
version, err := server.GetSdkVersion()
```

InitMetrics()

Amazon GameLift Servers SDK에 대한 지표 수집을 초기화합니다. 이 메서드는 서버 성능 및 상태를 모니터링하는 데 도움이 되는 지표 보고를 설정합니다. [InitSDK\(\)](#) 이후 [ProcessReady\(\)](#) 이전에 이 메서드를 호출합니다.

구문

```
func InitMetrics() error
```

```
func InitMetrics(metricsParameters MetricsParameters) error
```

파라미터

MetricsParameters(선택 사항)

지표 수집을 구성하는 MetricsParameters 객체입니다. 제공되지 않은 경우 기본 지표 구성이 사용됩니다. MetricsParameters 구조는 다음 필드를 포함합니다.

- StatsdHost - StatsD 서버의 호스트 이름 또는 IP 주소.
- StatsdPort - StatsD 서버의 포트 번호.
- CrashReporterHost - 충돌 리포터 서비스의 호스트 이름 또는 IP 주소.
- CrashReporterPort - 충돌 리포터 서비스의 포트 번호.
- FlushIntervalMs - 지표 데이터 플러시에 대한 밀리초 단위의 간격.
- MaxPacketSize - 바이트 단위의 지표 패킷 최대 크기.

MetricsParameters 구조에 대한 자세한 내용은 [C# 서버 SDK 5.x - 데이터 유형](#)을 참조하세요.

반환 값

성공하면 지표 수집이 성공적으로 초기화되었음을 나타내는 nil 오류가 반환됩니다.

예제

기본 구성으로 지표 초기화:

```
err := server.InitMetrics()
```

사용자 지정 구성으로 지표 초기화:

```
metricsParams := MetricsParameters{
    StatsdHost:      "localhost",
    StatsdPort:      8125,
    CrashReporterHost: "localhost",
    CrashReporterPort: 9125,
    FlushIntervalMs:  5000,
    MaxPacketSize:    1024,
}
```

```
err := server.InitMetrics(metricsParams)
```

InitMetricsFromEnvironment()

환경 변수에서 구성을 사용하여 Amazon GameLift Servers SDK에 대한 지표 수집을 초기화합니다. 이 메서드는 런타임 환경에서 파생된 기본 설정을 사용하여 지표 보고를 설정합니다.

[InitSDK\(\)](#) 이후 [ProcessReady\(\)](#) 이전에 이 메서드를 호출합니다.

구문

```
func InitMetricsFromEnvironment() error
```

반환 값

성공하면 지표 수집이 환경 구성을 사용하여 성공적으로 초기화되었음을 나타내는 nil 오류가 반환됩니다.

예제

```
err := server.InitMetricsFromEnvironment()
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출합니다. 이 메서드는 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다.

구문

```
func InitSDK(params ServerParameters) error
```

파라미터

[ServerParameters](#)

Amazon GameLift Servers Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 구성합니다.

- 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.

- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Servers Anywhere 컴퓨팅이 포함된 Amazon GameLift Servers 플릿의 ID입니다.
- Amazon GameLift Servers 작업을 통해 생성된 인증 토큰입니다.

Amazon GameLift Servers 관리형 EC2 플릿에서 게임 서버를 초기화하려면 파라미터 없이 `ServerParameters` 객체를 구성합니다. 이 호출을 통해, Amazon GameLift Servers 에이전트는 컴퓨팅 환경을 설정하고 자동으로 Amazon GameLift Servers 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.

예제

Amazon GameLift Servers Anywhere 예제

```
//Define the server parameters
serverParameters := ServerParameters {
    WebSocketURL: "wss://us-west-1.api.amazongamelift.com",
    ProcessID: "PID1234",
    HostID: "HardwareAnywhere",
    FleetID: "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa",
    AuthToken: "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

Amazon GameLift Servers 관리형 EC2 예제

```
//Define the server parameters
serverParameters := ServerParameters {}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다.

[InitSDK\(\)](#) 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

구문

```
func ProcessReady(param ProcessParameters) error
```

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 Servers 서비스가 간접 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로를 포함하는 [LogParameters](#) 데이터 유형입니다.

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다. 메서드가 성공한 경우 nil를 반환합니다.

예제

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 종료 중임을 Amazon GameLift Servers에 알립니다. 활성 게임 세션의 종료 등 다른 모든 정리 작업 이후 및 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.

구문

```
func ProcessEnding() error
```

반환 값

0 오류 코드 또는 정의된 오류 코드를 반환합니다.

예제

```

// operations to end game sessions and the server process
defer func() {
    err := server.ProcessEnding()
    server.Destroy()
    if err != nil {
        fmt.Println("ProcessEnding() failed. Error: ", err)
        os.Exit(-1)
    } else {
        os.Exit(0)
    }
}

```

ActivateGameSession()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화 후 `onStartGameSession()` 콜백 함수의 일부로 호출됩니다.

구문

```
func ActivateGameSession() error
```

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다.

예제

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.

구문

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

파라미터

playerSessionCreationPolicy

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.

예제

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.

구문

```
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.

예제

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift Servers에서 onProcessTerminate() 콜백을 수신한 후 이 작업을 수행합니다. 다음과 같은 이유로 Amazon GameLift Servers가 onProcessTerminate()를 호출합니다.

- 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우

- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- [스팟 인스턴스 중단](#)으로 인해 인스턴스가 종료되는 경우

구문

```
func GetTerminationTime() (int64, error)
```

반환 값

성공하면 서버 프로세스가 종료될 예정인 타임스탬프(Epoch 초) 및 nil 오류 종료를 반환합니다. 값은 종료 시간이며, 0001 00:00:00에서 경과된 틱 수로 표시됩니다. 예를 들어, 날짜 시간 값 2020-09-13 12:26:40 -0002는 637355968000000000 틱 수와 같습니다. 사용 가능한 종료 시간이 없는 경우 오류 메시지를 반환합니다.

예제

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Amazon GameLift Servers에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 유효한지 확인합니다. 플레이어 세션이 검증된 후 Amazon GameLift Servers는 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
func AcceptPlayerSession(playerSessionID string) error
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

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 끊겼음을 Amazon GameLift Servers에 알립니다. 이에 따라 Amazon GameLift Servers는 플레이어 슬롯을 사용 가능한 것으로 변경합니다.

구문

```
func RemovePlayerSession(playerSessionID string) error
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
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

구문

```
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.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. `NextToken`을 생략하고 `Limit` 값을 10으로 설정하면, Amazon GameLift Servers가 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 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 채우기 기능](#)을 참조하세요.

이 작업은 비동기식입니다. 새로운 플레이어가 매치되면 Amazon GameLift Servers는 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

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
    (result.StartMatchBackfillResult, error)
```

파라미터

[StartMatchBackfillRequest](#)

다음 정보를 전달하는 StartMatchBackfillRequest 객체입니다.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift Servers가 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 StartMatchBackfillResult 객체나 오류 메시지를 포함한 결함을 반환합니다.

예제

```
// 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 [FlexMatch 채우기 기능](#)을 참조하세요.

구문

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

파라미터

[StopMatchBackfillRequest](#)

다음은 취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체입니다.

- 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.

예제


```
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 Servers Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은 [RegisterCompute](#)를 참조하세요.

구문

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

반환 값

다음에 포함하는 GetComputeCertificateResult 객체를 반환합니다.

- **CertificatePath:** 컴퓨팅 리소스에 있는 TLS 인증서의 경로입니다. Amazon GameLift Servers 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 이름입니다.

예제

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

권한을 다른 로 확장하기 위해 생성한 서비스 역할 자격 증명을 검색합니다 AWS 서비스 Amazon GameLift Servers. 이러한 자격 증명을 통해 게임 서버는 AWS 리소스를 사용할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#) 섹션을 참조하세요.

구문

```
func GetFleetRoleCredentials(  
    req request.GetFleetRoleCredentialsRequest,  
) (result.GetFleetRoleCredentialsResult, error) {  
    return srv.getFleetRoleCredentials(&req)  
}
```

파라미터

[GetFleetRoleCredentialsRequest](#)

AWS 리소스에 대한 제한된 액세스를 게임 서버로 확장하는 역할 자격 증명입니다.

반환 값

다음에 포함하는 GetFleetRoleCredentialsResult 객체를 반환합니다.

- AssumedRoleUserArn - 사용자가 맡을 서비스 역할의 Amazon 리소스 이름(ARN)입니다.
- AssumedRoleId - 서비스 역할이 속한 사용자의 ID입니다.
- AccessKeyId - 인증하고 AWS 리소스에 액세스를 제공하기 위한 액세스 키 ID입니다.
- SecretAccessKey - 인증에 대한 비밀 액세스 키 ID를 지정합니다.
- SessionToken - AWS 리소스와 상호 작용하는 현재 활성 세션을 식별하는 토큰입니다.
- Expiration - 세션 자격 증명 만료될 때까지 남은 시간입니다.

예제

```
// 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 Servers Game Server SDK를 메모리에서 분리합니다. ProcessEnding() 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 게임 세션 이후마다 서버 프로세스를 종료하지 않는 경우, Destroy() 및 InitSDK()를 차례로 호출하여 Amazon GameLift Servers가 게임 세션을 ProcessReady()로 호스팅할 준비가 되었음을 알리기 전에 다시 초기화합니다.

구문

```
func Destroy() error {
    return srv.destroy()
}
```

반환 값

메서드가 실패하면 오류 메시지와 함께 오류를 반환합니다.

예제

```
// operations to end game sessions and the server process
defer func() {
    err := server.ProcessEnding()
    server.Destroy()
    if err != nil {
        fmt.Println("ProcessEnding() failed. Error: ", err)
        os.Exit(-1)
    } else {
        os.Exit(0)
    }
}
```

Amazon GameLift Servers용 C++(Unreal) 서버 SDK 5.x - 작업

Amazon GameLift Servers Unreal 서버 SDK 5.x 참조를 사용하면 Amazon GameLift Servers와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 섹션을 참조하세요. 또한 Unreal용 Amazon GameLift Servers 플러그인을 사용하는 경우 [Unreal Engine용 Amazon GameLift Servers 플러그인](#) 페이지를 참조하세요.

Note

이 주제에서는 Unreal Engine용으로 빌드할 때 사용할 수 있는 Amazon GameLift Servers C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DBUILD_FOR_UNREAL=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

Amazon GameLift Servers용 C++(Unreal) 서버 SDK 5.x - 데이터 유형

Amazon GameLift Servers Unreal 서버 SDK 5.x 참조를 사용하면 Amazon GameLift Servers와 함께 사용할 멀티플레이어 게임을 준비하는 데 도움이 됩니다. 통합 프로세스에 대한 자세한 내용은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 섹션을 참조하세요. 또한 Unreal용 Amazon GameLift Servers 플러그인을 사용하는 경우 [Unreal Engine용 Amazon GameLift Servers 플러그인](#) 페이지를 참조하세요.

Note

이 주제에서는 Unreal Engine용으로 빌드할 때 사용할 수 있는 Amazon GameLift Servers C++ API에 대해 설명합니다. 특히, 이 설명서는 `-DBUILD_FOR_UNREAL=1` 옵션을 사용하여 컴파일하는 코드에 적용됩니다.

[Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)

데이터 타입

- [FProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [FServerParameters](#)
- [FStartMatchBackfillRequest](#)
- [FPlayer](#)
- [FGameLiftDescribePlayerSessionsRequest](#)
- [FStopMatchBackfillRequest](#)
- [FAttributeValue](#)
- [FGameLiftGetFleetRoleCredentialsRequest](#)
- [FGameLiftLongOutcome](#)

- [FGameLiftStringOutcome](#)
- [FGameLiftDescribePlayerSessionsOutcome](#)
- [FGameLiftDescribePlayerSessionsResult](#)
- [FGenericOutcome](#)
- [FGameLiftPlayerSession](#)
- [FGameLiftGetComputeCertificateOutcome](#)
- [FGameLiftGetComputeCertificateResult](#)
- [FGameLiftGetFleetRoleCredentialsOutcome](#)
- [FGetFleetRoleCredentialsResult](#)
- [FGameLiftError](#)
- [Enums](#)

FProcessParameters

이 데이터 유형에는 [ProcessReady\(\)](#)에서 Amazon GameLift Servers로 보낸 파라미터 집합이 포함됩니다.

속성	설명
LogParameters	게임 세션 중에 생성되는 파일에 대한 디렉터리 경로가 있는 객체입니다. Amazon GameLift Servers는 나중에 액세스할 수 있도록 파일을 복사하고 저장합니다. 유형: TArray<FString> 필수 항목 여부: 아니요
OnHealthCheck	Amazon GameLift Servers가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출하고 60초 동안 응답을 기다립니다. 서버 프로세스가 정상이면 TRUE를 반환하고, 정상이 아니면 FALSE를 반환합니다. 응답이 반환되지 않으면 Amazon

GameLift Servers는 서버 프로세스를 정상 이 아닌 것으로 기록합니다.

이 속성은 다음과 같이 정의된 대리자 함수입니다. DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck)

유형: FOnHealthCheck

필수 항목 여부: 아니요

OnProcessTerminate

Amazon GameLift Servers가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 [ProcessEnding\(\)](#) 호출로 응답한 후 서버 프로세스를 종료합니다.

유형: FSimpleDelegate

필수 항목 여부: 예

OnStartGameSession

Amazon GameLift Servers가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응답으로 이 함수를 호출합니다. 콜백 함수가 [GameSession](#) 객체를 전달합니다.

이 속성은 다음과 같이 정의된 대리자 함수입니다. DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);

유형: FOnStartGameSession

필수 항목 여부: 예

OnUpdateGameSession	Amazon GameLift Servers가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수입니다. 매치 채우기 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 이 함수를 호출합니다. GameSession 객체, 상태 업데이트 (updateReason) 및 매치 채우기 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	GameSession 객체입니다. GameSession 객체에는 게임 세션을 설명하는 속성이 포함되어 있습니다. 유형: <code>Aws::GameLift::Server::GameSession</code> 필수 항목 여부: 아니요
UpdateReason	게임 세션이 업데이트되는 이유입니다. 유형: <code>enum class UpdateReason</code> • MATCHMAKING_DATA_UPDATED • BACKFILL_FAILED • BACKFILL_TIMED_OUT • BACKFILL_CANCELLED 필수 항목 여부: 아니요
BackfillTicketId	게임 세션 업데이트를 시도하는 채우기 티켓의 ID입니다. 유형: <code>char[]</code> 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. <code>arn:aws:gamelift:<region>::gamesession/</code>

속성	설명
	<fleet ID>/<custom ID string or idempotency token> 유형: char[] 필수 항목 여부: 아니요
이름	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. 유형: int 필수 항목 여부: 아니요
Port	게임 세션의 포트 번호입니다. Amazon GameLift Servers 게임 서버에 연결하려면 앱에 IP 주소와 포트 번호가 모두 필요합니다. 유형: int 필수 항목 여부: 아니요

속성	설명
IpAddress	게임 세션의 IP 주소입니다. Amazon GameLift Servers 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
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

속성	설명
DnsName	게임 세션을 실행하는 인스턴스에 할당된 DNS 식별자입니다. 값은 다음 형식을 사용합니다. - TLS 지원 플릿: <unique identifier>.<region identifier>.amazon.gamelift.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

FServerParameters

Amazon GameLift Servers Anywhere 서버와 Amazon GameLift Servers 서비스 간의 연결을 유지하는데 사용되는 정보입니다. 이 정보는 [InitSDK\(\)](#)에서 새 서버 프로세스를 시작할 때 사용됩니다. Amazon GameLift Servers 관리형 EC2 인스턴스에 호스팅되는 서버의 경우 빈 객체를 사용합니다.

속성	설명
websocketUrl	Amazon GameLift Servers Anywhere 컴퓨팅 리소스에 대해 RegisterCompute 를 수행하면 GameLiftServerSdkEndpoint Amazon GameLift Servers가 반환됩니다. 유형: char[] 필수 항목 여부: 예

속성	설명
processId	게임을 호스팅하는 서버 프로세스에 등록된 고유 식별자입니다. 유형: char[] 필수 항목 여부: 예
hostId	HostID는 컴퓨터를 등록할 때 사용되는 ComputeName 입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: char[] 필수 항목 여부: 예
fleetId	컴퓨팅이 등록된 플릿의 고유 식별자입니다. 자세한 내용은 RegisterCompute를 참조하세요. 유형: char[] 필수 항목 여부: 예
authToken	Amazon GameLift Servers에서 생성한 인증 토큰으로, Amazon GameLift Servers에 서버를 인증합니다. 자세한 내용은 GetComputeAuthToken을 참조하세요. 유형: char[] 필수 항목 여부: 예

FStartMatchBackfillRequest

매치메이킹 채우기 요청을 생성하는 데 사용되는 정보입니다. 게임 서버는 [StartMatchBackfill\(\)](#) 호출을 통해 이 정보를 Amazon GameLift Servers에 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 Amazon GameLift Servers에서 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다. 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. Amazon GameLift Servers는 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: TMap>FString, int32< 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 Attribute Value를 참조하세요. 유형: TMap>FString, FAttributeValue< 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 유형: std::string 필수 항목 여부: 아니요

속성	설명
Team	매치에서 플레이어가 배정되는 팀의 이름입니다. 매치메이킹 규칙 세트에 팀 이름을 정의합니다. 유형: FString 필수 항목 여부: 아니요

FGameLiftDescribePlayerSessionsRequest

검색할 플레이어 세션을 지정하는 객체입니다. 서버 프로세스는 Amazon GameLift Servers로 [DescribePlayerSessions\(\)](#) 호출을 통해 이 정보를 제공합니다.

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 필수 항목 여부: 아니요
PlayerId	플레이어의 고유 식별자입니다. 이 파라미터를 사용하여 특정 플레이어에 대한 모든 플레이어

속성	설명
	세션을 요청합니다. 플레이어 ID 생성을(를) 참조하세요. 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
NextToken	결과와 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

속성	설명
Limit	반환할 최대 결과 수입니다. 플레이어 세션 ID가 제공된 경우 이 파라미터가 무시됩니다. 유형: int 필수 항목 여부: 아니요

FStopMatchBackfillRequest

매치메이킹 채우기 요청을 취소하는 데 사용되는 정보입니다. 게임 서버는 [StopMatchBackfill\(\)](#) 호출을 통해 이 정보를 Amazon GameLift Servers 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. 유형: FString 필수 항목 여부: 예

FAttributeValue

[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 FAttributeValue 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> 필수 항목 여부: 아니요

FGameLiftGetFleetRoleCredentialsRequest

이 데이터 유형은 AWS 리소스에 대한 제한된 액세스를 게임 서버까지 확장하는 역할 자격 증명을 제공합니다. 자세한 내용은 [Amazon GameLift Servers의 IAM 서비스 역할 설정](#) 섹션을 참조하세요.

속성	설명
RoleArn	리소스에 대한 제한된 액세스를 확장하는 서비스 역할의 Amazon AWS 리소스 이름(ARN)입니다. 유형: FString 필수 항목 여부: 아니요
RoleSessionName	역할 자격 증명의 사용을 설명하는 세션 이름입니다. 유형: FString 필수 항목 여부: 아니요

FGameLiftLongOutcome

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

속성	설명
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGameLiftStringOutcome

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 rvalue 로 변환합니다. 유형: FString&& 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다.

속성	설명
	유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGameLiftDescribePlayerSessionsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “FGameLiftDescribePlayerSessionsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue 로 변환합니다. 유형: FGameLiftDescribePlayerSessionsResult&& 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGameLiftDescribePlayerSessionsResult

속성	설명
PlayerSessions	유형: TArray<FGameLiftPlayerSession> 필수 항목 여부: 예
NextToken	결과와 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 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGenericOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예

속성	설명
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGameLiftPlayerSession

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 필수 항목 여부: 예
PlayerSessionId	유형: FString 필수 항목 여부: 예
포트	유형: int

속성	설명
	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

FGameLiftGetComputeCertificateOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “FGameLiftGetComputeCertificateResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue 로 변환합니다. 유형: FGameLiftGetComputeCertificateResult&& 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예

속성	설명
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGameLiftGetComputeCertificateResult

컴퓨팅의 TLS 인증서 경로 및 컴퓨팅 호스트 이름입니다.

속성	설명
CertificatePath	유형: FString 필수 항목 여부: 예
ComputeName	유형: FString 필수 항목 여부: 예

FGameLiftGetFleetRoleCredentialsOutcome

이 데이터 유형은 작업의 결과이며 다음과 같은 속성을 가진 객체를 생성합니다.

속성	설명
Result	작업 결과입니다. 유형: the section called “FGetFleetRoleCredentialsResult” 필수 항목 여부: 아니요
ResultWithOwnership	호출 코드가 객체의 소유권을 가질 수 있도록 작업 결과를 rvalue 로 변환합니다. 유형: FGameLiftGetFleetRoleCredentialsResult&&

속성	설명
	필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “FGameLiftError” 필수 항목 여부: 아니요

FGetFleetRoleCredentialsResult

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 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.

속성	설명
	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 식별하기 위한 토큰입니다. 유형: FString 필수 항목 여부: 아니요
Success	작업의 성공 여부입니다. 유형: bool 필수 항목 여부: 예
Error	작업이 실패한 경우 발생하는 오류입니다. 유형: the section called “GameLiftError” 필수 항목 여부: 아니요

FGameLiftError

속성	설명
ErrorType	오류의 유형입니다. 유형: GameLiftErrorType 열거형. 필수 항목 여부: 아니요

속성	설명
ErrorName	오류 유형 이름입니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요
ErrorMessage	오류 메시지입니다. 유형: <code>std::string</code> 필수 항목 여부: 아니요

Enums

Amazon GameLift Servers용 서버 SDK(Unreal)에 대해 정의된 Enums는 다음과 같이 정의됩니다.

FAttributeType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

오류 유형을 나타내는 문자열 값입니다. 유효한 값으로는 다음이 포함됩니다.

- SERVICE_CALL_FAILED - AWS 서비스에 대한 호출이 실패했습니다.
- LOCAL_CONNECTION_FAILED - Amazon GameLift Servers에 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 Servers 서버 또는 클라이언트는 이미 Initialize()를 사용하여 초기화되었습니다.

- FLEET_MISMATCH - 대상 플릿이 gameSession 또는 playerSession의 플릿과 일치하지 않습니다.
- GAMELIFT_CLIENT_NOT_INITIALIZED - Amazon GameLift Servers 클라이언트가 초기화되지 않았습니다.
- GAMELIFT_SERVER_NOT_INITIALIZED - Amazon GameLift Servers 서버가 초기화되지 않았습니다.
- GAME_SESSION_ENDED_FAILED - Amazon GameLift Servers용 서버 SDK가 서비스에 연락하여 게임 세션이 종료되었음을 보고할 수 없습니다.
- GAME_SESSION_NOT_READY - Amazon GameLift Servers 서버 게임 세션이 활성화되지 않았습니다.
- GAME_SESSION_READY_FAILED - Amazon GameLift Servers용 서버 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 Servers 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 Servers용 서버 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 Servers용 서버 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.

EPlayerSessionCreationPolicy

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.

EPlayerSessionStatus

- ACTIVE
- COMPLETED
- NOT_SET
- RESERVED
- TIMEDOUT

Amazon GameLift Servers용 C++(Unreal) 서버 SDK 5.x - 데이터 유형

주제

- [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

```
FGameLiftStringOutcome GetSdkVersion();
```

반환 값

성공하면 현재 SDK 버전을 [the section called “FGameLiftStringOutcome”](#) 객체로 반환합니다. 반환된 객체는 버전 번호(예: 5.0.0)를 포함합니다. 실패하면 오류 메시지를 반환합니다.

예제

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

관리형 EC2 플릿의 Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출합니다. 이 메서드는 호스트 환경에서 서버 파라미터를 읽어 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다. 맥동성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

구문

```
FGameLiftGenericOutcome InitSDK()
```

반환 값

성공하는 경우, 서버 프로세스가 [ProcessReady\(\)](#)를 호출할 준비가 되었음을 나타내는 `InitSdkOutcome` 객체를 반환합니다.

예제

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK();
```

InitSDK()

Anywhere 플릿 또는 관리형 컨테이너 플릿용 Amazon GameLift Servers SDK를 초기화합니다. Amazon GameLift Servers와 관련된 다른 초기화가 발생하기 전에 시작 시 이 메서드를 호출합니다. 이 메서드는 명시적인 서버 파라미터를 통해 서버와 Amazon GameLift Servers 서비스 간의 통신을 설정합니다. 맥등성 토큰을 사용하므로 실패 시 이 호출을 안전하게 다시 시도할 수 있습니다.

구문

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

파라미터

[FServerParameters](#)

Amazon GameLift Servers Anywhere 플릿에서 게임 서버를 초기화하려면 다음 정보를 사용하여 `ServerParameters` 객체를 구성합니다.

- 게임 서버에 연결하는 데 사용되는 WebSocket의 URL입니다.
- 게임 서버를 호스팅하는 데 사용되는 프로세스의 ID입니다.
- 게임 서버 프로세스를 호스팅하는 컴퓨팅의 ID입니다.
- Amazon GameLift Servers Anywhere 컴퓨팅이 포함된 Amazon GameLift Servers 플릿의 ID입니다.
- Amazon GameLift Servers 작업을 통해 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.

예제

```
//Define the server parameters
FServerParameters serverParameters;
parameters.m_authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
parameters.m_fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
parameters.m_hostId = "HardwareAnywhere";
parameters.m_processId = "PID1234";
parameters.m_webSocketUrl = "wss://us-west-1.api.amazongamelift.com";

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. [InitSDK\(\)](#) 호출 후 이 메서드를 호출합니다. 이 메서드는 프로세스당 한 번만 호출해야 합니다.

구문

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

파라미터**processParameters**

다음 서버 프로세스 관련 정보를 전달하는 [FProcessParameters](#) 객체입니다.

- 게임 서버 코드에 구현되어 있는 콜백 메서드의 이름으로, 서버 프로세스와 통신할 수 있도록 Amazon GameLift Servers 서비스가 호출하는 메서드입니다.

- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 캡처 및 저장하도록 하려는 모든 게임 세션별 파일에 대한 경로입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

이 예에서는 [ProcessReady\(\)](#) 호출 및 위임 함수 구현을 모두 보여줍니다.

```
//Calling ProcessReady tells Amazon GameLift Servers this game server is ready to
//receive incoming game sessions!
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = GameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

서버 프로세스가 종료 중임을 Amazon GameLift Servers에 알립니다. 활성 게임 세션의 종료 등 다른 모든 정리 작업 이후 및 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.

구문

```
FGameLiftGenericOutcome ProcessEnding()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback before
//shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
//etc., before being shut down.
```

```
//In this case, we simply tell Amazon GameLift Servers we are indeed going to shutdown.
params->OnTerminate.BindLambda(=[]() {
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    GameLiftSdkModule->ProcessEnding();
});
```

ActivateGameSession()

Amazon GameLift Servers에 서버 프로세스가 게임 세션을 활성화했으며 이제 플레이어 연결을 수신할 준비가 되었음을 알립니다. 이 작업은 모든 게임 세션 초기화 후 onStartGameSession() 콜백 함수의 일부로 호출되어야 합니다.

구문

```
FGameLiftGenericOutcome ActivateGameSession()
```

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

이 예에서는 onStartGameSession() 위임 함수의 일부로 ActivateGameSession()이 호출되는 것을 보여줍니다.

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

UpdatePlayerSessionCreationPolicy()

현재 게임 세션의 새 플레이어 세션 수락 가능성을 업데이트합니다. 모든 새 플레이어 세션을 수락하거나 거부하도록 게임 세션을 설정할 수 있습니다.

구문

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

파라미터

playerCreationSessionPolicy

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.

예제

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 “FGameLiftError”](#)를 반환합니다.

구문

```
FGameLiftStringOutcome GetGameSessionId()
```

파라미터

이 작업에는 파라미터가 없습니다.

반환 값

성공하면 게임 세션 ID를 [the section called “FGameLiftStringOutcome”](#) 객체로 반환합니다. 실패하면 오류 메시지를 반환합니다.

게임 세션으로 활성화되지 않은 유틸 프로세스의 경우 호출은 Success=True 및 GameSessionId=""를 반환합니다.

예제

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

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift Servers에서 onProcessTerminate() 콜백을 수신한 후 조치를 취합니다. Amazon GameLift Servers는 다음과 같은 이유로 onProcessTerminate()를 호출합니다.

- 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우
- 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우
- [스팟 인스턴스 중단](#)으로 인해 인스턴스가 종료되는 경우

구문

```
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.

예제

```
AwsDateTimeOutcome TermTimeOutcome = GameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스에 연결되었고 검증이 필요함을 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 유효한지 확인합니다. 플레이어 세션이 검증된 후 Amazon GameLift Servers는 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

이 예제는 유효하지 않은 플레이어 세션 ID의 검증 및 거부를 포함하는 연결 요청을 처리합니다.

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerId, 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 끊겼음을 Amazon GameLift Servers에 알립니다. 이에 따라 Amazon GameLift Servers는 플레이어 슬롯을 사용 가능한 것으로 변경합니다.

구문

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
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

구문

```

FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
    FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)

```


파라미터

FGameLiftDescribePlayerSessionsRequest

검색할 플레이어 세션을 설명하는 [the section called “FGameLiftDescribePlayerSessionsRequest”](#) 객체입니다.

반환 값

성공하는 경우, 요청 파라미터에 적합한 플레이어 세션 객체 집합이 들어 있는 [the section called “FGameLiftDescribePlayerSessionsOutcome”](#) 객체를 반환합니다.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 요청하는 예입니다. NextToken을 생략하고 Limit 값을 10으로 설정하면, Amazon GameLift Servers가 요청과 일치하는 처음 10개의 플레이어 세션 레코드를 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 채우기 기능](#)을 참조하세요.

이 작업은 비동기식입니다. 새로운 플레이어가 매치되면 Amazon GameLift Servers는 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 StartMatchBackfillRequest 객체입니다.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift Servers가 자동으로 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.

예제

```
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 [FlexMatch 채우기 기능](#)을 참조하세요.

구문

```

FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
&stopBackfillRequest);

```

파라미터

[FStopMatchBackfillRequest](#)

다음은 취소할 매치메이킹 티켓을 식별하는 StopMatchBackfillRequest 객체입니다.

- 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.

예제

```

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 Servers Anywhere 컴퓨팅 리소스 및 Amazon GameLift Servers 간의 네트워크 연결을 암호화하는 데 사용되는 TLS 인증서의 경로를 검색합니다. Amazon GameLift Servers Anywhere 플릿에 컴퓨팅 디바이스를 등록할 때 인증서 경로를 사용할 수 있습니다. 자세한 내용은 [RegisterCompute](#)를 참조하세요.

구문

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

반환 값

다음에 포함하는 GetComputeCertificateResponse 객체를 반환합니다.

- CertificatePath: 컴퓨팅 리소스에 있는 TLS 인증서의 경로입니다.
- HostName: 컴퓨팅 리소스의 호스트 이름입니다.

예제

```

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()

Amazon GameLift Servers가 다른 AWS 서비스와 상호 작용할 수 있도록 승인하는 IAM 역할 자격 증명을 검색합니다. 자세한 내용은 [Amazon GameLift Servers 호스팅 게임 서버를 다른 AWS 리소스에 연결](#) 단원을 참조하십시오.

구문

```

FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)

```

파라미터

[FGameLiftGetFleetRoleCredentialsRequest](#)

반환 값

[the section called “FGameLiftGetFleetRoleCredentialsOutcome”](#) 객체를 반환합니다.

예제

```

FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)
{

```

```

#if WITH_GAMELIFT
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest sdkRequest;
sdkRequest.SetRoleArn(TCHAR_TO_UTF8(*request.m_roleArn));
sdkRequest.SetRoleSessionName(TCHAR_TO_UTF8(*request.m_roleSessionName));

auto outcome = Aws::GameLift::Server::GetFleetRoleCredentials(sdkRequest);

if (outcome.IsSuccess()) {
    auto& outres = outcome.GetResult();
    FGameLiftGetFleetRoleCredentialsResult result;
    result.m_assumedUserRoleArn = UTF8_TO_TCHAR(outres.GetAssumedUserRoleArn());
    result.m_assumedRoleId = UTF8_TO_TCHAR(outres.GetAssumedRoleId());
    result.m_accessKeyId = UTF8_TO_TCHAR(outres.GetAccessKeyId());
    result.m_secretAccessKey = UTF8_TO_TCHAR(outres.GetSecretAccessKey());
    result.m_sessionToken = UTF8_TO_TCHAR(outres.GetSessionToken());
    result.m_expiration = FDateTime::FromUnixTimestamp(outres.GetExpiration());
    return FGameLiftGetFleetRoleCredentialsOutcome(result);
}
else {
    return FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftError(outcome.GetError()));
}
#else
return
FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftGetFleetRoleCredentialsResult());
#endif
}

```

Destroy()

Amazon GameLift Servers Game Server SDK를 메모리에서 분리합니다. ProcessEnding() 이후 및 프로세스를 종료하기 전에 이 메서드를 호출하는 것이 가장 좋습니다. Anywhere 플릿을 사용 중이고 게임 세션 이후마다 서버 프로세스를 종료하지 않는 경우, Destroy() 및 InitSDK()를 차례로 호출하여 Amazon GameLift Servers가 게임 세션을 ProcessReady()로 호스팅할 준비가 되었음을 알리기 전에 다시 초기화합니다.

구문

```
FGameLiftGenericOutcome Destroy();
```

파라미터

파라미터는 없습니다.

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
// First call ProcessEnding()
FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule->ProcessEnding();

// Then call Destroy() to free the SDK from memory
FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
}
else {
    if (!processEndingOutcome.IsSuccess()) {
        const FGameLiftError& error = processEndingOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
    if (!destroyOutcome.IsSuccess()) {
        const FGameLiftError& error = destroyOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
}
```

Amazon GameLift Servers용 서버 SDK 버전 4 이하

본 참조는 Amazon GameLift Servers용 서버 SDK 이하 버전에 대해 문서화합니다. 서버 SDK는 게임 서버가 Amazon GameLift Servers 서비스와 통신하는 데 사용하는 핵심 기능을 제공합니다. 예를 들어 게임 서버는 서비스로부터 게임 세션을 시작 및 중지하라는 프롬프트를 수신하고 서비스에 대한 정기적인 게임 세션 상태 업데이트를 제공합니다. 호스팅을 위해 배포하기 전에 게임 서버를 서버 SDK와 통합합니다.

서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 사용할 사용자 지정 멀티플레이어 게임 서버를 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

SDK 버전 4.0.2의 경우 [공식 GitHub 릴리스](#)에서 다운로드할 수 있습니다. GameLiftLocal.jar를 포함한 전체 SDK 패키지는 GameLift-CSharp-ServerSDK-4.0.2.zip 아티팩트에 포함되어 있습니다.

Amazon GameLift Servers용 서버 SDK의 최신 메이저 버전은 5.x입니다. 다음 호스팅 기능을 사용하려면 버전 5.x를 업데이트해야 합니다.

- Amazon GameLift Servers Anywhere
- Amazon GameLift Servers 관리형 컨테이너
- Unreal Engine 및 Unity용 Amazon GameLift Servers 플러그인

주제

- [Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업](#)
- [Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업](#)
- [Amazon GameLift Servers용 서버 SDK\(Unreal\) - 작업](#)

Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업

해당 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#) 페이지를 참조하세요.

Amazon GameLift Servers 4.x용 C++ 서버 SDK - 데이터 유형

해당 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 데이터 유형](#) 페이지를 참조하세요.

이 API는 `GameLiftServerAPI.h`, `LogParameters.h` 및 `ProcessParameters.h`에 정의되어 있습니다.

[Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업](#)

`DescribePlayerSessionsRequest`

이 데이터 유형은 검색할 플레이어 세션을 지정하는 데 사용됩니다. 다음과 같이 사용할 수 있습니다.

- 특정 플레이어 세션을 요청하려면 `PlayerSessionId`를 제공합니다.
- 지정한 게임 세션의 모든 플레이어 세션을 요청하려면 `GameSessionId`를 제공합니다.
- 지정한 플레이어의 모든 플레이어 세션을 요청하려면 `PlayerId`를 제공합니다.

플레이어 세션이 대량일 경우 페이지 매김 파라미터를 사용하여 결과를 순차적 블록으로 검색합니다.

내용

`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 `NextToken`과 함께 사용합니다. 플레이어 세션 ID가 지정된 경우 이 파라미터가 무시됩니다.

유형: 정수

필수 항목 여부: 아니요

NextToken

결과와 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마세요. 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](#)을(를) 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 GameLift Servers가 업로드 및 저장할 파일을 식별합니다. 이 정보는 [ProcessReady\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

logPaths

Amazon GameLift Servers가 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 경로는 c:\game\MyGame\sessionLogs(Windows 인스턴스) 또는 /local/game/MyGame/sessionLogs(Linux 인스턴스)가 됩니다.

유형: std::vector<std::string>

필수 여부: 아니요

ProcessParameters

이 데이터 유형에는 [ProcessReady\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스로 보낸 파라미터 집합이 포함됩니다.

내용

포트

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 여부: 아니요

onStartGameSession

Amazon GameLift Servers 서비스가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 [GameSession](#) 객체를 전달합니다(Amazon GameLift Servers Service API Reference에서 정의함).

유형: `const std::function<void(Aws::GameLift::Model::GameSession)>`
`onStartGameSession`

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift Servers 서비스가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수의 이름입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 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 Servers 서비스가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출합니다. 이 함수를 호출한 후 Amazon GameLift Servers는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상 상태로 기록합니다.

유형: `std::function<bool()>` `onHealthCheck`

필수 항목 여부: 아니요

onUpdateGameSession

Amazon GameLift Servers 서비스가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수의 이름입니다. [매치 채우기](#) 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하

기 위해 Amazon GameLift Servers에서 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 유형은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 해당 정보는 [StartMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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 작업](#)을 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 Servers 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 GameLift Servers는 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. 해당 정보는 [StopMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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

Amazon GameLift Servers 4.x용 C++ 서버 SDK - 데이터 유형

주제

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [Destroy\(\)](#)

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되어 있고 유효성 검사가 필요함을 Amazon GameLift Servers 서비스에 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 게임 세션에서 플레이어 슬롯을 예약했는지 여부를 확인합니다. 검증되면 Amazon GameLift Servers가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```


파라미터

playerSessionId

AWS SDK Amazon GameLift Servers API 작업 [CreatePlayerSession](#) 호출에 대한 응답으로 Amazon GameLift Servers 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.

예제

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 Servers 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.

예제

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 객체는 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 데이터 유형과 구조가 동일합니다.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. `NextToken`을 생략하고 `Limit` 값을 10으로 설정하면, Amazon GameLift Servers가 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.

예제

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

플릿 및 해당 인스턴스와 연결된 pem 인코딩 TLS 인증서의 파일 위치를 검색합니다. 인증서 구성이 GENERATED로 설정된 새 플릿을 생성할 때 인증서를 AWS Certificate Manager 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.

예제

```
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.

예제

```
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 Servers 서비스로부터 `onProcessTerminate()` 콜백을 받은 후 이 작업을 수행합니다. 다음과 같은 이유로 Amazon GameLift Servers가 `onProcessTerminate()`를 호출할 수 있습니다. (1) 서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우, (2) 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우 또는 (3) [스팟 중단](#)으로 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 종료](#)에 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.

예제

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift Servers 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.

예제

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =  
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Amazon GameLift Servers 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 메시드가 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.

예제

```
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 Servers 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 [InitSDK\(\)](#)를 간접 호출하고, 서버 프로세스가 게임 세션을 호스트하기 전에 수행해야 하는 설정 작업을 완료한 후에만 이 메시지를 호출할 수 있습니다. 이 메시드는 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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 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.

예제

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 Servers 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 Servers 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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 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.

예제

```
// 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 Servers 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift Servers가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(
    const std::string& playerId);
```

파라미터

playerSessionId

AWS SDK Amazon GameLift Servers API 작업 [CreatePlayerSession](#) 호출에 대한 응답으로 Amazon GameLift Servers 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.

예제

```
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 기능](#)에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift Servers 서비스는 OnUpdateGameSession() 콜백 함수를 간접 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 GameLift Servers가 자동으로 ID를 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.

예제

```
// 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 기능](#)에 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.

예제

```
// 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 4.0.1에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 [ProcessEnding\(\)](#)을 호출해야 합니다.

Amazon GameLift Servers 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 GameLift Servers에 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.

예제

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.

예제

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 Servers 4.x용 C# 서버 SDK - 작업

해당 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#) 페이지를 참조하세요.

Amazon GameLift Servers 4.x용 C# 서버 SDK - 데이터 유형

해당 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#) 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C# 서버 SDK 5.x - 데이터 유형](#) 페이지를 참조하세요.

[Amazon GameLift Servers 4.x용 C# 서버 SDK - 작업](#)

주제

- [LogParameters](#)
- [DescribePlayerSessionsRequest](#)
- [ProcessParameters](#)
- [StopMatchBackfillRequest](#)

LogParameters

이 데이터 유형은 게임 세션 중에 생성된 파일 가운데 게임 세션 종료 시 Amazon GameLift Servers가 업로드 및 저장할 파일을 식별합니다. 이 정보는 [ProcessReady\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

logPaths

Amazon GameLift Servers가 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 여부: 아니요

DescribePlayerSessionsRequest

이 데이터 유형은 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.

내용

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

결과와 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마세요. 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 생성을\(를\)](#) 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\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스로 보낸 파라미터 집합이 포함됩니다.

내용

포트

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 GameLift Servers 서비스가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응답으로 이 함수를 호출합니다. 콜백 함수는 [GameSession](#) 객체를 취합니다(Amazon GameLift ServersService API Reference에서 정의함).

유형: `void OnStartGameSessionDelegate(GameSession gameSession)`

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift Servers 서비스가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수의 이름입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 [ProcessEnding\(\)](#) 호출로 응답한 후 서버 프로세스를 종료합니다.

유형: `void OnProcessTerminateDelegate()`

필수 항목 여부: 예

onHealthCheck

Amazon GameLift Servers 서비스가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출합니다. 이 함수를 호출한 후 Amazon GameLift Servers는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상 상태로 기록합니다.

유형: bool OnHealthCheckDelegate()

필수 항목 여부: 예

onUpdateGameSession

Amazon GameLift Servers 서비스가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수의 이름입니다. [매치 채우기](#) 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 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 유형은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 해당 정보는 [StartMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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 작업](#)을 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 Servers 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 GameLift Servers는 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. 해당 정보는 [StopMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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

Amazon GameLift Servers 4.x용 C# 서버 SDK - 데이터 유형

주제

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되어 있고 유효성 검사가 필요함을 Amazon GameLift Servers 서비스에 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 게임

세션에서 플레이어 슬롯을 예약했는지 여부를 확인합니다. 검증되면 Amazon GameLift Servers가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
GenericOutcome AcceptPlayerSession(String playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다. 플레이어 세션 ID는 PlayerSession 객체에서 지정되며, 이 객체는 GameLift API 작업 [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#) 또는 [DescribePlayerSessions](#)에 대한 클라이언트 호출에 응답하여 반환됩니다.

유형: 문자열

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

이 예는 잘못된 플레이어 세션 ID 검증 및 거부를 비롯해 연결 요청을 처리하기 위한 함수를 보여줍니다.

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId){
    var acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```


ActivateGameSession()

Amazon GameLift Servers 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.

예제

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 객체는 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 데이터 유형과 구조가 동일합니다.

예제

다음은 지정된 게임 세션에 활성 상태로 연결되어 있는 모든 플레이어 세션을 검색하는 요청을 보여주는 예입니다. NextToken을 생략하고 Limit 값을 10으로 설정하면, Amazon GameLift Servers가 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.

예제

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

플릿 및 해당 인스턴스와 연결된 pem 인코딩 TLS 인증서의 파일 위치를 검색합니다. 인증서 구성이 GENERATED로 설정된 새 플릿을 생성할 때 인증서를 AWS Certificate Manager 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 요소](#)를 참조하세요.

예제

```
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.

예제

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

종료 시간을 사용할 수 있는 경우 서버 프로세스가 종료되도록 예약된 시간을 반환합니다. 서버 프로세스는 Amazon GameLift Servers 서비스로부터 `onProcessTerminate()` 콜백을 받은 후 이 작업을 수행합니다. 다음과 같은 이유로 Amazon GameLift Servers가 `onProcessTerminate()`를 호출할 수 있습니다. (1) 상태 불량인 경우(서버 프로세스가 상태 불량을 보고했거나 Amazon GameLift Servers에 응답하지 않은 경우), (2) 스케일 다운 이벤트 중에 인스턴스를 종료하는 경우 또는 (3) [스팟 인스턴스 중단](#)으로 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 종료](#)에 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.

예제

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift Servers 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.

예제

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

ProcessEnding()

Amazon GameLift Servers 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 메시드가 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.

예제

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
    Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Amazon GameLift Servers 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 [InitSDK\(\)](#)를 간접 호출하고, 서버 프로세스가 게임 세션을 호스트하기 전에 수행해야 하는 설정 작업을 완료한 후에만 이 메시지를 호출할 수 있습니다. 이 메시드는 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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 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.

예제

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 Servers 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift Servers가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
GenericOutcome RemovePlayerSession(String playerId)
```

파라미터

playerSessionId

새로운 플레이어 세션이 생성되었을 때 Amazon GameLift Servers가 발행한 고유 ID입니다. 플레이어 세션 ID는 `PlayerSession` 객체에서 지정되며, 이 객체는 GameLift API 작업 [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#) 또는 [DescribePlayerSessions](#)에 대한 클라이언트 호출에 응답하여 반환됩니다.

유형: 문자열

필수 항목 여부: 예

반환 값

성공 또는 실패와 함께 오류 메시지로 구성된 일반적인 결과를 반환합니다.

예제

```
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 기능](#)에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift Servers 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.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift Servers가 자동으로 ID를 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.

예제

```
// 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 기능](#)에 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.

예제

```
// 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 4.0.1에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 [ProcessEnding\(\)](#)을 호출해야 합니다.

Amazon GameLift Servers 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 GameLift Servers에 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.

예제

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 Servers 서비스 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.

예제

이 예는 모든 플레이어를 수락하도록 현재 게임 세션의 참여 정책을 설정합니다.

```
var updatePlayerSessionCreationPolicyOutcome =
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

Amazon GameLift Servers용 서버 SDK(Unreal) - 작업

Unreal용 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#) 페이지를 참조하세요.

이 API는 GameLiftServerSDK.h 및 GameLiftServerSDKModels.h에 정의되어 있습니다.

Unreal Engine 플러그인을 설치하려면 코드 샘플 [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#) 섹션을 참조하세요.

Amazon GameLift Servers용 서버 SDK(Unreal) - 데이터 유형

Unreal용 Amazon GameLift Servers 서버 SDK 참조를 사용하여 Amazon GameLift Servers와 함께 호스팅할 멀티플레이어 게임을 통합합니다. 통합 프로세스에 대한 지침은 [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다](#). 페이지를 참조하세요.

Note

본 페이지는 Amazon GameLift Servers의 서버 SDK의 이전 버전에 대한 참조입니다. 최신 버전은 [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 데이터 유형](#) 페이지를 참조하세요.

이 API는 GameLiftServerSDK.h 및 GameLiftServerSDKModels.h에 정의되어 있습니다.

Unreal Engine 플러그인을 설치하려면 코드 샘플 [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#) 섹션을 참조하세요.

[Amazon GameLift Servers용 서버 SDK\(Unreal\) - 작업](#)

주제

- [FDescribePlayerSessionsRequest](#)
- [FProcessParameters](#)
- [FStartMatchBackfillRequest](#)
- [FStopMatchBackfillRequest](#)

FDescribePlayerSessionsRequest

이 데이터 유형은 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 순차적 블록으로 검색합니다.

내용

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

결과와 다음 순차 페이지의 시작을 나타내는 토큰입니다. 반환된 토큰을 이 작업에 대한 이전 호출과 함께 사용합니다. 결과 집합의 시작을 지정하려면 값을 지정하지 마세요. 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](#)을(를) 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\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스로 보낸 파라미터 집합이 포함됩니다.

내용

포트

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 여부: 아니요

onStartGameSession

Amazon GameLift Servers 서비스가 새 게임 세션을 활성화하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 클라이언트 요청 [CreateGameSession](#)에 대한 응

답으로 이 함수를 호출합니다. 콜백 함수는 [GameSession](#) 객체를 취합니다(Amazon GameLift ServersService API Reference에서 정의함).

형식: FOnStartGameSession

필수 항목 여부: 예

onProcessTerminate

Amazon GameLift Servers 서비스가 서버 프로세스를 강제로 종료하기 위해 간접 호출하는 콜백 함수의 이름입니다. 이 함수를 호출한 후 Amazon GameLift Servers는 서버 프로세스가 종료될 때까지 5분을 기다렸다가 [ProcessEnding\(\)](#) 호출로 응답한 후 서버 프로세스를 종료합니다.

형식: FSimpleDelegate

필수 여부: 아니요

onHealthCheck

Amazon GameLift Servers 서비스가 서버 프로세스에서 상태 보고서를 요청하기 위해 간접 호출하는 콜백 함수의 이름입니다. Amazon GameLift Servers는 60초 간격으로 이 기능을 호출합니다. 이 함수를 호출한 후 Amazon GameLift Servers는 60초 동안 응답을 기다립니다. 아무 것도 수신되지 않으면 프로세스를 비정상 상태로 기록합니다.

형식: FOnHealthCheck

필수 여부: 아니요

onUpdateGameSession

Amazon GameLift Servers 서비스가 업데이트된 게임 세션 객체를 전달하기 위해 간접 호출하는 콜백 함수의 이름입니다. [매치 채우기](#) 요청이 처리되면 업데이트된 매치메이커 데이터를 제공하기 위해 Amazon GameLift Servers에서 이 함수를 호출합니다. [GameSession](#) 객체, 상태 업데이트(updateReason) 및 매치 채우기 티켓 ID를 전달합니다.

유형: FonUpdateGameSession

필수 여부: 아니요

FStartMatchBackfillRequest

이 데이터 유형은 매치메이킹 채우기 요청을 보내는 데 사용됩니다. 해당 정보는 [StartMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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 작업](#)을 참조하세요.

형식: FString

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 Servers 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 GameLift Servers는 UUID 형식으로 값을 생성합니다. 이 식별자를 사용하여 매치 채우기 티켓 상태를 추적하거나 필요한 경우 요청을 취소합니다.

형식: FString

필수 여부: 아니요

FStopMatchBackfillRequest

이 데이터 형식은 매치메이킹 채우기 요청을 취소하는 데 사용됩니다. 해당 정보는 [StopMatchBackfill\(\)](#) 호출을 통해 Amazon GameLift Servers 서비스에 전달됩니다.

내용

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

[Amazon GameLift Servers용 서버 SDK\(Unreal\) - 데이터 유형](#)

주제

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)

- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

지정된 플레이어 세션 ID를 가진 플레이어가 서버 프로세스에 연결되어 있고 유효성 검사가 필요함을 Amazon GameLift Servers 서비스에 알립니다. Amazon GameLift Servers는 플레이어 세션 ID가 게임 세션에서 플레이어 슬롯을 예약했는지 여부를 확인합니다. 검증되면 Amazon GameLift Servers가 플레이어 슬롯의 상태를 RESERVED에서 ACTIVE로 변경합니다.

구문

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

파라미터

playerSessionId

AWS SDK Amazon GameLift Servers API 작업 [CreatePlayerSession](#) 호출에 대한 응답으로 Amazon GameLift Servers 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 Servers 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 객체는 AWS SDK Amazon GameLift Servers API [PlayerSession](#) 데이터 유형과 구조가 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()

폴릿 및 해당 인스턴스와 연결된 pem 인코딩 TLS 인증서의 파일 위치를 검색합니다. 인증서 구성이 GENERATED로 설정된 새 폴릿을 생성할 때 인증서를 AWS Certificate Manager 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 폴릿의 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 요소](#)를 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.

예제

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Amazon GameLift Servers SDK를 초기화합니다. 이 메서드는 다른 Amazon GameLift Servers 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 Servers 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 Servers 서비스에 서버 프로세스가 게임 세션을 호스팅할 준비가 되었음을 알립니다. 성공적으로 [InitSDK\(\)](#)를 간접 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 Servers 서비스가 호출하는 메서드입니다.
- 서버 프로세스가 수신하는 포트 번호입니다.
- Amazon GameLift Servers가 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.

예제

[Unreal Engine 플러그인 사용](#) 섹션에 있는 샘플 코드를 참조하세요.

RemovePlayerSession()

Amazon GameLift Servers 서비스에 지정된 플레이어 세션 ID의 플레이어가 서버 프로세스로부터 연결이 해제되었음을 알립니다. 이에 대한 응답으로, Amazon GameLift Servers가 플레이어 슬롯을 새 플레이어에 할당할 수 있는 사용 가능 상태로 변경합니다.

구문

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```

파라미터

playerSessionId

AWS SDK Amazon GameLift Servers API 작업 [CreatePlayerSession](#) 호출에 대한 응답으로 Amazon GameLift Servers 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 기능](#)에 대해 자세히 알아보세요.

이 작업은 비동기식입니다. 새로운 플레이어가 성공적으로 매치되면 Amazon GameLift Servers 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.

- 채우기 요청에 할당할 티켓 ID. 이 정보를 선택 사항입니다. ID를 제공하지 않으면 Amazon GameLift Servers가 자동으로 ID를 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 기능](#)에 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 4.0.1에서는 이 메서드를 사용하지 않습니다. 대신 게임 세션이 종료된 후에 서버 프로세스가 [ProcessEnding\(\)](#)을 호출해야 합니다.

Amazon GameLift Servers 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 GameLift Servers에 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 Servers 서비스 API 참조의 [UpdateGameSession\(\)](#) 작업도 참조하세요.)

구문

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

파라미터

정책

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.

게임 세션 배치 이벤트

Amazon GameLift Servers는 각 게임 세션 배치 요청이 처리될 때 해당 요청에 대해 이벤트를 내보냅니다. [게임 세션 배치의 이벤트 알림 설정](#)에 설명된 대로 Amazon SNS 주제에 이러한 이벤트를 게시할 수 있습니다. 또한 이러한 이벤트는 Amazon CloudWatch Events에 거의 실시간으로 전송됩니다.

이 주제에서는 게임 세션 배치 이벤트의 구조를 설명하고 각 이벤트 유형의 예를 제공합니다. 게임 세션 배치 요청의 상태에 대한 자세한 내용은 Amazon GameLift Servers API 참조의 [GameSessionPlacement](#)를 참조하세요.

배치 이벤트 구문

이벤트는 JSON 객체로 표현됩니다. 이벤트 구조는 유사한 최상위 필드 및 서비스별 세부 정보를 포함하는 CloudWatch 이벤트 패턴을 따릅니다.

최상위 필드에는 다음이 포함됩니다(자세한 내용은 [이벤트 패턴](#) 참조).

버전

이 필드는 항상 0으로 설정됩니다.

id

이벤트의 고유 추적 식별자입니다.

detail-type

값은 항상 GameLift Queue Placement Event입니다.

source

값은 항상 aws.gamelift입니다.

계정

를 관리하는 데 사용되는 AWS 계정입니다Amazon GameLift Servers.

시간

이벤트 타임스탬프입니다.

리전

배치 요청이 처리되는 AWS 리전입니다. 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 홈 리전입니다.

gameSessionLocation

게임 세션이 실행 중인 플릿 위치입니다. 이 값은 플릿의 홈 리전 또는 원격 위치를 지정할 수 있습니다.

playerGatewayStatus

이 게임 세션을 실행하는 플릿 위치에서 플레이어 게이트웨이의 상태입니다.

가능한 값은 다음과 같습니다.

- **ENABLED** - 플레이어 게이트웨이는 이 게임 세션의 플레이어 연결을 라우팅하는 데 사용할 수 있습니다.
- **DISABLED** - 이 게임 세션에는 플레이어 게이트웨이를 사용할 수 없습니다.

computeName

게임 세션을 호스팅하는 컴퓨팅 리소스에 대한 설명 레이블입니다. EC2 플릿의 경우 EC2 인스턴스 ID입니다. 컨테이너 플릿의 경우 플릿 인스턴스의 각 게임 서버 컨테이너 그룹에 컴퓨팅 이름이 할당됩니다. Anywhere 플릿의 경우 사용자 지정 컴퓨팅 이름입니다.

placedPlayerSessions

게임 세션 배치 요청에서 각 플레이어에 대해 생성된 플레이어 세션 모음입니다.

예제

```
{
  "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/us-
east-1/4444dddd-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",
    "playerGatewayStatus": "ENABLED",
    "gameSessionLocation": "us-east-1",
    "computeName": "i-1234567890abcdef0",
    "placedPlayerSessions": [
      {
        "playerId": "player-1"
        "playerSessionId": "psess-1232131232324124123123"
      }
    ]
  }
}
```

PlacementCancelled

GameLift 서비스 [StopGameSessionPlacement](#)의 호출과 함께 배치 요청이 취소되었습니다.

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.

예제

```
{
  "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.

예제

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementTimedOut",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

PlacementFailed

Amazon GameLift Servers는 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.

예제

```
{
  "version": "0",
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "252386620677",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementFailed",
    "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z"
  }
}
```

Amazon GameLift Servers AMI 버전

다음 표에서 Amazon GameLift Servers가 관리형 EC2 호스팅에 사용하는 최신 Amazon Machine Image(AMI)를 확인하세요. [Amazon GameLift Servers의 구성 및 취약성 분석](#)에 설명된 대로 최신 AMI 버전 업데이트를 배포하려면 정기적으로 새 Amazon GameLift Servers 관리형 EC2 플릿을 생성해야 합니다.

Amazon GameLift Servers용 서버 SDK 5+와 함께 사용할 AMI

Amazon GameLift Servers는 다음 AMI를 사용하여 Amazon GameLift Servers 서버 SDK 버전 5와 통합된 게임 서버를 호스팅합니다.

Amazon GameLift Servers 이미지	아키텍처	최신 패치	기본 AWS 이미지
Amazon Linux 2023 BASE_AMI_LINUX_2023	x86	2026-03-27	Amazon Linux 2023 AMI 2023년 10월 20260302.1 x86_64 HVM 커널-6.1(릴리스 정보)
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_5	x86	2026-03-27	Amazon Linux 2 커널 5.10 AMI 2.0.20260302.0 x86_64 HVM gp2(릴리스 정보)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_5	x86	2026-03-27	Windows_Server-2016-English-Full-Base-2026.03.11(릴리스 정보)
Amazon Linux 2023 BASE_AMI_LINUX_2023_ARM	ARM64	2026-03-27	Amazon Linux 2023 AMI 2023년 10월 20260302.1 arm64 HVM 커널-6.1(릴리스 정보)
Amazon Linux 2 BASE_AMI_LINUX_2_ARM	ARM64	2026-03-27	Amazon Linux 2 LTS Arm64 커널 5.10 AMI 2.0.20260302.0 arm64 HVM gp2(릴리스 정보)

Amazon GameLift Servers 이미지	아키텍처	최신 패치	기본 AWS 이미지
Windows 2022 BASE_AMI_WINDOWS_2022	x86	2026-03-27	Windows_Server-2022-English-Full-Base-2026.03.11(릴리스 정보)

Amazon GameLift Servers용 서버 SDK 4와 함께 사용할 AMI

Amazon GameLift Servers는 다음 AMI를 사용하여 Amazon GameLift Servers용 서버 SDK 버전 4와 통합된 게임 서버를 호스팅합니다.

Amazon GameLift Servers 이미지	아키텍처	최신 패치	기본 AWS 이미지
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_4	x86	2026-03-27	Amazon Linux 2 AMI 2.0.20260302.0 x86_64 HVM gp2(릴리스 정보)
Amazon Linux 2023 BASE_AMI_LINUX_2023_SDK_4 중국(베이징) 및 중국(닝샤) 리전에서만 사용할 수 있습니다.	x86	2026-03-27	Amazon Linux 2023 AMI 2023년 10월 20260302.1 x86_64 HVM 커널-6.1(릴리스 정보)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_4	x86	2026-03-27	Windows_Server-2016-English-Full-Base-2026.03.11(릴리스 정보)

자세한 정보는 다음 자료를 참조하세요.

- [Amazon Linux 2023 릴리스 정보](#)

- [Amazon Linux 2 릴리스 정보](#)
- [AWS Windows AMI 버전 기록](#)
- [2024년 소프트웨어 업데이트 서비스 및 Windows Server 업데이트 서비스 변경 내용](#)

Amazon GameLift Servers 서비스 위치

Amazon GameLift Servers 기능은 여러 AWS 리전 및 로컬 영역에서 사용할 수 있습니다. 게임 서버를 플레이어가 있는 위치에 배치하도록 호스팅 솔루션을 설계할 수 있습니다.

지원되는 AWS 위치

다음 표에는 Amazon GameLift Servers 리소스를 지원하는 AWS 리전 및 로컬 영역의 목록이 나와 있습니다. 각 위치에서 생성할 수 있는 리소스 유형을 나타냅니다.

지리적 위치	위치 코드	관리형 플릿의 홈 리전 (단일 위치)	관리형 플릿의 홈 리전 (다중 위치)	관리형 플릿의 원격 위치(다중 위치)	Anywhere 플릿	게임 세션 대기열	FlexMatch 매치메이커 및 규칙 세트	DDoS 보호(플레이어 게이트웨이)
미국 동부(버지니아 북부)	us-east-1	예	예	예	예	예	예	예 [†]
미국 동부(오하이오)	us-east-2	예		예	예	예		
미국 서부(캘리포니아 북부)	us-west-1	예		예	예	예		
미국 서부(오리건)	us-west-2	예	예	예	예	예	예	예 [†]
Africa (Cape Town)	af-south-1			예				

지리적 위치	위치 코드	관리형 풀릿의 홈 리전 (단일 위치)	관리형 풀릿의 홈 리전 (다중 위치)	관리형 풀릿의 원격 위치(다중 위치)	Anywhere 풀릿	게임 세션 대기열	FlexMatch 매치메이커 및 규칙 세트	DDoS 보호(플레이어 게이트웨이)
아시아 태평양 양(태국)	ap-southeast-7			예				
Asia Pacific (Hong Kong)	ap-east-1			예				
아시아 태평양 양(말레이시아)	ap-southeast-5			예				
아시아 태평양 양(뭄바이)	ap-south-1	예		예	예	예		
아시아 태평양 양(오사카)	ap-northeast-3			예				
아시아 태평양 양(서울)	ap-northeast-2	예	예	예	예	예	예	예 [†]
아시아 태평양 양(싱가포르)	ap-southeast-1	예		예	예	예		
아시아 태평양 양(시드니)	ap-southeast-2	예	예	예	예	예	예	예 [†]
아시아 태평양 양(도쿄)	ap-northeast-1	예	예	예	예	예	예	예 [†]

지리적 위치	위치 코드	관리형 풀릿의 홈 리전 (단일 위치)	관리형 풀릿의 홈 리전 (다중 위치)	관리형 풀릿의 원격 위치(다중 위치)	Anywhere 풀릿	게임 세션 대기열	FlexMatch 매치메이커 및 규칙 세트	DDoS 보호(플레이어 게이트웨이)
캐나다(중부)	ca-central-1	예		예	예	예		
유럽(프랑크푸르트)	eu-central-1	예	예	예	예	예	예	예 [†]
유럽(아일랜드)	eu-west-1	예	예	예	예	예	예	예 [†]
Europe (London)	eu-west-2	예		예	예	예		
유럽(밀라노)	eu-south-1			예				
유럽(파리)	eu-west-3			예				
유럽(스톡홀름)	eu-north-1			예				
중동(바레인)	me-south-1			예				
남아메리카 (상파울루)	sa-east-1	예		예	예	예		
애틀랜타 로컬 영역	us-east-1-atl-1			예				

지리적 위치	위치 코드	관리형 풀릿의 홈 리전 (단일 위치)	관리형 풀릿의 홈 리전 (다중 위치)	관리형 풀릿의 원격 위치(다중 위치)	Anywhere 풀릿	게임 세션 대기열	FlexMatch 매치메이커 및 규칙 세트	DDoS 보호(플레이어 게이트웨이)
시카고 로컬 영역	us-east-1-chi-1			예				
댈러스 로컬 영역*	us-east-1-dfw-1			예				
댈러스 로컬 영역 2	us-east-1-dfw-2			예				
덴버 로컬 영역	us-west-2-den-1			예				
휴스턴 로컬 영역	us-east-1-iah-1			예				
캔자스시티 로컬 영역	us-east-1-mci-1			예				
로스앤젤레스 로컬 영역	us-west-2-lax-1			예				
피닉스 로컬 영역	us-west-2-phx-1			예				

지리적 위치	위치 코드	관리형 플릿의 홈 리전 (단일 위치)	관리형 플릿의 홈 리전 (다중 위치)	관리형 플릿의 원격 위치(다중 위치)	Anywhere 플릿	게임 세션 대기열	FlexMatch 매치메이커 및 규칙 세트	DDoS 보호(플레이어 게이트웨이)
라고스, 남아프리카 로컬 영역	af-south-1-los-1			예				

* 이미 옵트인한 AWS 계정에서 사용할 수 있습니다.

† DDoS 보호를 위한 유효한 플릿 홈 리전(플레이어 게이트웨이).

Note

일부 AWS 리전 및 모든 로컬 영역은에 대해 기본적으로 활성화되어 있지 않습니다 AWS 계정. 게임 서버를 해당 위치에 배포하려면 먼저 이러한 리전 또는 로컬 영역에 옵트인해야 합니다. 기본적으로 활성화되지 않는 리전과 이를 활성화하는 방법에 대한 자세한 내용은 AWS 일반 참조의 [AWS 리전관리](#)를 참조하세요. AWS 로컬 영역 사용자 가이드의 [로컬 영역 시작하기](#)를 참조하세요. (2022년 2월 28일 이전에 생성된 플릿은 이 요구 사항에 영향을 받지 않습니다.) 또한 Amazon GameLift Servers 관리자 정책에서 ec2:DescribeRegions 작업을 허용하도록 업데이트해야 합니다. 기본적으로 활성화되지 않는 리전의 정책 예제는 [관리 권한 예제](#) 섹션을 참조하세요.

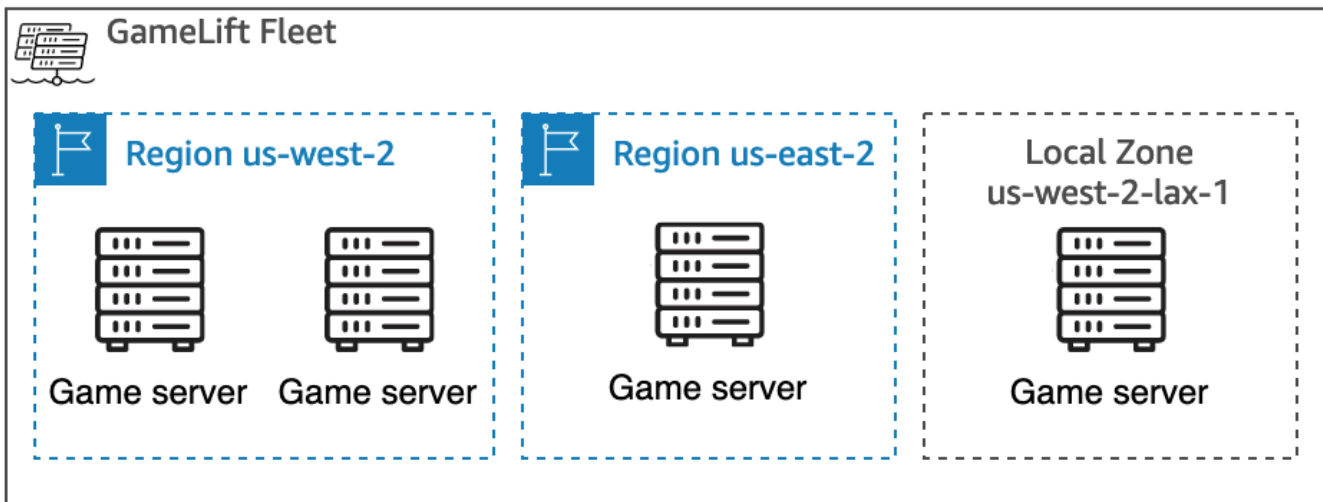
관리형 호스팅 위치

Amazon GameLift Servers 관리형 호스팅은 게임 서버 리소스의 플릿을 배포합니다. 각 플릿은 플릿의 홈 리전 AWS 리전인에 생성됩니다. 플릿의 홈 리전은 플릿의 Amazon 리소스 번호(ARN)에서 참조됩니다.

홈 리전에서만 리소스를 호스팅하여 단일 리전 플릿을 배포할 수 있습니다. 또는 여러 지리적 위치에 리소스를 호스팅하여 다중 위치 플릿을 배포할 수 있습니다. 다중 위치 플릿에는 홈 리전과 하나 이상의 원격 위치가 있습니다. 플릿에서 위치를 그룹화하는 방법에는 제한이 없습니다. 다중 위치 플릿의 호스팅 용량을 관리할 때 각 위치의 용량을 개별적으로 설정할 수 있습니다.

다중 위치 플릿의 원격 위치는 다른 AWS 리전 또는 로컬 영역일 수 있습니다. 로컬 영역은 AWS 리전의 확장입니다. 이를 통해 컴퓨팅 리소스를 사용자에게 더 가깝게 배치하여 지연 시간이 짧은 게임 플레이를 제공할 수 있습니다. 자세한 내용은 [AWS 로컬 영역](#)을 참조하세요. 로컬 영역의 위치 코드는 상위 리전 코드 뒤에 붙는 물리적 위치 식별자입니다. 예를 들어 로스앤젤레스 로컬 영역 코드는 `us-west-2-lax-1`입니다.

다음 다이어그램은 두 AWS 리전 및 하나의 로컬 영역에 리소스가 있는 다중 위치 플릿을 보여줍니다. 플릿의 홈 리전은 `us-west-2`이며 `us-east-2` 리전과 `us-west-2-lax-1` 로컬 영역이라는 두 개의 원격 위치가 있습니다.



플릿 리소스 외에도 Amazon GameLift Servers를 사용한 관리형 호스팅은 다음과 같은 유형의 리소스를 사용할 수도 있습니다. 리소스 유형을 AWS 리전 지원하는 특징에서 이러한 리소스를 생성합니다.

- **빌드** - 관리형 EC2 플릿으로 호스팅되는 게임 서버 빌드입니다. 배포할 플릿과 동일한 리전에 빌드 리소스를 생성합니다.
- **스크립트** - Amazon GameLift Servers Realtime으로 게임을 호스팅하기 위한 구성 스크립트입니다. 배포할 플릿과 동일한 리전에 스크립트 리소스를 생성합니다.
- **컨테이너 그룹 정의** - 관리형 컨테이너 플릿에서 컨테이너를 실행하기 위한 구성입니다. 컨테이너 플릿에 배포할 게임 서버 소프트웨어로 하나 이상의 컨테이너 이미지를 식별합니다. 컨테이너 그룹 정의를 배포할 플릿과 동일한 리전에 생성합니다.
- **게임 세션 대기열** - 이 리소스는 게임 세션에 대한 요청을 처리하고 새 게임 세션을 시작합니다. 처리는 대기열 AWS 리전 이 있는에서 수행됩니다. 게임 세션 배치 프로세스의 지연 시간을 줄이려면 이를 사용할 플레이어 근처에 지리적으로 대기열을 생성합니다.

Amazon GameLift Servers Anywhere의 위치

Amazon GameLift Servers Anywhere 플릿은 사용자가 제공하는 호스팅 하드웨어 모음입니다. 게임 서버 소프트웨어 배포, 업데이트 유지, 서버 프로세스 시작 등 호스팅 리소스에 대한 모든 활동을 관리합니다. Anywhere 플릿을 생성하여 Amazon GameLift Servers 서비스를 자체 관리형 호스팅 리소스와 연결합니다. Amazon GameLift Servers는 플레이어 연결 요청을 처리하고, 사용 가능한 호스팅 리소스를 찾으며, 새 게임 세션을 시작하고, 게임 클라이언트에 연결 정보를 제공하는 등 게임 세션 배치를 관리합니다. 지원하는 모든 AWS 리전 에서 Anywhere 플릿을 생성할 수 있습니다.

Anywhere 플릿을 등록하여 하드웨어 호스팅 인스턴스를 추가합니다. 등록된 각 인스턴스에는 사용자 지정 위치가 연결되어 있어야 합니다. 사용자 지정 위치는 AWS 리전 또는 로컬 영역과 관련이 없습니다. 하드웨어의 물리적 위치를 나타내는 데 사용됩니다.

Anywhere 플릿 생성 및 게임 서버 통합 테스트에 대한 자세한 내용은 [Amazon GameLift Servers Anywhere 플릿 생성](#) 및 [Amazon GameLift Servers Anywhere를 사용하는 로컬 테스트 설정](#) 섹션을 참조하세요.

Amazon GameLift Servers FlexMatch의 위치

FlexMatch 리소스는 매치메이킹에 대한 플레이어 요청을 처리하는 데 사용됩니다. 여기에는 매치메이킹 구성 리소스와 규칙 세트 리소스가 포함됩니다. 처리는 FlexMatch 리소스 AWS 리전 가 위치한에서 이루어집니다. 매치메이킹 프로세스의 지연 시간을 줄이려면 이를 사용할 플레이어 근처에 지리적으로 리소스를 생성합니다. 매치메이킹 구성과 사용하는 규칙 세트는 동일한 AWS 리전에 있어야 합니다. 리소스를 AWS 리전 지원하는 모든에서 FlexMatch 리소스를 생성할 수 있습니다.

호스팅 솔루션의 FlexMatch 설정에 대한 자세한 내용은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.

Amazon GameLift Servers 중국

Sinnet에서 운영하는 중국(베이징) 리전 또는 NWCD에서 운영하는 중국(닝샤) 리전의 리소스에 대해 Amazon GameLift Servers를 사용할 경우에는 별도의 AWS (중국) 계정이 있어야 합니다. 중국 리전에서 일부 기능을 사용할 수 없다는 점을 참고하세요. 이러한 리전에서 Amazon GameLift Servers 사용에 대한 자세한 내용은 다음 리소스를 참조하세요.

- [중국의 Amazon Web Services](#)
- [Amazon GameLift Servers](#)(중국에서 Amazon Web Services 시작하기)

Amazon GameLift Servers 엔드포인트 및 할당량

- 서비스와 프로그래밍 방식으로 연결하는 데 사용할 수 있는 서비스 엔드포인트 및 비컨 엔드포인트는 [Amazon GameLift Servers 서비스 엔드포인트](#)를 참조하세요.
- AWS 계정당 서비스 리소스 또는 작업 사용에 대한 Amazon GameLift Servers 할당량은 [Amazon GameLift Servers 서비스 할당량](#)을 참조하세요. 할당량과 할당량 증가를 요청하는 방법에 대한 자세한 내용은 [AWS Service Quotas](#)를 참조하세요. Amazon GameLift Servers 콘솔을 사용하여 할당량 증가를 요청할 수 있습니다.

Amazon GameLift Servers API 제한

이 섹션의 표에는 Amazon GameLift Servers API 작업에 대한 기본 속도 제한이 나열되어 있습니다. 이러한 제한은 서비스 안정성을 유지하고 공정한 사용을 보장하는 데 도움이 됩니다. 각 API 작업에는 리소스 영향 및 사용 패턴에 따라 특정 제한이 있습니다.

기본적으로 각 고객 계정의 스로틀 제한은 버스트 요청의 경우 500으로, 일반 트래픽의 경우 500으로 설정됩니다. 또한 각 API에는 아래에 설명된 대로 고유한 기본 제한 페어가 있습니다.

API 제한 참조

다음 표에는 Amazon GameLift Servers API 작업에 대한 기본 속도 제한이 나열되어 있습니다. 리소스 수준 스로틀링으로 표시된 API에는 데이터베이스 핫 키 문제를 방지하기 위한 추가 제한이 적용될 수 있습니다.

API 제한 참조

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
AcceptMatch	100	100	아니요	한도는 계정당으로 적용됩니다.
ClaimGameServer	40	20	아니요	한도는 계정당으로 적용됩니다.
CreateAlias	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateBuild	10	1	아니요	한도는 계정당으로 적용됩니다.
CreateContainerFleet	4	1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
CreateContainerGroupDefinition	10	1	아니요	한도는 계정당으로 적용됩니다.
CreateFleet	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateFleetLocations	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateGameServerGroup	5	0.1	아니요	한도는 계정당으로 적용됩니다.
CreateGameSession	50	50	아니요	한도는 계정당으로 적용됩니다.
CreateGameSessionQueue	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateLocation	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateMatchmakingConfiguration	4	1	아니요	한도는 계정당으로 적용됩니다.
CreateMatchmakingRuleSet	4	1	아니요	한도는 계정당으로 적용됩니다.
CreatePlayerSession	200	200	아니요	한도는 계정당으로 적용됩니다.
CreatePlayerSessions	200	200	아니요	한도는 계정당으로 적용됩니다.
CreateScript	10	1	아니요	한도는 계정당으로 적용됩니다.
CreateVpcPeeringAuthorization	1	0.1	아니요	한도는 계정당으로 적용됩니다.
CreateVpcPeeringConnection	1	0.1	아니요	한도는 계정당으로 적용됩니다.
DeleteAlias	4	1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
DeleteBuild	10	1	아니요	한도는 계정당으로 적용됩니다.
DeleteContainerFleet	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteContainerGroupDefinition	10	1	아니요	한도는 계정당으로 적용됩니다.
DeleteFleet	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteFleetLocations	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteGameServerGroup	5	0.1	아니요	한도는 계정당으로 적용됩니다.
DeleteGameSessionQueue	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteLocation	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteMatchmakingConfiguration	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteMatchmakingRuleSet	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteScalingPolicy	4	1	아니요	한도는 계정당으로 적용됩니다.
DeleteScript	10	1	아니요	한도는 계정당으로 적용됩니다.
DeleteVpcPeeringAuthorization	1	0.1	아니요	한도는 계정당으로 적용됩니다.
DeleteVpcPeeringConnection	1	0.1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
DeregisterGameServer	40	20	아니요	한도는 계정당으로 적용됩니다.
DeregisterCompute	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeAlias	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeBuild	20	5	아니요	한도는 계정당으로 적용됩니다.
DescribeCompute	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeContainerFleet	20	5	아니요	한도는 계정당으로 적용됩니다.
DescribeContainerGroupDefinition	20	5	아니요	한도는 계정당으로 적용됩니다.
DescribeEC2InstanceLimits	40	15	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetAttributes	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetCapacity	4	1	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetDeployment	20	5	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetEvents	10	5	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetLocationAttributes	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetLocationCapacity	10	5	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
DescribeFleetLocationUtilization	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetPortSettings	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeFleetUtilization	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeGameServer	40	20	아니요	한도는 계정당으로 적용됩니다.
DescribeGameServerGroup	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeGameServerInstances	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeGameSessionDetails	10	5	아니요	한도는 계정당으로 적용됩니다. 기본 제한이 더 높으며 리소스 수준 제한을 지원하므로 대신 DescribeGameSessions 사용을 고려하는 게 좋습니다.
DescribeGameSessionPlacement	75	75	예	한도는 계정당으로 적용됩니다. 리소스 수준 스로틀링이 적용됩니다.
DescribeGameSessionQueues	10	2	아니요	한도는 계정당으로 적용됩니다.
DescribeGameSessions	20	20	예	한도는 계정당으로 적용됩니다. 리소스 수준 스로틀링이 적용됩니다.
DescribeInstances	10	2	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
DescribeMatchmaking	200	200	예	한도는 계정당으로 적용됩니다. 리소스 수준 스로틀링이 적용됩니다.
DescribeMatchmakingConfigurations	15	7	아니요	한도는 계정당으로 적용됩니다.
DescribeMatchmakingRuleSets	20	10	아니요	한도는 계정당으로 적용됩니다.
DescribePlayerSessions	200	200	예	한도는 계정당으로 적용됩니다. 리소스 수준 스로틀링이 적용됩니다.
DescribeRuntimeConfiguration	4	1	아니요	한도는 계정당으로 적용됩니다.
DescribeScalingPolicies	20	20	아니요	한도는 계정당으로 적용됩니다.
DescribeScript	20	5	아니요	한도는 계정당으로 적용됩니다.
DescribeVpcPeeringAuthorizations	1	0.1	아니요	한도는 계정당으로 적용됩니다.
DescribeVpcPeeringConnections	1	0.1	아니요	한도는 계정당으로 적용됩니다.
GetComputeAccess	4	1	아니요	한도는 계정당으로 적용됩니다.
GetComputeAuthToken	10	5	아니요	한도는 계정당으로 적용됩니다.
GetGameSessionLogUrl	4	1	아니요	한도는 계정당으로 적용됩니다.
GetInstanceAccess	4	1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
ListAliases	10	2	아니요	한도는 계정당으로 적용됩니다.
ListBuilds	20	5	아니요	한도는 계정당으로 적용됩니다.
ListContainerFleets	20	5	아니요	한도는 계정당으로 적용됩니다.
ListCompute	10	2	아니요	한도는 계정당으로 적용됩니다.
ListContainerGroup Definitions	20	5	아니요	한도는 계정당으로 적용됩니다.
ListContainerGroup DefinitionVersions	20	5	아니요	한도는 계정당으로 적용됩니다.
ListFleets	10	2	아니요	한도는 계정당으로 적용됩니다.
ListFleetDeployments	20	5	아니요	한도는 계정당으로 적용됩니다.
ListGameServerGroups	10	2	아니요	한도는 계정당으로 적용됩니다.
ListGameServers	10	2	아니요	한도는 계정당으로 적용됩니다.
ListLocations	10	2	아니요	한도는 계정당으로 적용됩니다.
ListScripts	20	5	아니요	한도는 계정당으로 적용됩니다.
ListTagsForResource	20	5	아니요	한도는 계정당으로 적용됩니다.
PutScalingPolicy	4	1	아니요	한도는 계정당으로 적용됩니다.
RegisterCompute	10	5	아니요	한도는 계정당으로 적용됩니다.
RegisterGameServer	40	20	아니요	한도는 계정당으로 적용됩니다.
RequestUploadCredentials	4	1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
ResolveAlias	4	1	아니요	한도는 계정당으로 적용됩니다.
ResumeGameServerGroup	1	0.1	아니요	한도는 계정당으로 적용됩니다.
SearchGameSessions	10	5	아니요	한도는 계정당으로 적용됩니다.
StartFleetActions	1	0.1	아니요	한도는 계정당으로 적용됩니다.
StartGameSessionPlacement	75	75	아니요	한도는 계정당으로 적용됩니다.
StartMatchBackfill	70	70	아니요	한도는 계정당으로 적용됩니다.
StartMatchmaking	100	100	아니요	한도는 계정당으로 적용됩니다.
StopFleetActions	1	0.1	아니요	한도는 계정당으로 적용됩니다.
StopGameSessionPlacement	20	20	아니요	한도는 계정당으로 적용됩니다.
StopMatchmaking	100	100	아니요	한도는 계정당으로 적용됩니다.
SuspendGameServerGroup	1	0.1	아니요	한도는 계정당으로 적용됩니다.
TagResource	20	5	아니요	한도는 계정당으로 적용됩니다.
TerminateGameSession	10	2	아니요	한도는 계정당으로 적용됩니다.
UntagResource	20	5	아니요	한도는 계정당으로 적용됩니다.
UpdateAlias	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateBuild	10	1	아니요	한도는 계정당으로 적용됩니다.

API 작업	버스트 제한	비율 제한	리소스 수준 스로틀링	참고
UpdateContainerGroupDefinition	10	1	아니요	한도는 계정당으로 적용됩니다.
UpdateContainerFleet	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateFleetAttributes	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateFleetCapacity	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateFleetPortSettings	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateGameServer	40	20	아니요	한도는 계정당으로 적용됩니다.
UpdateGameServerGroup	5	0.1	아니요	한도는 계정당으로 적용됩니다.
UpdateGameSession	10	2	아니요	한도는 계정당으로 적용됩니다.
UpdateGameSessionQueue	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateMatchmakingConfiguration	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateRuntimeConfiguration	4	1	아니요	한도는 계정당으로 적용됩니다.
UpdateScript	10	1	아니요	한도는 계정당으로 적용됩니다.
ValidateMatchmakingRuleSet	4	1	아니요	한도는 계정당으로 적용됩니다.

리소스 수준 스로틀링

데이터베이스 핫 키 문제를 방지하기 위해 일부 API 작업에는 리소스 수준 스로틀링이 적용됩니다. 이러한 작업에 대한 API 수준 제한을 늘리는 동안 리소스 수준에서 여전히 스로틀될 수 있습니다.

다음 API 작업에는 리소스 수준 스로틀링이 적용됩니다.

- DescribeGameSessionPlacement
- DescribeGameSessions
- DescribeMatchmaking
- DescribePlayerSessions

리소스 수준 스로틀링은 리소스 식별자가 포함된 특정 스로틀 키를 사용하여 적용됩니다. 예를 들어 DescribeGameSessionPlacement는 계정 및 배치 ID 조합당 스로틀링을 적용하는 'Operation:GameLift/DescribeGameSessionPlacement,aws-account-placement-id:' 키를 사용합니다. 이렇게 하면 고객이 특정 리소스에 대한 요청으로 시스템에 부담을 주지 않습니다.

Amazon GameLift Servers SDK API 제한

다음 표에는 Amazon GameLift Servers SDK API 작업에 대한 기본 속도 제한이 나열되어 있습니다. 이러한 제한은 성능을 최적화하고 게임 서버 환경에서 리소스 경합을 방지합니다. 이러한 제한을 이해하는 것은 Amazon GameLift Servers와의 효율적인 서버 측 통합에 중요합니다.

서버 SDK 제한은 계정 및 프로세스 식별자가 포함된 특정 스로틀링 키를 사용하여 적용됩니다. 일부 제한은 개별 플레이어의 악용을 방지하기 위해 플레이어별로 적용됩니다.

Note

이 표는 변경될 수 있습니다. 현재 제한에 대해 신뢰할 수 있는 소스는 내부 SDC 구성입니다. 필요한 경우 AWS Support에 문의하여 최신 정보를 확인하세요.

서버 SDK API 제한 참조

API 작업	버스트 제한	비율 제 한	스토링 키	참고
AcceptPlayerSession	3	1	aws-account-and-process-id-and-player-id	한도는 플레이어당 및 프로세스당으로 적용됩니다. 한도가 플레이어당으로 적용되므로 한도 증가가 필요하지 않습니다.
RemovePlayerSession	3	1	aws-account-and-process-id-and-player-id	한도는 플레이어당 및 프로세스당으로 적용됩니다. 한도가 플레이어당으로 적용되므로 한도 증가가 필요하지 않습니다.
ActivateGameSession	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
ActivateHostProcessV2	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
StartMatchBackfill	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
DescribePlayerSessions	10	5	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
GetComputeCertificate	3	0.1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.

API 작업	버스트 제한	비율 제한	스로틀링 키	참고
GetCustomerRoleCredentials	3	0.1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
InitSDK	3	0.1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
RefreshHostProcess	3	0.1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
StopMatchmaking	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
ProcessEnding	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
ProcessReady	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
UpdateGameSession	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.

API 작업	버스트 제한	비율 제한	스로틀링 키	참고
UpdateHostProcess	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
GetGameSessionId	10	5	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
GetTerminationTime	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.
UpdatePlayerSessionCreationPolicy	3	1	aws-account-and-process-id	한도는 서버 프로세스당으로 적용됩니다. 한도가 프로세스당으로 적용되므로 한도 증가가 필요하지 않습니다.

Note

이는 기본 제한입니다. 이러한 API에 스로틀링이 발생하는 경우 계정 수준 스로틀링 때문일 수 있습니다. 한도 증가를 요청하려면 AWS Support에 문의하세요.

프로세스당 및 플레이어당 한도는 일반적인 사용 패턴을 수용하도록 설계되었으며 거의 조정할 필요가 없습니다. 이러한 API에서 스로틀링이 발생하는 경우 한도 증가를 요청하기 전에 잠재적인 최적화 기회를 위해 구현을 검토합니다.

계정 수준 스로틀링은 다음과 같은 추가 규칙을 통해 적용될 수도 있습니다.

- SdkWebSocket/AccountLevelHighUsage,aws-account
- SdkWebSocket/AccountLevelThrottle,aws-account

Amazon GameLift Servers UDP 핑 비콘

UDP 핑 비콘은 플레이어 디바이스와 Amazon GameLift Servers 호스팅 위치 간의 네트워크 지연 시간을 측정할 수 있는 방법을 제공합니다. 핑 비콘을 사용하면 정확한 지연 시간 데이터를 수집하여 게임 서버 배치에 대해 정보에 입각한 결정을 내리고, 지연 시간 요구 사항에 따라 플레이어 매치메이킹을 개선할 수 있습니다.

UDP 핑 비콘 작동 방식

Amazon GameLift Servers는 게임 서버를 배포할 수 있는 각 호스팅 위치에 고정 UDP 엔드포인트(핑 비콘)를 제공합니다. 대부분의 게임 서버는 UDP를 사용하여 통신하므로 UDP 핑 비콘을 사용하여 지연 시간을 측정하면 ICMP 핑을 사용하는 것보다 더 정확한 결과를 얻을 수 있습니다. 네트워크 디바이스는 종종 ICMP 패킷을 UDP 패킷과 다르게 처리하므로 플레이어가 경험할 실제 성능을 반영하지 않는 지연 시간 측정이 발생할 수 있습니다.

UDP 핑 비콘을 사용하면 게임 클라이언트가 이러한 엔드포인트에 UDP 메시지를 보내고 비동기 응답을 수신하여 플레이어의 디바이스와 잠재적 호스팅 위치 간의 실제 게임 트래픽 조건을 더 잘 나타내는 지연 시간 측정값을 제공할 수 있습니다. 엔드포인트는 영구적이며 Amazon GameLift Servers가 해당 위치에서 게임 호스팅을 지원하는 한 계속 사용할 수 있습니다.

UDP 핑 비콘의 일반적인 사용 사례

여러 가지 방법으로 UDP 핑 비콘을 사용하여 게임의 네트워킹 경험을 최적화할 수 있습니다.

최적의 호스팅 위치 선택

다양한 지리적 리전에서 지연 시간 데이터를 수집하여 플레이어 기반에 맞게 게임 서버를 호스팅하는 데 가장 적합한 기본 및 백업 위치를 식별합니다.

플레이어 지연 시간을 기반으로 게임 세션 배치

새 게임 세션을 요청할 때 플레이어 지연 시간 데이터를 포함하면 지연 시간이 가장 짧은 환경을 제공하는 위치를 선택하는 데 도움이 됩니다.

지연 시간을 기반으로 매치메이킹 최적화

매치메이킹을 요청할 때 플레이어 지연 시간 데이터를 제공하여 유사한 지연 시간 프로필을 가진 플레이어를 매칭하고 매치된 플레이어를 위한 최적의 위치에 게임 세션을 배치합니다.

Note

매치메이킹 요청을 생성할 때 플릿이 없는 위치에 지연 시간 정보를 제공해서는 안 됩니다. 이렇게 하면 Amazon GameLift Servers가 플릿 용량이 없는 위치에 게임 세션을 배치하려고 시도 하여 매치메이킹 요청에 실패할 수 있습니다.

비콘 엔드포인트 가져오기

Amazon GameLift Servers 위치에 대한 핑 비콘 도메인 및 포트 정보를 검색하려면 [ListLocations](#) API 작업을 사용합니다. 이 API에서 반환되는 위치 집합은 호출 시 AWS 리전 지정한 (또는 지정하지 않은 경우 기본 리전)에 따라 달라집니다.

다음에서 호출하는 경우:

- 다중 위치를 지원하는 플릿의 홈 리전: API는 모든 호스팅 위치에 대한 정보를 반환합니다.
- 단일 위치를 지원하는 플릿의 홈 리전: API는 해당 위치에 대한 정보를 반환합니다.

다중 위치 플릿의 원격 위치일 수만 있는 위치를 사용하여 이 API를 호출하는 경우 API는 해당 위치 유형에 서비스 엔드포인트가 없기 때문에 오류를 반환합니다.

단일 및 다중 위치 플릿을 지원하는 홈 리전을 식별하려면 [지원되는 AWS 위치](#)의 지원되는 위치 표를 참조하세요.

예제

```
aws gamelift list-locations --region ap-northeast-2
```

이는 다중 위치 플릿을 AWS 리전 지원하므로 여러 위치가 반환됩니다. 다음은 반환 값 중 하나의 예입니다.

```
[...]
{
  "LocationName": "ap-northeast-1",
  "PingBeacon": {
    "UDPEndpoint": {
      "Domain": "gamelift-ping.ap-northeast-1.api.aws",
      "Port": 7770
    }
  }
}
```

```

    }
  }
}

```

Important

각 지연 시간 측정 전에 ListLocations를 호출하는 대신 핑 비콘 정보를 캐시합니다. 도메인 및 포트 정보는 정적이며 API는 대용량 요청을 위해 설계되지 않았습니다.

지연 시간 측정 구현

UDP 핑 비콘을 사용하여 지연 시간 측정을 구현할 때는 다음 모범 사례를 따르세요.

- 다음 방법 중 하나를 사용하여 핑 비콘 정보를 저장합니다.
 - 게임 클라이언트의 엔드포인트를 하드코딩합니다.
 - 게임 백엔드의 정보를 캐시합니다.
 - 정기 업데이트 메커니즘(일별/주별)을 구현하여 정보를 새로 고칩니다.
- UDP 핑 메시지 전송:
 - 비어 있지 않은 한 메시지 본문에 원하는 내용을 넣고 메시지를 최대 크기인 300바이트로 유지합니다.
 - 각 위치에 대해 다음 속도 제한을 준수합니다.
 - 고유한 발신자 IP 주소 및 포트 조합당 초당 3개의 트랜잭션(TPS)
 - 고유한 발신자 IP 주소당 1000TPS
- 지연 시간 계산:
 - 각 위치에 여러 핑을 전송하여 평균 지연 시간을 계산합니다.
 - 더 빠른 결과를 얻으려면 동시 핑을 여러 위치로 보내는 것이 좋습니다.
 - UDP는 전송을 100% 보장하지 않으므로 필요에 따라 재시도 로직을 사용하여 짧은 시간(일반적으로 1~3초) 내에 반환되지 않은 패킷에 대해 새 패킷을 전송합니다.
 - 메시지 전송과 응답 수신 간의 시간 차이를 계산합니다.
 - 위치에 대한 UDP 핑의 대부분이 지속적으로 응답을 받지 못하는 경우 표준 [Amazon GameLift Servers 서비스 엔드포인트](#)에 대한 ICMP 핑을 대체로 사용하여 지연 시간을 계산합니다.

i Tip

플레이어가 로컬 네트워크에서 열어야 하는 포트 목록을 문서화하는 위치에 포트 7770을 포함하는 것이 좋습니다. 이는 이 포트가 차단된 경우 지연 시간을 측정하기 위한 대체(예: ICMP 사용)가 있어야 하는 또 다른 이유입니다.

코드 예제

다음은 UDP 핑을 보내고 지연 시간을 계산하는 방법을 보여주는 몇 가지 간단한 예제입니다.

C++

```
#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <chrono>

int main() {
    // Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    const char* domain = "gamelift-ping.ap-south-1.api.aws";
    const int port = 7770;
    const char* message = "Ping";           // Your message
    const int num_pings = 3;                // Number of pings to send

    // Create socket
    int sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
        std::cerr << "Error creating socket" << std::endl;
        return 1;
    }

    // Resolve domain name to IP address
    struct hostent* host = gethostbyname(domain);
    if (host == nullptr) {
        std::cerr << "Error resolving hostname" << std::endl;
        close(sock);
    }
}
```

```

    return 1;
}

// Set up the server address structure
struct sockaddr_in server_addr;
std::memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
std::memcpy(&server_addr.sin_addr, host->h_addr, host->h_length);

// Set socket timeout
struct timeval tv;
tv.tv_sec = 1; // 1 second timeout
tv.tv_usec = 0;
if (setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv)) < 0) {
    std::cerr << "Error setting socket timeout" << std::endl;
    close(sock);
    return 1;
}

double total_latency = 0;
int successful_pings = 0;

for (int i = 0; i < num_pings; ++i) {
    auto start = std::chrono::high_resolution_clock::now();

    // Send the message
    ssize_t bytes_sent = sendto(sock, message, std::strlen(message), 0,
                                (struct sockaddr*)&server_addr,
sizeof(server_addr));

    if (bytes_sent < 0) {
        std::cerr << "Error sending message" << std::endl;
        continue;
    }

    // Receive response
    char buffer[1024];
    socklen_t server_addr_len = sizeof(server_addr);
    ssize_t bytes_received = recvfrom(sock, buffer, sizeof(buffer), 0,
                                       (struct sockaddr*)&server_addr,
&server_addr_len);

    auto end = std::chrono::high_resolution_clock::now();

```

```

        if (bytes_received < 0) {
            std::cerr << "Error receiving response or timeout" << std::endl;
        } else {
            std::chrono::duration<double, std::milli> latency = end - start;
            total_latency += latency.count();
            successful_pings++;

            std::cout << "Received response, latency: " << latency.count() << " ms"
<< std::endl;
        }

        // Wait a bit before next ping
        usleep(1000000); // 1s
    }

    // Close the socket
    close(sock);

    if (successful_pings > 0) {
        double avg_latency = total_latency / successful_pings;
        std::cout << "Average latency: " << avg_latency << " ms" << std::endl;
    } else {
        std::cout << "No successful pings" << std::endl;
    }

    return 0;
}

```

C#

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Diagnostics;
using System.Threading.Tasks;

class UdpLatencyTest
{
    static async Task Main()
    {

```

```
// Replace with Amazon GameLift Servers UDP ping beacon domain for your
desired location
string domain = "gamelift-ping.ap-south-1.api.aws";
int port = 7770;
string message = "Ping";           // Your message
int numPings = 3;                  // Number of pings to send
int timeoutMs = 1000;              // Timeout in milliseconds

await MeasureLatency(domain, port, message, numPings, timeoutMs);
}

static async Task MeasureLatency(string domain, int port, string message, int
numPings, int timeoutMs)
{
    using (var udpClient = new UdpClient())
    {
        try
        {
            // Resolve domain name to IP address
            IPAddress[] addresses = await Dns.GetHostAddressesAsync(domain);
            if (addresses.Length == 0)
            {
                Console.WriteLine("Could not resolve domain name.");
                return;
            }

            IPEndPoint endPoint = new IPEndPoint(addresses[0], port);
            byte[] messageBytes = Encoding.UTF8.GetBytes(message);

            // Set receive timeout
            udpClient.Client.ReceiveTimeout = timeoutMs;

            double totalLatency = 0;
            int successfulPings = 0;
            var stopwatch = new Stopwatch();

            for (int i = 0; i < numPings; i++)
            {
                try
                {
                    stopwatch.Restart();

                    // Send message
```

```

        await udpClient.SendAsync(messageBytes, messageBytes.Length,
endPoint);

        // Wait for response
        UdpReceiveResult result = await
ReceiveWithTimeoutAsync(udpClient, timeoutMs);

        stopwatch.Stop();
        double latency = stopwatch.Elapsed.TotalMilliseconds;

        totalLatency += latency;
        successfulPings++;

        string response = Encoding.UTF8.GetString(result.Buffer);
        Console.WriteLine($"Ping {i + 1}: {latency:F2}ms - Response:
{response}");
    }
    catch (SocketException ex)
    {
        Console.WriteLine($"Ping {i + 1}: Failed - {ex.Message}");
    }
    catch (TimeoutException)
    {
        Console.WriteLine($"Ping {i + 1}: Timeout");
    }

    // Wait before next ping
    await Task.Delay(1000); // 1s between pings
}

if (successfulPings > 0)
{
    double averageLatency = totalLatency / successfulPings;
    Console.WriteLine($"Summary:");
    Console.WriteLine($"Successful pings: {successfulPings}/
{numPings}");
    Console.WriteLine($"Average latency: {averageLatency:F2}ms");
}
else
{
    Console.WriteLine("\nNo successful pings");
}
}
catch (Exception ex)

```



```

        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
}

static async Task<UdpReceiveResult> ReceiveWithTimeoutAsync(UdpClient client,
int timeoutMs)
{
    using var cts = new System.Threading.CancellationTokenSource(timeoutMs);
    try
    {
        {
            return await client.ReceiveAsync().WaitAsync(cts.Token);
        }
        catch (OperationCanceledException)
        {
            {
                throw new TimeoutException("Receive operation timed out");
            }
        }
    }
}
}

```

Python

```

import socket
import time
import statistics
from datetime import datetime

def udp_ping(host, port, timeout=2):
    """
    Send a UDP ping and return the round trip time in milliseconds.
    Returns None if timeout occurs.
    """
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(timeout)
    message = b'ping'

    try:
        # Resolve hostname first
        try:
            socket.gethostbyname(host)
        except socket.gaierror as e:
            print(f"Could not resolve hostname: {e}")
    
```

```

        return None

    start_time = time.time()
    sock.sendto(message, (host, port))

    # Wait for response
    data, server = sock.recvfrom(1024)
    end_time = time.time()

    # Calculate round trip time in milliseconds
    rtt = (end_time - start_time) * 1000
    return rtt

except socket.timeout:
    print(f"Request timed out")
    return None
except Exception as e:
    print(f"Error: {type(e).__name__}: {e}")
    return None
finally:
    sock.close()

def main():
    # Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    host = "gamelift-ping.ap-south-1.api.aws"
    port = 7770
    num_pings = 3
    latencies = []

    print(f"\nPinging {host}:{port} {num_pings} times...")
    print(f"Start time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")

    for i in range(num_pings):
        print(f"Ping {i+1}:")
        rtt = udp_ping(host, port)

        if rtt is not None:
            print(f"Response from {host}: time={rtt:.2f}ms")
            latencies.append(rtt)

    # Wait 1 second between pings
    if i < num_pings - 1:
        time.sleep(1)

```

```
print()

# Calculate and display statistics
print("-" * 50)
print(f"Ping statistics for {host}:")
print(f"    Packets: Sent = {num_pings}, Received = {len(latencies)}, "
      f"    Lost = {num_pings - len(latencies)} "
      f"    f"({((num_pings - len(latencies)) / num_pings * 100):.1f}% loss)")

if latencies:
    print("\nRound-trip latency statistics:")
    print(f"    Minimum = {min(latencies):.2f}ms")
    print(f"    Maximum = {max(latencies):.2f}ms")
    print(f"    Average = {statistics.mean(latencies):.2f}ms")

print(f"\nEnd time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

if __name__ == "__main__":
    main()
```

Amazon GameLift Servers 릴리스 정보

Amazon GameLift Servers 릴리스 정보는 서비스와 관련된 새로운 기능, 업데이트, 수정 사항에 대한 세부 정보를 제공합니다.

SDK 버전

다음 표에는 SDK 버전 정보와 함께 모든 Amazon GameLift Servers 릴리스 정보가 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 Servers SDK 및 플러그인은 오픈 소스입니다. [Amazon GameLift Servers 개발 도구 가져오기](#)을(를) 참조하세요. 최신 버전을 가져오려면 [Amazon GameLift Servers GitHub 조직](#)을 참조하세요.

현재 버전

버전	이름
1.0.0	SDK
1.0.1	SDK
1.0.2	SDK
1.0.3	SDK
1.0.4	SDK
1.0.5	SDK
1.0.6	SDK
1.0.7	SDK
1.0.8	SDK
1.0.9	SDK
1.0.10	SDK
1.0.11	SDK
1.0.12	SDK
1.0.13	SDK
1.0.14	SDK
1.0.15	SDK
1.0.16	SDK
1.0.17	SDK
1.0.18	SDK
1.0.19	SDK
1.0.20	SDK
1.0.21	SDK
1.0.22	SDK
1.0.23	SDK
1.0.24	SDK
1.0.25	SDK
1.0.26	SDK
1.0.27	SDK
1.0.28	SDK
1.0.29	SDK
1.0.30	SDK
1.0.31	SDK
1.0.32	SDK
1.0.33	SDK
1.0.34	SDK
1.0.35	SDK
1.0.36	SDK
1.0.37	SDK
1.0.38	SDK
1.0.39	SDK
1.0.40	SDK
1.0.41	SDK
1.0.42	SDK
1.0.43	SDK
1.0.44	SDK
1.0.45	SDK
1.0.46	SDK
1.0.47	SDK
1.0.48	SDK
1.0.49	SDK
1.0.50	SDK
1.0.51	SDK
1.0.52	SDK
1.0.53	SDK
1.0.54	SDK
1.0.55	SDK
1.0.56	SDK
1.0.57	SDK
1.0.58	SDK
1.0.59	SDK
1.0.60	SDK
1.0.61	SDK
1.0.62	SDK
1.0.63	SDK
1.0.64	SDK
1.0.65	SDK
1.0.66	SDK
1.0.67	SDK
1.0.68	SDK
1.0.69	SDK
1.0.70	SDK
1.0.71	SDK
1.0.72	SDK
1.0.73	SDK
1.0.74	SDK
1.0.75	SDK
1.0.76	SDK
1.0.77	SDK
1.0.78	SDK
1.0.79	SDK
1.0.80	SDK
1.0.81	SDK
1.0.82	SDK
1.0.83	SDK
1.0.84	SDK
1.0.85	SDK
1.0.86	SDK
1.0.87	SDK
1.0.88	SDK
1.0.89	SDK
1.0.90	SDK
1.0.91	SDK
1.0.92	SDK
1.0.93	SDK
1.0.94	SDK
1.0.95	SDK
1.0.96	SDK
1.0.97	SDK
1.0.98	SDK
1.0.99	SDK
1.0.100	SDK

1. **Native**
2. **클릭**
3. **콜라**
4. **모바일**
5. **이벤트**
6. **클릭 SDK**
7. **SDK**

```

1 Unreal(C
2 Unity
3 H+)
4 Unreal

```

2019

2019

1950

0951

052

0352

	Realtime
	K
	H
	E
	R
	@DK SDK
	Bnreal(C ity H) Unreal
	<u>ONE</u>)
	<u>652</u> S
	<u>851</u> S
	<u>851</u> S
	<u>070</u> S
	<u>051</u> S

[illegible]

	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine

[illegible]

[illegible]

 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	
 Unreal	

	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine
	Unreal Engine

 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	
 Unreal Engine	

328106-1	이	상	328106-1	이	상
328105-1	이	상	328105-1	이	상
328102-1	이	상	328102-1	이	상
328102-0	이	상	328102-0	이	상
311709-0	이	상	311709-0	이	상
311708-1	이	상	311708-1	이	상

[illegible]



릴리스 노트

다음 릴리스 정보는 시간 순서로 나열되어 있으며 최신 업데이트가 먼저 나열됩니다. Amazon GameLift Servers는 2016년에 처음 릴리스되었습니다. 여기에 나열된 릴리스 정보보다 이전 릴리스 정보는 [SDK 버전](#)에서 릴리스 날짜 링크를 참조하세요.

2026년 4월 2일: 게임 세션 데이터, 새 이벤트 필드 및 콘솔 개선 사항에 컴퓨팅 이름 Amazon GameLift Servers 추가

이 릴리스를 사용하면 고객 피드백을 Amazon GameLift Servers 기반으로 게임을 더 간단하게 온보딩하고와 통합할 수 있습니다. 이러한 업데이트를 통해 고객은 플릿 내에서 게임 세션을 더 쉽게 시작하고 관리할 수 있습니다.

Amazon GameLift Servers 이제 게임 세션 API 응답에 ComputeName 필드를 반환합니다. 이 필드를 사용하여 IP 주소를 교차 참조하지 않고 게임 세션을 호스팅하는 컴퓨팅을 식별할 수 있습니다. 이 필드는 관리형 EC2 플릿, 관리형 컨테이너 플릿 및 Anywhere 플릿에 사용할 수 있습니다. 이제 ComputeName 및 GameSessionLocation 필드가 [PlacementFulfilled](#) 및 [MatchmakingSucceeded](#) 이벤트에도 포함됩니다.

Amazon GameLift Servers API 참조 링크:

- [GameSession](#)

이 릴리스에는 몇 가지 Amazon GameLift Servers 콘솔 개선 사항도 포함되어 있습니다. 이러한 업데이트를 통해 이제 다음을 수행할 수 있습니다.

- 빌드 직접 업로드 - Amazon S3 가져오기 방법 외에도 빌드 파일을 선택하여 콘솔 Amazon GameLift Servers에서 직접 업로드할 수 있습니다.
- 플릿 구성 복제 - 기존 관리형 EC2 플릿 구성을 복제하여 새 플릿을 생성합니다.
- 인스턴스 및 컨테이너에 연결 - SSM을 사용하여 컨테이너에 대한 새 액세스 명령을 포함하여 게임 세션, 플레이어 세션 및 컴퓨팅 세부 정보 페이지에서 직접 연결합니다.
- 컴퓨팅 및 인스턴스 데이터 보기 - 이제 게임 세션 테이블에 관리형 EC2 ComputeName 및 컨테이너 플릿에 대한 및 InstanceId 열이 표시됩니다.
- 컨테이너 플릿 컴퓨팅 세부 정보 보기 - 컨테이너 플릿에는 이제 컴퓨팅 테이블과 컴퓨팅 세부 정보 보기가 포함됩니다.
- 게임 세션 로그 위치 보기 - 이제 게임 세션 세부 정보 페이지에 컨테이너 플릿에 대한 Amazon S3 URIs 및 Amazon CloudWatch 로그 그룹 ARNs 표시됩니다.
- 데이터 새로 고침 - 이제 모든 목록 페이지 및 세부 정보 보기에서 새로 고침 버튼을 사용할 수 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers 콘솔](#)
- [PlacementFulfilled](#), Amazon GameLift Servers 개발자 안내서
- [MatchmakingSucceeded](#), Amazon GameLift Servers FlexMatch 개발자 안내서
- [컨테이너 플릿 인스턴스에 연결](#), Amazon GameLift Servers 개발자 안내서

2026년 3월 27일:는 Amazon GameLift Servers 이제 추가에서 차세대 EC2 인스턴스를 지원합니다. AWS 리전

Amazon GameLift Servers는 이제 추가에서 기존 Amazon EC2 인스턴스 선택을 지원 AWS 리전하므로 플레이어와 더 가깝게 게임 서버를 호스팅할 수 있는 유연성이 향상됩니다. 이 확장은 다음 패밀리의 인스턴스를 다룹니다.

- [범용](#)(M 시리즈)

- [컴퓨팅 최적화](#)(C 시리즈)
- [메모리 최적화](#)(R 시리즈)

이제 이러한 인스턴스 유형을 Amazon GameLift Servers 관리형 EC2 플릿 및 관리형 컨테이너 플릿과 함께 사용할 수 있습니다. 플릿 구성을 변경할 필요가 없습니다. 현재 사용하는 것과 동일한 인스턴스 유형 및 설정을 사용하여 새로 지원되는 리전에 플릿을 배포하기만 하면 됩니다. AWS 중국 리전 AWS 리전 을 Amazon GameLift Servers제외하고에서 지원하는 모든에서 사용할 수 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers 인스턴스 요금](#)
- [Amazon GameLift Servers 서비스 위치](#), Amazon GameLift Servers 개발자 안내서
- [CreateFleet](#), Amazon GameLift Servers API 참조
- [CreateContainerFleet](#), Amazon GameLift Servers API 참조

2026년 3월 4일: Amazon GameLift Servers에서 이제 SDKv5의 Linux 기반 EC2 및 컨테이너 플릿에 대한 DDoS 보호 제공

Amazon GameLift Servers는 이제 SDKv5의 Linux 기반 EC2 및 컨테이너 플릿에 대해 distributed-denial-of-service(DDoS) 보호를 제공합니다. 플레이어 게이트웨이 프록시 릴레이 네트워크는 게임 서버 인스턴스와 함께 배치되어 트래픽 검증, 플레이어당 속도 제한 및 게임 서버 IP 주소 난독화를 제공하며 지연 시간이 거의 추가되지 않고 추가 비용이 들지 않습니다. 이 기능은 처음에 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(프랑크푸르트), 유럽(아일랜드), 아시아 태평양(시드니), 아시아 태평양(도쿄), 아시아 태평양(서울) 리전에서 사용할 수 있습니다. 서비스 위치에 대한 자세한 내용은 섹션을 참조하세요 [Amazon GameLift Servers 서비스 위치](#).

자세히 알아보기:

- [블로그 게시물: Amazon GameLift Servers DDoS 보호 소개](#)
- [를 사용한 DDoS 보호 Amazon GameLift Servers](#), Amazon GameLift Servers 개발자 안내서
- [플레이어 게이트웨이 작동 방식](#), Amazon GameLift Servers 개발자 안내서
- [플레이어 게이트웨이를 게임에 통합](#), Amazon GameLift Servers 개발자 안내서
- [C++용 클라이언트 SDK](#)
- [Unreal용 클라이언트 SDK](#). 액세스하려면 GitHub의 Epic Games 조직에 가입해야 합니다. 자세한 내용은 [GitHub의 Unreal Engine](#)을 참조하세요.

2026년 2월 24일: Amazon GameLift Servers 플러그인이 이제 Unity 6.3 및 Unreal Engine 5.7을 지원합니다.

업데이트된 플러그인 버전:

- [Unity용 플러그인](#), 버전 3.2.1(Unity용 서버 SDK, 버전 5.4.0 포함)
- [Unreal용 플러그인](#), 버전 3.1.1(Unreal용 서버 SDK, 버전 5.4.0 포함)

Amazon GameLift Servers는 Unreal Engine 및 Unity용 플러그인 및 서버 SDK의 새 버전을 릴리스합니다. 이러한 최신 버전에는 다음 업데이트가 포함됩니다.

- 이제 Unreal Engine 5.7 및 Unity 6.3에서 사용을 지원합니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인과 게임 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#), Amazon GameLift Servers 개발자 안내서
- [Unity용 Amazon GameLift Servers 플러그인과 게임 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers Unity Engine 프로젝트에 통합](#), Amazon GameLift Servers 개발자 안내서

2026년 1월 29일: Amazon GameLift Servers 이제에서 0개 인스턴스로/에서 플릿 크기 조정 지원

Amazon GameLift Servers는 이제 게임 세션 활동을 기반으로 인스턴스 0개로 자동 조정을 지원합니다. 플릿은 게임 세션 활동이 없는 정의된 기간에 따라 0으로 스케일 다운되고 게임 세션이 요청될 때 0에서 스케일 업되어 비용 최적화 옵션을 제공합니다.

Amazon GameLift Servers API 참조 링크:

- [AWS SDK 작업 UpdateFleetCapacity](#)

자세히 알아보기:

- [제로로/제로에서 Amazon GameLift 서버 플릿 크기 조정 관리](#)

2026년 1월 22일: Amazon GameLift Servers Realtime은 이제 Amazon Linux 2023에서 Node.js 24.x를 지원합니다.

Amazon GameLift Servers Realtime은 이제 Amazon Linux 2023에서 Node.js 24.x를 지원합니다. 이 업데이트를 통해 최신 Amazon Linux 2023 운영 체제에서 실행되는 동안 최신 Node.js 기능 및 보안 개선 사항에 액세스할 수 있습니다.

이제 Realtime플릿을 생성할 때 선택적 설치 스크립트를 포함하여 특정 요구 사항에 맞게 인스턴스 시작 중에 소프트웨어를 설치할 수 있습니다. 새 런타임 버전을 사용하려면 스크립트를 생성할 때 Node.js Realtime 24.x를 지정합니다.

Amazon GameLift Servers API 참조 링크:

- [AWS SDK 작업 CreateScript](#)

자세히 알아보기:

- [Node.js 런타임 버전](#)
- [설치 스크립트 추가](#)

2025년 12월 9일: Amazon GameLift Servers 이제 웹 콘솔에서 AI 기반 지원을 제공합니다.

Amazon GameLift Servers는 이제 Amazon Q Developer를 활용하여 게임 개발자를 위한 맞춤형 지침을 제공하여 웹 콘솔에서 AI 기반 지원을 제공합니다. 이 새로운 기능은 전문 Amazon GameLift Servers 지식을 통합하여 고객이 복잡한 워크플로를 탐색하고, 문제를 해결하고, 게임 서버 배포를 보다 효율적으로 최적화할 수 있도록 지원합니다.

이 새로운 기능은 Amazon GameLift Servers 콘솔 내에서 직접 게임 서버 통합, 플릿 구성 및 성능 최적화를 위한 AI 지원 권장 사항을 제공합니다. 이 개선 사항은 의사 결정 프로세스를 간소화하고, 문제 해결 시간을 단축하고, 전체 리소스 사용률을 개선하여 비용을 절감하고, 플레이어 경험을 개선하는 것을 목표로 합니다.

자세히 알아보기:

- [Amazon GameLift Servers 콘솔](#)

2025년 11월 6일: Amazon GameLift Servers는 이제 Windows Server 2022 운영 체제를 사용하는 게임 빌드를 지원합니다.

Amazon GameLift Servers 고객은 이제 Windows Server 2022 운영 체제를 사용하여 게임 서버를 호스팅할 수 있습니다. Windows Server 2022는 보안을 포함하여 Windows Server 2016보다 몇 가지 향상된 기능을 제공합니다. 이 운영 체제는 중국 리전을 제외한 모든 AWS 리전에서 사용할 수 있습니다.

2027년 1월에 Windows Server 2016에 대한 지원을 종료할 경우 고객은 최신 Windows 운영 체제를 사용하고 중요한 보안 업데이트를 계속 받을 수 있습니다. Windows Server 2022에 대한 지원은 2031년 10월 13일까지 계속됩니다.

Amazon GameLift Servers API 참조 링크:

- [AWS SDK 작업 CreateBuild](#)
- [CLI 명령 upload-build](#)
- [CLI 명령 create-build](#)

2025년 10월 28일: 모든 서버 SDK 및 플러그인에 OpenTelemetry(OTel) 지표 수집기 Amazon GameLift Servers에 추가

Amazon GameLift Servers는 이제 모든 서버 SDK 및 플러그인에 내장된 OpenTelemetry(OTel) 지표 컬렉션을 포함하여 게임 서버 성능 모니터링을 위한 표준화된 관찰성을 제공합니다. 이 통합을 통해 게임 서버에 업계 표준 원격 측정 기능을 직접 제공하여 추가 계측 또는 구성 없이 포괄적인 모니터링 및 문제 해결을 가능하게 합니다.

OpenTelemetry 통합을 통해 이제 다음을 수행할 수 있습니다.

- 표준화된 프로토콜을 사용하여 필수 게임 서버 지표, 로그, 추적 자동 캡처
- 기존 모니터링 플랫폼 및 관찰성 도구와 원활하게 통합
- 모든 플릿 인스턴스에서 게임 서버 성능 및 상태를 실시간으로 모니터링
- 다양한 모니터링 솔루션에서 작동하는 원격 측정 데이터로 더 빠르게 문제 해결

OpenTelemetry는 원격 측정 데이터를 수집하고, 처리하며, 내보내는 표준화된 방법을 제공하는 오픈 소스 관찰성 프레임워크입니다.

자세히 알아보기:

- [서버 원격 측정 지표로 모니터링](#), Amazon GameLift Servers 개발자 안내서

2025년 10월 2일: Amazon GameLift Servers에 플릿 인스턴스로의 콘솔 기반 원격 액세스 추가

Amazon GameLift Servers는 이제 Amazon GameLift Servers 콘솔을 통해 관리형 EC2 및 컨테이너 플릿 인스턴스에 대한 직접 원격 액세스를 제공하므로 인증 관리가 필요하지 않습니다. 이 새로운 기능은 Amazon EC2 Systems Manager(SSM)를 사용하여 추가 설정 또는 자격 증명 관리 없이 안전한 브라우저 기반 터미널 액세스를 제공합니다.

콘솔 기반 원격 액세스를 통해 이제 다음을 수행할 수 있습니다.

- 실시간으로 게임 서버 통합 문제 해결
- 활성 게임 세션 중에 게임 서버 활동 모니터링 및 로그 파일 액세스
- 실제 플레이어 트래픽을 사용하여 진단 및 벤치마킹 도구 실행

이 기능은 서버 SDK 버전 5.x를 실행하는 플릿에 사용할 수 있습니다. 이전 SDK 버전을 실행하는 플릿의 경우 원격 액세스를 위해 기존 AWS CLI 메서드를 계속 사용할 수 있습니다.

플릿 인스턴스에 원격으로 연결하려면 Amazon GameLift Servers 콘솔을 열고 액세스하려는 관리형 EC2 또는 컨테이너 플릿을 찾습니다. 플릿의 세부 정보에서 인스턴스 탭을 열고 목록에서 인스턴스를 선택한 다음 연결을 선택합니다.

자세히 알아보기:

- [플릿 인스턴스에 연결](#), Amazon GameLift Servers 개발자 안내서

2025년 9월 23일: Amazon GameLift Servers에 텍사스주 댈러스의 새로운 로컬 영역에 대한 지원 추가

Amazon GameLift Servers 관리형 호스팅을 사용하면 새로운 텍사스주 댈러스의 로컬 영역(us-east-1-dfw-2)에 게임 서버 리소스를 배포할 수 있도록 됩니다. 로컬 영역을 사용하여 플레이어와 지리적으로 가까운 게임 서버를 배치할 수 있는 능력을 제공하여 지연 시간을 줄이고 게임 플레이를 크게 개선할 수 있습니다. 이 새 로컬 영역에서는 C6gn, C6i, C6in, M6g, M6i, M6in, M8g, R6i 등의 EC2 인스턴스 유형을 사용할 수 있습니다. 각 인스턴스 유형에 대한 자세한 설명은 [AWS EC2 인스턴스 유형](#)을 참조하세요.

새 댈러스 로컬 영역에서 게임 세션 호스팅을 시작하려면 먼저 AWS 계정의 로컬 영역에 옵트인합니다. 그런 다음 신규 또는 기존 다중 위치 플릿에 원격 위치로 추가할 수 있습니다. 게임에서 Amazon

GameLift Servers FlexMatch를 사용하는 경우 매치메이킹 대기열의 풀릿을 업데이트하여 새 로컬 영역을 포함합니다. 다중 위치 풀릿을 사용하면 각 위치에서 호스팅 용량을 직접 관리할 수 있습니다.

AWS 리전 새 델러스, 텍사스, 로컬 영역의 상위 영역은 us-east-1 (버지니아)입니다. 기존 텍사스주 델러스 로컬 영역(us-east-1-dfw-1)은 더 이상 새 옵트인 요청을 수락하지 않습니다.

자세히 알아보기:

- [시작하기 AWS 로컬 영역](#), AWS 로컬 영역 사용 설명서
- [Amazon GameLift Servers 서비스 위치](#), Amazon GameLift Servers 개발자 안내서
- [풀릿 위치 업데이트](#), Amazon GameLift Servers 개발자 안내서

2025년 8월 12일: UE 버전 5.6을 지원하는 Unreal Engine용 Amazon GameLift Servers 플러그인

업데이트된 플러그인 버전:

- [Unreal용 플러그인](#), 버전 3.0.1(Unreal용 서버 SDK, 버전 5.3.1 포함)

Amazon GameLift Servers는 Unreal Engine용 플러그인과 Unreal Engine용 서버 SDK의 새 버전을 릴리스합니다. 이러한 최신 버전에는 다음 업데이트가 포함됩니다.

- 이제 Unreal Engine 5.6에서 사용을 지원합니다.
- Android 클라이언트 대상 패키징 시 컴파일 문제가 해결됩니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인과 게임 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#), Amazon GameLift Servers 개발자 안내서

2025년 6월 24일: 멀티플레이어 게임의 실시간 네트워크 지연 시간을 측정하는 데 도움이 되는 UDP 핑 비콘 Amazon GameLift Servers 출시

업데이트된 SDK 버전:

- AWS SDK 1.11.595

Amazon GameLift Servers는 플레이어 디바이스와 게임 서버 위치 간의 지연 시간을 정확하게 측정하는 데 도움이 되는 UDP 핑 비콘이라는 고정 엔드포인트 세트를 정식 출시합니다. UDP 핑 비콘 엔드포인트는 AWS 중국 리전을 제외하고 Amazon GameLift Servers에서 지원하는 모든 AWS 글로벌 리전 및 로컬 영역에서 사용할 수 있습니다.

대부분의 멀티플레이어 게임은 실시간 게임의 성능 이점 때문에 UDP(사용자 데이터그램 프로토콜)를 기본 패킷 전송 프로토콜로 사용합니다. 최상의 플레이어 경험을 제공하려면 네트워크 지연 시간을 이해하고 최적화하는 것이 중요합니다. UDP 핑 비콘은 플레이어와 게임 서버 간의 실제 UDP 패킷 지연 시간을 측정하는 일관되고 신뢰할 수 있는 방법을 제공하므로 플레이어와 서버 간 매칭 및 게임 세션 배치에 대해 더 나은 결정을 내릴 수 있습니다.

[ListLocations](#) API는 반환되는 위치 목록의 일부로 엔드포인트 도메인 및 포트 정보를 포함하도록 확장 되었으므로 엔드포인트에 프로그래밍 방식으로 쉽게 액세스할 수 있습니다.

게임 클라이언트가 이러한 엔드포인트에 UDP 메시지를 보내고 동일한 데이터와 함께 비동기 응답을 수신하여 플레이어의 디바이스와 잠재적 호스팅 위치 간의 실제 게임 트래픽 조건을 더 잘 나타내는 지연 시간 측정값을 제공할 수 있습니다. 이러한 엔드포인트는 영구적이며 Amazon GameLift Servers가 해당 위치에서 게임 호스팅을 지원하는 한 계속 사용할 수 있습니다.

자세히 알아보기:

- [UDP 핑 비콘](#), Amazon GameLift Servers 개발자 안내서
- [ListLocations](#), Amazon GameLift Servers API 참조

2025년 6월 24일: Amazon GameLift Servers에 태국 및 말레이시아에 대한 위치 지원 추가

Amazon GameLift Servers 관리형 호스팅을 사용하면 이제 태국의 방콕 및 말레이시아의 쿠알라룸푸르에 게임 서버 리소스를 배포하여 동남아시아 전역의 플레이어에게 게임 범위를 확장할 수 있습니다. 이러한 새 리전을 통해 지연 시간을 줄이고 해당 영역에서 플레이어의 게임 플레이 경험을 개선하는 데 도움이 됩니다.

다음은 다중 위치 AWS 리전 플릿의 원격 위치로 사용할 수 있습니다. 이러한 위치에서 게임 세션 호스팅을 시작하려면 이를 새 다중 위치 플릿 또는 기존 다중 위치 플릿에 원격 위치로 추가합니다. 다중 위치 플릿을 사용하면 각 위치에서 호스팅 용량을 직접 관리할 수 있습니다.

- 아시아 태평양(태국)(ap-southeast-7)

- 아시아 태평양(말레이시아)(ap-southeast-5)

AWS 계정에는 기본적으로 활성화되어 AWS 리전 있지 않습니다. Amazon GameLift Servers 리소스를 배포하기 전에 각 리전에 옵트인해야 합니다.

자세히 알아보기:

- [Amazon GameLift Servers 서비스 위치](#), Amazon GameLift Servers 호스팅 개발자 안내서
- [플릿 위치 업데이트](#), Amazon GameLift Servers, 호스팅 개발자 안내서
- [계정에서 AWS 리전 활성화 또는 비활성화](#), AWS 계정 관리 참조 안내서.

2025년 5월 29일: Amazon GameLift Servers가 Unreal Engine 및 Unity용 서버 SDK 및 플러그인의 새 버전 릴리스

업데이트된 SDK 버전:

- [C++ 서버 SDK](#), 버전 5.3.0
- [C# 서버 SDK](#), 버전 5.3.0
- [Go 서버 SDK](#), 버전 5.3.0
- [Unreal용 서버 SDK](#), 버전 5.3.0
- [Unity용 서버 SDK](#), 버전 5.3.0

업데이트된 플러그인 버전:

- [Unreal용 플러그인](#), 버전 3.0.0
- [Unity용 플러그인](#), 버전 3.1.0

C++, C#, Go, Unreal, Unity용 서버 SDK의 새 버전과 Unreal Engine 및 Unity용 새 플러그인 버전이 이제 오픈 소스로 공개됩니다. 모두 [Amazon GameLift Servers GitHub 조직](#)에서 사용할 수 있습니다. 자세한 업데이트 정보는 각 리포지토리의 릴리스 정보 및 Readme 파일을 참조하세요.

주요 서버 SDK 업데이트:

- 모든 서버 SDK에서 클라이언트 측 유효성 검사 및 오류 응답이 향상되었습니다.
- [OnProcessTerminate](#) 콜백 함수에 게임 서버 프로세스를 종료하는 기본 로직이 포함되었습니다.
- [InitSDK\(\)](#) 함수가 멍등성 토큰을 사용하여 여러 번의 재시도를 지원합니다.

- [OnUpdateGameSession](#) 콜백이 autoBackfillMode에 대한 데이터를 전달할 수 있습니다.

주요 플러그인 업데이트:

- Unreal Engine용 플러그인에 더 많은 자동화와 더 적은 사전 조건(CMake, OpenSSL, Unreal 교차 컴파일 도구 체인)으로 더 간소화된 설치 및 설정 프로세스가 포함되었습니다.
- Unreal Engine용 플러그인은 게임 클라이언트 및 서버 빌드 경로의 공간 지원을 포함하여 관리형 EC2 워크플로에 대한 UI 환경을 개선했습니다. 또한 편집기에서 게임 클라이언트를 시작할 때 명령 줄 인수를 추가할 수 있습니다.
- Unreal Engine용 플러그인은 UE5에서 ARM 서버 빌드를 지원할 수 있습니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers용 서버 SDK 5.x](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers를 Unity 프로젝트에 통합](#), Amazon GameLift Servers 개발자 안내서

2025년 4월 24일: Amazon GameLift Servers용 C# 서버 SDK 5의 .NET 8 지원

업데이트된 SDK 버전:

- C# Server SDK, 버전 5.2.1

C#을 사용하는 게임 개발자의 경우 .NET 8을 Amazon GameLift Servers 프로젝트의 대상 프레임워크로 사용할 수 있습니다. .NET 8을 사용하면 적시(JIT) 컴파일 개선, 메모리 사용량 최적화, 더 빠른 시작 시간 등 성능 개선을 활용할 수 있습니다. 현재 .NET 6을 사용하는 경우 C# 서버 SDK를 최신 버전으로 업데이트하는 등 .NET 8로의 마이그레이션을 계획하는 것이 좋습니다. Microsoft는 2026년 11월까지 보안 패치 및 기술 업데이트와 함께 .NET 8 지원을 발표했습니다.

[Amazon GameLift Servers용 C# 서버 SDK의 최신 버전을 다운로드합니다.](#)

자세히 알아보기:

- [Amazon GameLift Servers 개발 도구 가져오기](#), Amazon GameLift Servers 개발자 안내서

2025년 3월 27일: Amazon GameLift Servers에서 차세대 EC2 인스턴스 패밀리에 대한 지원 확장

업데이트된 SDK 버전:

- AWS SDK 1.11.535

이제 5세대~8세대 인스턴스 패밀리에서 다양한 Amazon EC2 인스턴스를 선택하여 Amazon GameLift Servers로 게임 서버 호스팅을 미세 조정할 수 있습니다. 각각의 새로운 EC2 세대는 최첨단 AWS Graviton4 및 Intel Xeon 기반 인스턴스를 제공하는 8세대 인스턴스를 통해 EC2 컴퓨팅, 메모리, 네트워크 분야의 발전을 제공합니다. 차세대 인스턴스는 다음 인스턴스 패밀리에서 사용할 수 있습니다.

- [범용](#)(M 시리즈)
- [컴퓨팅 최적화](#)(C 시리즈)
- [메모리 최적화](#)(R 시리즈)

로컬 스토리지(d), 향상된 네트워킹(n), 특정 프로세서 아키텍처(Intel/AMD/Graviton - i/a/g)를 제공하는 변형을 선택할 수도 있습니다. AWS 중국 리전을 Amazon GameLift Servers제외하고에서 AWS 리전 지원하는 모든에서 차세대 인스턴스를 사용할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 서비스 위치](#) 섹션을 참조하세요.

이러한 새 인스턴스 유형을 Amazon GameLift Servers 관리형 EC2 플릿 및 관리형 컨테이너 플릿과 함께 사용합니다. 기존 게임 호스팅을 새 인스턴스 유형(동일한 아키텍처)으로 전환할 때는 인스턴스 유형을 제외한 모든 구성 설정이 변경되지 않은 채 새 플릿을 배포합니다.

자세히 알아보기:

- [Amazon GameLift Servers 인스턴스 요금](#)
- [CreateFleet](#), Amazon GameLift Servers API 참조
- [CreateContainerFleet](#), Amazon GameLift Servers API 참조

2025년 3월 11일: 신속한 온보딩을 위해 Amazon GameLift Servers의 새로운 게임 서버 래퍼 출시

Amazon GameLift Servers용 새 게임 서버 래퍼는 Amazon GameLift Servers에서 호스팅되는 게임 서버를 가져오는 데 필요한 시간을 크게 줄입니다. 코드 변경 없이 래퍼를 사용하여 게임에 기본 게임 세션 관리 기능을 추가하고 Amazon GameLift Servers Anywhere 플릿, 관리형 EC2 플릿 또는 관리형 컨테이너 플릿에 배포할 수 있습니다. 이 도구는 자체 게임 서버 또는 샘플 게임의 기능을 사용하여 Amazon GameLift Servers 기능에 대한 실습 평가를 수행하는 데 적합합니다. 또한 빠른 프로토타입 생성 또는 테스트와 같은 게임 서버 반복을 빠르게 배포하는 데 유용합니다.

기본 게임 세션 관리 기능을 사용하면 게임 서버가 Amazon GameLift Servers 서비스와의 연결을 초기화하고, 게임 세션을 시작 및 중지하라는 프롬프트에 응답하며, 게임 세션이 완료되면 종료할 수 있습니다.

[GitHub에서 게임 서버 래퍼를 다운로드합니다.](#) 게임 프로젝트에서 래퍼를 설치하고 사용하는 방법은 GitHub의 Readme 파일을 참조하세요.

2025년 3월 6일: Amazon GameLift는 현재 Amazon GameLift Servers 및 Amazon GameLift Streams

Amazon GameLift는 개발자가 세계에서 가장 까다로운 게임을 빌드, 확장, 제공할 수 있도록 지원하는 완전관리형 서비스입니다. Amazon GameLift Streams의 정식 출시로 Amazon GameLift는 이제 대규모 게임 서버와 게임 플레이의 원활한 스트리밍 기능을 모두 제공합니다.

[Amazon GameLift Servers](#)는 게임 개발자에게 전용 게임 서버를 배포, 운영, 확장할 수 있는 기능을 제공합니다. 몇 분 만에 클라우드에 고성능 게임 서버를 배포하고 플레이어 수요에 맞게 확장 및 축소할 수 있습니다. AWS 검증된 컴퓨팅 환경을 기반으로 구축되는 단일 게임에서 1억 명의 동시 플레이어, 초당 10만 명의 플레이어 추가, 분당 9천 개의 새 컴퓨팅 인스턴스를 Amazon GameLift Servers 지원합니다. 또한 엔터프라이즈급 보안, 대규모 군중을 위한 매치메이킹, 종량제 유연성을 통해 새로운 게임 아이디어로 작업하던 수백만 명의 플레이어로 게임을 실행하던 시작할 수 있도록 합니다.

[Amazon GameLift Streams](#)를 사용하면 게임 개발자가 iOS, Android, FireOS, 게이머용 PC를 포함한 디바이스에서 지연 없이 최대 1080p 해상도와 60fps(초당 프레임)로 게임 스트리밍 경험을 제공할 수 있습니다. 게시자는 단일 AWS 상품을 사용하여 수정 없이 몇 분 안에 게임 콘텐츠를 완전 관리형 클라우드 기반 GPU 인스턴스에 배포하고 AWS Network Backbone을 통해 웹 브라우저를 통해 최종 플레이어 디바이스로 직접 전송할 수 있습니다. 플레이어는 다운로드 또는 설치를 기다리지 않고 몇 초 만에 게임을 시작할 수 있으며 PC 또는 게임 콘솔에서 로컬로 게임을 플레이하는 것과 거의 구별할 수 없는 게임 플레이 경험을 제공합니다.

2025년 1월 14일: Amazon GameLift Servers의 게임 세션 배치 우선순위 지정에 대한 확장된 지원 출시

업데이트된 SDK 버전:

- AWS SDK 1.11.485

고객 피드백에 대한 응답으로 개별 게임 세션 배치 요청의 위치 우선순위를 지정할 수 있는 새로운 기능을 출시합니다. 위치별로 배치 우선순위를 지정하도록 구성된 대기열의 경우 이제 각 배치 요청과 함께 우선순위 위치의 사용자 지정 목록을 제공할 수 있습니다.

이 새로운 기능을 통해 고객은 필요에 따라 각 배치 요청의 위치 우선순위를 동적으로 변경할 수 있습니다. 추가 유연성은 플레이어 위치, 플릿 로드 또는 서버 상태와 같은 변화하는 조건에 더 잘 대응할 수 있음을 의미합니다. 배치 위치를 선택하는 방법을 추가로 사용자 지정하려는 고객을 지원할 수도 있습니다.

자세히 알아보기:

- [게임 세션 배치 우선순위](#), Amazon GameLift Servers 개발자 안내서
- [StartGameSessionPlacement](#), Amazon GameLift Servers API 참조

2025년 1월 2일: Amazon GameLift Servers의 게임 세션 종료에 대한 새로운 지원 출시

업데이트된 SDK 버전:

- AWS SDK 1.11.477

고객 피드백에 대한 응답으로 개별 게임 세션을 더 쉽게 종료할 수 있는 새로운 기능을 출시합니다. 이번 릴리스에서는 Amazon GameLift Servers 콘솔에서 직접 또는 용 AWS CLI 또는 AWS SDK를 사용하여 게임 세션을 종료할 수 있습니다Amazon GameLift Servers.

이 새로운 기능은 활성 상태로 유지되지만 상태가 좋지 않은 게임 세션을 해결해야 하는 필요성을 해결하여 컴퓨팅 리소스가 새 게임 세션을 호스팅하지 못하도록 합니다. 이전에는 고객이 게임 세션을 수동으로 종료하기 위해 컴퓨팅에 원격으로 액세스해야 했습니다.

두 가지 종료 방법 중에서 선택할 수 있습니다. 첫 번째 방법은 사용자 지정 종료 시퀀스를 사용하여 게임 세션을 정상적으로 종료하려고 시도합니다. 여기에는 플레이어에게 알리고 게임 데이터를 확인하는 작업이 포함될 수 있습니다. 두 번째 방법은 서버 프로세스를 강제로 중지하여 게임 세션을 즉시 종

료합니다. 이 두 번째 방법을 통해 서버 프로세스가 응답하지 않는 경우에도 게임 세션이 종료되도록 합니다.

자세히 알아보기:

- [Amazon GameLift Servers를 사용하여 게임 세션 종료](#), Amazon GameLift Servers 개발자 안내서
- [TerminateGameSession](#), Amazon GameLift Servers API 참조

2024년 12월 19일: Amazon GameLift Servers의 관리형 컨테이너에 대한 게임 엔진 플러그인 지원 출시

업데이트된 플러그인 버전:

Unreal Engine용 Amazon GameLift Servers 플러그인, 버전 2.0.0

- 관리형 컨테이너를 지원하는 C++ 서버 SDK 5.2.0을 지원하도록 업그레이드되었습니다.
- Unreal Engine 5.4 및 5.5에 대한 지원이 추가되었습니다.

Unity용 Amazon GameLift Servers 플러그인, 버전 3.0.0

- 관리형 컨테이너를 지원하는 C++ 서버 SDK 5.2.0을 지원하도록 업그레이드되었습니다.
- Windows 및 Mac OS용 Unity 2021.3 LTS 및 2022.3 LTS를 지원합니다.

Unreal 및 Unity 게임 엔진용 Amazon GameLift Servers 플러그인은 Amazon GameLift Servers를 사용하여 게임을 시작하고 실행하는 단계를 간소화하는 도구와 워크플로를 제공합니다. Amazon GameLift Servers는 게임 개발자가 세션 기반 멀티플레이어 게임을 위한 전용 게임 서버를 관리하고 확장하는 데 사용할 수 있는 완전 관리형 클라우드 호스팅 서비스입니다.

최신 플러그인 버전은 다음과 같은 향상된 기능을 제공합니다.

- 관리형 컨테이너를 사용한 호스팅을 위한 안내형 워크플로입니다. 이 워크플로는 게임 서버 소프트웨어로 컨테이너 이미지를 설정하고 게임 서버에 클라우드 기반 호스팅 솔루션을 배포하는 단계를 안내합니다. 워크플로는 간단한 배포와 게임 세션 배치 대기열 및 FlexMatch 매치메이커를 사용한 보다 완전한 배포라는 두 가지 배포 시나리오를 제공합니다. 각 시나리오는 Amazon GameLift Servers 컨테이너 플릿과 지원 AWS 리소스를 생성합니다.
- AWS 사용자 프로필을 설정하고 플러그인 사용을 위한 AWS 액세스 자격 증명을 관리하는 프로세스가 개선되었습니다. 여러 프로필을 유지하여 다양한 AWS 계정, 계정 사용자 및 리전에서 작업할 수 있습니다.

- 기존 컨테이너 플릿을 업데이트하는 추가 기능입니다. 처음부터 시작할 필요 없이 새 컨테이너 이미지(예: 게임 서버 버전 업데이트용)를 배포하고 플릿 구성 설정을 변경할 수 있습니다.
- Amazon GameLift Servers Anywhere 플릿 및 관리형 EC2를 사용하여 호스팅하기 위한 워크플로를 개선했습니다. 고객 피드백을 기반으로 한 개선 사항에는 유용한 리소스에 대한 팁과 링크가 포함된 더 나은 지침이 포함됩니다.

관리형 컨테이너 및 관리형 EC2 솔루션의 배포 시나리오는 AWS CloudFormation 템플릿을 사용하여 각 시나리오에 대한 AWS 리소스를 생성하고 배포합니다. 이러한 템플릿은 Amazon GameLift Servers 플러그인 다운로드에 포함되어 있으며 편집할 수 있습니다. 그대로 사용하거나 게임에 맞게 수정할 수 있습니다.

자세히 알아보기:

- [Unreal용 플러그인: 관리형 컨테이너 플릿에 게임 배포](#), Amazon GameLift Servers 개발자 안내서
- [Unity용 플러그인: 관리형 컨테이너 플릿에 게임 배포](#), Amazon GameLift Servers 개발자 안내서
- [GitHub에서 Unreal용 플러그인 다운로드](#)
- [GitHub에서 Unity용 플러그인 다운로드](#)
- [Amazon GameLift Servers를 사용한 게임 호스팅](#)
- [Amazon GameLift Servers 포럼](#)

2024년 11월 12일: Amazon GameLift Servers의 관리형 컨테이너를 사용하여 멀티플레이어 게임 호스팅 출시

업데이트된 SDK 버전:

- AWS SDK 1.11.445
- 서버 SDK, 버전 5.2.0(모든 언어)

Amazon GameLift Servers는 일반 가용성을 위해 컨테이너화된 게임 서버 워크로드를 위한 새로운 호스팅 솔루션을 출시합니다. 이번 릴리즈를 통해 게임 개발자가 일관되고 안전한 환경, 간소화된 배포 프로세스, 최적화된 리소스 사용률 등 컨테이너화의 이점을 활용할 수 있습니다.

관리형 컨테이너 플릿은 Amazon GameLift Servers에서 사용자를 대신하여 구성에 따라 관리하는 Amazon EC2 인스턴스를 사용합니다. 게임용 사용자 지정 컨테이너 아키텍처를 구축하고 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 저장하여 컨테이너 이미지를 제공합니다. 컨

테이너 플릿은 Linux 기반 게임 서버에서만 사용할 수 있습니다. 게임 서버는 서버 SDK 5.2.0 이상과 통합되어야 합니다.

관리형 컨테이너 플릿을 사용하면 관리형 EC2 플릿과 동일한 이점을 얻을 수 있습니다. 여기에는 온디맨드 및 스팟 인스턴스 유형, 지능형 용량 조정, 대기열을 사용한 게임 세션 배치, 매치메이킹에 대한 지원이 포함됩니다. 컨테이너에 대한 일부 새 플릿과 함께 다른 플릿 유형과 동일한 지표를 얻을 수도 있습니다. 컨테이너 플릿의 기타 기능은 다음과 같습니다.

- 컨테이너화된 워크로드에 대한 서버리스 환경과 조정합니다. 컨테이너당 하나의 게임 서버 프로세스를 실행하고 최적의 리소스 사용을 위해 각 플릿 인스턴스에 많은 컨테이너를 패킹합니다. 여러 게임 서버 프로세스가 있는 컨테이너를 원하는 경우 자동화된 호스트 관리를 위해 Amazon GameLift Servers 에이전트를 사용할 수 있습니다.
- 플릿 생성이 간소화되었습니다. 컨테이너 플릿은 합리적인 제안/기본값으로 배포 구성 설정을 최소화하도록 설계되었습니다. 작동 중인 플릿을 빠르게 배포한 다음 필요에 따라 개별 설정을 사용자 지정할 수 있습니다.
- 컨테이너 아키텍처용 버전 관리 도구입니다. 이제 컨테이너 그룹 정의(컨테이너 '작업'과 유사)를 업데이트하고, 여러 버전을 유지 관리하며, 플릿에 배포할 버전을 지정할 수 있습니다.
- 플릿 업데이트 도구입니다. 컨테이너 플릿을 사용하면 게임 서버 버전 업데이트를 릴리스할 경우 더 이상 새 플릿을 생성할 필요가 없습니다. 대신 컨테이너 이미지를 업데이트하고 기존 플릿에 업데이트를 배포할 수 있습니다.

가 다중 위치 플릿을 Amazon GameLift Servers 지원하는 AWS 리전 모든에서 Amazon GameLift Servers 컨테이너 플릿을 빌드하고 지원되는 모든 원격 위치에 컨테이너 플릿 인스턴스를 배포할 수 있습니다. 자세한 내용은 [Amazon GameLift Servers 서비스 위치](#) 섹션을 참조하세요. 관리형 컨테이너는 현재 AWS 중국 리전에서 사용할 수 없습니다.

자세히 알아보기:

- 블로그 게시물: [완전관리형 컨테이너를 활용하여 Amazon GameLift Servers에서 글로벌 규모의 멀티플레이어 게임 호스팅](#)
- [관리형 컨테이너](#) 개요, Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers의 컨테이너 작동 방식](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers 관리형 컨테이너를 사용한 호스팅을 위한 개발 로드맵](#), Amazon GameLift Servers 개발자 안내서
- [CreateContainerFleet](#), Amazon GameLift Servers API 참조

2024년 9월 19일: Amazon GameLift Servers의 Unreal Engine용 C++ 서버 SDK 및 플러그인에 대한 업데이트 출시

업데이트된 SDK 버전:

C++ 서버 SDK 버전 5.1.3

- 새로운 로깅 기능 이제 SDK 요청 로그에 액세스할 수 있습니다.
- SDK 메시지 전송 신뢰성이 개선되었습니다. 이제 SDK가 더 강력한 재연결 메커니즘을 사용하여 네트워크 중단 또는 무작위 메시지 삭제 이벤트를 복구합니다.

업데이트된 플러그인 버전:

Unreal Engine용 Amazon GameLift Servers 플러그인, 버전 1.1.2

- C++ 서버 SDK 5.1.3 최신 버전을 지원하도록 업그레이드되었습니다.
- Unreal Engine용 Amazon GameLift Servers 플러그인에서 플릿에 대한 서버 빌드 실행 파일을 검색할 때 모든 파일을 검색할 수 있는 옵션이 추가되었습니다.

Unreal용 C++ 서버 SDK 플러그인 버전 5.1.2

- C++ 서버 SDK 5.1.3 최신 버전을 지원하도록 업그레이드되었습니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인과 게임 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers 플러그인 및 SDK 다운로드](#)

2024년 9월 5일: 플릿 생성 프로세스의 Amazon GameLift Servers 관찰성 개선

고객 피드백을 기반으로 관리형 EC2 플릿을 생성하고 게임 세션을 호스팅할 준비를 하기 위한 Amazon GameLift Servers 워크플로를 구체화했습니다. 개선 사항에는 다음이 포함됩니다.

- 플릿 생성 프로세스의 각 단계에 대한 보다 구체적이고 정확한 설명을 제공했습니다. 이렇게 향상된 가시성을 통해 문제를 더 쉽게 파악하고 해결할 수 있습니다.

- 빌드 및 활성화 단계는 인스턴스 배포 작업(빌드)의 분리성을 강화해 게임 서버 프로세스를 시작하고 Amazon GameLift Servers 서비스(활성화)에 연결하는 데 도움이 됩니다. 이 변경 사항을 통해 문제의 가능한 원인을 더 쉽게 인식할 수 있습니다. 또한 이제 활성화 단계에 있을 때 플릿에 원격으로 연결할 수 있습니다.
- 두 개의 새로운 플릿 생성 이벤트는 게임 서버 설치 스크립트의 성공 또는 실패를 전달합니다. 게임 서버 빌드에 설치 스크립트가 포함된 경우는 스크립트 실행을 Amazon GameLift Servers 시도하고 다음 새 이벤트 중 하나를 내보냅니다.
 - FLEET_CREATION_COMPLETED_INSTALLER
 - FLEET_CREATION_FAILED_INSTALLER

자세히 알아보기:

- [Amazon GameLift Servers 플릿 문제 디버깅](#), Amazon GameLift Servers 개발자 안내서
- [이벤트 데이터 유형](#), Amazon GameLift Servers API 참조

2024년 8월 15일: Amazon GameLift Servers 콘솔 환경 개선

고객 피드백에 따라 [Amazon GameLift Servers 콘솔](#) 환경을 다음과 같이 업데이트했습니다.

- 이제 페이지에 대한 표시 기본 설정이 AWS 계정 사용자에게 자동으로 저장되고 페이지로 돌아갈 때마다 적용됩니다. 디스플레이 기본 설정을 사용하면 플릿 목록 페이지와 같이 테이블 디스플레이에 포함할 정보를 선택할 수 있습니다. 테이블의 오른쪽 상단 모서리에 있는  아이콘을 사용하여 표시할 기본 설정을 사용자 지정합니다.
- 플릿 위치와 인스턴스 유형 선택을 결합해 관리형 EC2 플릿에 대한 플릿 생성 워크플로가 간소화되었습니다. 위치 선택을 변경하더라도 플릿에 적합한 인스턴스 유형을 더 쉽게 찾을 수 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers 관리형 EC2 플릿 생성](#), Amazon GameLift Servers 개발자 안내서

2024년 7월 25일: AWS 나이지리아 로컬 영역에 대한 지원 Amazon GameLift Servers 추가

Amazon GameLift Servers 관리형 호스팅을 사용해 서아프리카 나이지리아에 게임 서버 리소스를 배포하고 아프리카 전역의 플레이어에게 게임 범위를 확장할 수 있습니다. AWS Local Zones를 사용하면 플레이어와 지리적으로 더 가까운 게임 서버를 배치하여 지연 시간을 줄이고 게임 플레이를 크게 개선할 수 있습니다.

나이지리아에서 게임 세션 호스팅을 즉시 시작하려면 새 나이지리아 로컬 영역을 새 또는 기존 다중 위치 플릿에 원격 위치로 추가합니다. 게임에서 Amazon GameLift Servers FlexMatch를 사용하는 경우 매치메이킹 대기열의 플릿을 업데이트하여 새 로컬 영역을 포함합니다. 다중 위치 플릿을 사용하면 각 위치에서 호스팅 용량을 직접 관리할 수 있습니다.

나이지리아 AWS 리전 라고스 로컬 영역의 상위 영역은 아프리카(케이프타운) 리전(af-south-1)이며, 원격 위치로 Amazon GameLift Servers도 지원합니다. 나이지리아 로컬 영역 이름은 af-south-1-los-1입니다.

자세히 알아보기:

- [Amazon GameLift Servers 서비스 위치](#), Amazon GameLift Servers 개발자 안내서
- [플릿 위치 업데이트](#), Amazon GameLift Servers 개발자 안내서

2024년 7월 2일: 플레이어 세션 데이터를 볼 수 있는 Amazon GameLift Servers 콘솔 신규 도구 릴리스

Amazon GameLift Servers 콘솔로 플레이어 세션 조회 도구를 제공하여 게임 세션 ID, 플레이어 세션 ID 또는 플레이어 ID로 플레이어 세션 정보를 검색할 수 있습니다. FlexMatch 매치메이킹을 사용하는 게임이 일치하는 모든 플레이어에 대해 플레이어 세션을 자동으로 생성합니다. 다른 모든 게임의 경우 플레이어 세션은 선택 기능입니다.

Amazon GameLift Servers 콘솔의 기본 탐색 창에서 플레이어 세션 조회 도구를 찾을 수 있습니다. 개별 플레이어 세션을 보거나 여러 플레이어 세션의 데이터를 비교합니다. 게임 세션 세부 정보 페이지를 볼 때 플레이어 세션 데이터를 열 수도 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers 콘솔의 게임 및 플레이어 세션](#), Amazon GameLift Servers 개발자 안내서

2024년 4월 24일: Amazon GameLift Servers에서 컨테이너 플릿 미리 보기 출시

Amazon GameLift Servers는 컨테이너 플릿에 대한 미리 보기를 제공하여 휴대성, 확장성, 내결함성 및 민첩성을 개선했습니다.

컨테이너 플릿에서 Amazon EC2 인스턴스는 하나 이상의 컨테이너를 호스팅합니다. 이러한 컨테이너에는 종속 도구 및 구성을 포함하여 필요한 모든 요소를 게임 서버에 포함합니다. SDK, 소프트웨어 패키지 등이 종속 항목에 포함됩니다. 프라이빗 Amazon Elastic Container Registry에 컨테이너를 업로드하면 Amazon GameLift Servers가 컨테이너로 플릿을 채웁니다.

컨테이너 플릿에서 작동하려면 게임 서버가 Linux에서 실행되고 Server SDK 5.x와 통합되어야 합니다. 컨테이너 플릿에서는 CPU 유닛 및 메모리와 같은 리소스의 소비를 최적화할 수 있도록 호스팅 리소스를 미세 조정하여 제어할 수 있습니다. 컨테이너에 여러 게임 서버를 호스팅하여 리소스 사용을 줄일 수도 있습니다.

컨테이너 플릿에서 온디맨드 인스턴스 유형, 스케일링(자동 및 수동), 대기열, 매치메이킹 등 다른 유형의 플릿과 동일하게 유용한 기능을 사용할 수 있습니다. 컨테이너에 대한 일부 새 플릿과 함께 다른 플릿 유형과 동일한 지표를 얻을 수도 있습니다. 컨테이너 플릿은 다음 위치 리전의 플레이어에게 글로벌한 도달 범위를 제공합니다.

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

더 많은 리전과 로컬 영역에 도달하려면 다중 위치 컨테이너 플릿을 생성합니다.

자세히 알아보기:

- [Amazon GameLift Servers 컨테이너를 사용한 호스팅 관리](#), Amazon GameLift Servers 개발자 안내서
- [CreateContainerGroupDefinition](#), Amazon GameLift Servers API 참조

2024년 2월 13일: Amazon GameLift Servers의 SDK 개선 사항 출시 및 Unreal Engine용 Amazon GameLift Servers 플러그인 설치 간소화

업데이트된 SDK 버전:

- Go 서버 SDK 버전 5.1.0
- C# 서버 SDK 버전 5.1.2
- C++ 서버 SDK 버전 5.1.2

아래 사항이 개선되었습니다.

- 네트워크 중단 시 자동 재연결을 추가하여 SDK의 신뢰성을 개선했습니다.
- [Go] 이제 서버 파라미터를 사용 여부에 상관없이 `InitSDK()`를 호출할 수 있습니다. Amazon GameLift Servers 관리형 EC2 플릿에서 실행되는 게임 서버는 환경 변수에서 직접 서버 파라미터를 읽습니다. Amazon GameLift Servers Anywhere 플릿의 게임 서버는 서버 파라미터를 사용하여 `InitSDK()`를 호출해야 합니다.

업데이트된 플러그인 버전:

- Unreal Engine용 Amazon GameLift Servers 플러그인, 버전 1.1.0
- Unity용 Amazon GameLift Servers 플러그인, 버전 2.1.0
- Unreal용 C++ 서버 SDK 플러그인 버전 5.1.1
- Unity용 C# 서버 SDK 플러그인 버전 5.1.2

아래 사항이 개선되었습니다.

- [Unreal Engine용 Amazon GameLift Servers 플러그인] 설치 지침을 업데이트하고 패키징을 간소화했습니다. 이 플러그인에 Unreal용 C++ 서버 SDK의 최신 버전이 포함되어 있습니다.
- 최신 버전의 Amazon GameLift Servers용 서버 SDK를 지원하도록 플러그인을 업그레이드했습니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인과 게임 통합](#), Amazon GameLift Servers 개발자 안내서
- [Amazon GameLift Servers 플러그인 및 SDK 다운로드](#)

2023년 12월 14일: 활성 게임 세션의 게임 속성을 업데이트할 수 있는 Amazon GameLift Servers 기능 추가

기존까지는 게임 세션 생성 시 게임 속성을 설정하고 게임 세션에서 지정된 속성을 검색할 수 있었습니다. 이제는 활성 게임 세션에서 이러한 속성을 추가하고 업데이트할 수 있습니다.

예를 들어 플레이어는 재생하려는 맵에 투표합니다. 게임 클라이언트가 UpdateGameSession을 호출하여 GameProperty 값을 {"Key": "map", "Value": "jungle"}로 수정합니다. 그러면 게임 세션에서 플레이어에 대한 새 맵을 구현합니다.

게임 관리자가 SearchGameSessions 작업을 사용하여 게임 속성에서 유용한 데이터를 검색할 수도 있습니다. 예를 들어 관리자가 Status 값이 ACTIVE이고 게임 속성이 {"Key": "map", "Value": "desert"}인 게임 세션을 나열할 수 있습니다.

자세히 알아보기:

- [the section called “게임 클라이언트 기능 통합”](#), Amazon GameLift Servers 개발자 안내서
- [GameProperty](#), Amazon GameLift Servers API 참조
- [UpdateGameSession](#), Amazon GameLift Servers API 참조
- [SearchGameSessions](#), Amazon GameLift Servers API 참조

2023년 11월 21일:에서 기반 Terraform 및 Pulumi와 같은 코드형 인프라 도구에 대한 지원을 Amazon GameLift Servers 시작합니다. AWS Cloud Control API

이제 코드형 인프라(IaC) 도구를 사용하여 전체 Amazon GameLift Servers 리소스 스택을 관리할 수 있습니다. 이러한 도구에는 AWS CloudFormation과 Terraform 및 Pulumi와 같은 타사 도구도 포함됩니다. 이번 추가 지원을 통해 게임 빌드에 집중하고 DevOps 전략을 활용하여 리소스 관리, CI/CD 및 고객에 대한 배포를 처리할 수 있습니다.

이제 AWS Cloud Control API를 사용하여 모든 Amazon GameLift Servers 리소스 유형을 프로비저닝하고 구성할 수도 있습니다. Amazon GameLift Servers APIs 또는 용 AWS CloudFormation 템플릿을 사용하여 리소스를 계속 사용할 수 있습니다Amazon GameLift Servers.

IaC로 사용가능한 Amazon GameLift Servers 리소스에 대한 자세한 내용은 [Amazon GameLift Servers 리소스 유형 참조](#)의 Amazon GameLift Servers 리소스 유형 참조를 확인하세요.

또한 이제 새 플릿 속성을 사용하여 AWS CloudFormation 템플릿 또는 AWS Cloud Control API를 사용하여 [플릿](#)을 자동으로 확장할 수 있습니다ScalingPolicies.

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 APIs 세트를 제공합니다.

자세히 알아보기:

- [AWS CloudFormation](#)
- [AWS 클라우드 제어 API](#)
- [AWS CC Terraform 공급자](#)
- [Pulumi](#)

2023년 11월 16일: Amazon GameLift Servers의 Unity용 독립 실행형 플러그인 업데이트

업데이트된 SDK 버전: Unity용 Amazon GameLift Servers 플러그인 버전 2.0.0

Unity용 Amazon GameLift Servers 플러그인은 Amazon GameLift Servers를 사용하여 Unity 게임을 시작하고 실행하는 단계를 간소화하는 도구와 워크플로를 제공합니다. Amazon GameLift Servers는 게임 개발자가 세션 기반 멀티플레이어 게임을 위한 전용 게임 서버를 관리하고 확장하는 데 사용할 수 있는 완전 관리형 서비스입니다.

이 버전에서는 서버 SDK 버전 5.x와 Amazon GameLift Servers Anywhere를 사용한 로컬 테스트 지원을 포함한 최신 Amazon GameLift Servers 기능을 사용하도록 Unity용 플러그인이 업데이트되었습니다. 플러그인은 Unity 버전 Unity 2021.3 LTS 및 2022.3 LTS와 호환됩니다.

플러그인 주요 기능은 다음과 같습니다.

- 다음 시나리오에 대한 Unity 편집기의 UI 워크플로 가이드:
 - 로컬 워크스테이션을 호스트로 사용하여 Amazon GameLift Servers와의 게임 통합을 테스트합니다. 이 워크플로 로컬 시스템에 대한 Amazon GameLift Servers Anywhere 플릿을 설정하고, 게임 서버 및 클라이언트의 인스턴스를 시작하며, Amazon GameLift Servers를 통해 게임 세션을 요청하고, 게임에 참여할 수 있습니다.
 - Amazon GameLift Servers 관리형 EC2 및 지원 AWS 리소스를 사용하여 통합 게임 서버에 클라우드 호스팅 솔루션을 배포합니다. 이 워크플로는 클라우드 호스팅을 위해 게임을 구성하는 데 도움이 되며 세 가지 배포 시나리오를 제공합니다.
 - 게임 서버를 단일 플릿에 배포합니다.
 - 게임 서버를 여러 AWS 리전의 저렴한 스팟 플릿 세트에 배포합니다.
 - FlexMatch 매치메이커를 사용하여 게임 서버를 배포합니다.

- AWS 계정 사용자에게 연결되는 사용자 프로필을 설정하고 기본 AWS 리전을 설정하는 기능입니다. 여러 계정, AWS 계정 사용자 및 리전에서 작동하도록 여러 프로필을 유지할 수 있습니다.
- Amazon GameLift Servers 통합 및 배포 프로세스를 간소화하는 데 도움이 되는 특별한 편의 기능은 다음과 같습니다.
 - 각 호스팅 솔루션에는 고유한 플레이어 IDs와 플레이어 검증을 제공하는 Amazon Cognito 사용자 풀을 포함한 지원 AWS 리소스가 포함되어 있습니다. 솔루션에는 스토리지용 Amazon S3 버킷, Amazon SNS 이벤트 알림, AWS Lambda 함수 및 기타 리소스도 포함됩니다.
 - Anywhere 워크플로의 경우 플러그인이 필요한 서버 파라미터 설정을 자동화합니다.
 - Amazon EC2 워크플로의 경우, 각 배포 솔루션은 Lambda 함수를 사용하여 내장된 클라이언트 백엔드 서비스를 제공합니다. 백엔드 서비스는 게임 클라이언트와 Amazon GameLift Servers 서비스 사이에 있으며 Amazon GameLift Servers 서비스에 대한 모든 직접 호출을 관리합니다.
- 게임 서버 및 게임 클라이언트 통합을 설명하는 간단한 샘플 멀티플레이어 게임의 자산 및 코드를 포함한 통합 테스트용 콘텐츠입니다.
- 자세한 통합 지침 및 샘플 코드가 포함된 플러그인 설명서.

Anywhere 및 Amazon EC2 플릿을 포함한 모든 배포 시나리오는 AWS CloudFormation 템플릿을 사용하여 게임 솔루션의 AWS 리소스를 설명하고 배포합니다. 이러한 템플릿은 Amazon GameLift Servers 플러그인 다운로드에 포함되어 있습니다. 그대로 사용하거나 게임에 맞게 변경할 수 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers Unity용 플러그인\(서버 SDK 5.x\)](#), Amazon GameLift Servers 개발자 안내서
- [GitHub에서 플러그인 다운로드](#)
- [Amazon GameLift Servers 호스팅 정보](#)
- [Amazon GameLift Servers 포럼](#)

2023년 11월 2일: 공유 자격 증명에 대한 Amazon GameLift Servers 지원 추가

업데이트된 SDK 버전: AWS SDK 1.11.193

새로운 Amazon GameLift Servers 공유 자격 증명 기능을 사용하면 관리형 EC2 플릿에 배포된 애플리케이션이 다른 AWS 리소스와 상호 작용할 수 있습니다. 이 업데이트는 Server SDK 버전 5.x 이상과 통합된 게임 서버 바이너리와 함께 번들 및 배포하는 애플리케이션에 적용됩니다. (게임 서버 실행 파일은 이미 Server SDK 5.x `GetFleetRoleCredentials()` 작업을 사용하여 자격 증명을 요청할 수 있습니다.)

예를 들어 Amazon CloudWatch 에이전트로 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 합니다. Amazon GameLift Servers는 게임 서버 빌드를 각 EC2 인스턴스에 배포할 때 공유 자격 증명 파일을 생성하고 인스턴스에 저장합니다. 인스턴스의 모든 애플리케이션은 공유 자격 증명을 사용할 수 있습니다. Amazon GameLift Servers는 인스턴스 수명 주기 동안 임시 자격 증명을 자동으로 새로 고칩니다.

다음 방법을 사용하여 관리형 EC2 플릿을 생성할 때 공유 자격 증명을 활성화할 수 있습니다.

- Amazon GameLift Servers 콘솔 플릿 생성 워크플로에서
- 새 파라미터 `InstanceRoleCredentialsProvider`를 사용하여 서비스 API 작업 `CreateFleet`을 호출하는 경우
- 파라미터를 `aws gamelift create-fleet` 사용하여 AWS CLI 작업을 호출하는 경우 `instance-role-credentials-provider`.

자세히 알아보기:

- [플릿의 다른 AWS 리소스와 통신](#), Amazon GameLift Servers 개발자 안내서
- [CreateFleet, InstanceRoleCredentialsProvider](#), Amazon GameLift Servers API 참조
- [IAM 서비스 역할 설정](#), Amazon GameLift Servers 개발자 안내서

2023년 9월 28일: Amazon GameLift Servers는 Unreal Engine용 새 독립형 플러그인 출시

업데이트된 SDK 버전: Unreal Engine용 Amazon GameLift Servers 플러그인 버전 1.0.0

Unreal Engine용 Amazon GameLift Servers 플러그인은 Amazon GameLift Servers를 사용하여 게임을 시작하고 실행하는 단계를 간소화하는 도구와 워크플로를 제공합니다. Amazon GameLift Servers는 게임 개발자가 세션 기반 멀티플레이어 게임을 위한 전용 게임 서버를 관리하고 확장하는 데 사용할 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 사용하여 Amazon GameLift Servers와의 게임 통합을 테스트합니다. 이 워크플로는 로컬 시스템에 대한 Amazon GameLift Servers Anywhere 플릿을 설정하고, 게

임 서버 및 클라이언트의 인스턴스를 시작하며, Amazon GameLift Servers를 통해 게임 세션을 요청하고, 새 게임 세션에 대한 연결 정보를 얻는 데 도움이 됩니다.

- 통합된 게임 서버를 위한 Amazon EC2 클라우드 호스팅 솔루션을 배포합니다. 이 워크플로는 게임을 클라우드 호스팅용으로 구성하는 데 도움이 되며, 다음과 같이 세 가지 다양한 배포 시나리오를 제공합니다. 첫 번째는 단일 플릿에 배포, 두 번째는 여러 리전의 스팟 플릿 세트에 배포, 세 번째는 FlexMatch 매치메이커를 사용하여 플릿 세트에 배포하는 것입니다. 각 배포 시나리오의 솔루션에는 Amazon GameLift Servers 리소스와 지원 AWS 리소스가 포함됩니다.
- AWS 계정 사용자에게 연결되는 사용자 프로필을 설정하고 기본 AWS 리전을 정의하는 기능입니다. 여러 계정, AWS 계정 사용자 및 리전에서 작동하도록 여러 프로필을 유지할 수 있습니다.
- Amazon GameLift Servers 통합 및 배포 프로세스를 간소화하는 데 도움이 되는 특별한 편의 기능은 다음과 같습니다.
 - 각 호스팅 솔루션에는 고유한 플레이어 IDs를 제공하는 기본 Amazon Cognito 사용자 풀, 스토리지용 Amazon S3 버킷, Amazon SNS 이벤트 알림 및 AWS Lambda 함수를 포함한 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 수신하여 Amazon GameLift Servers 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.

Anywhere 및 Amazon EC2 플릿을 포함한 모든 배포 시나리오에서는 AWS CloudFormation 템플릿을 사용하여 솔루션을 설명합니다. 플러그인은 게임에 대한 Amazon GameLift Servers 리소스를 배포할 때 이러한 템플릿을 사용합니다. 이러한 템플릿은 Amazon GameLift Servers 플러그인 다운로드에 포함되어 있으며 편집할 수 있습니다. 그대로 사용하거나 게임에 맞게 수정할 수 있습니다.

자세히 알아보기:

- [Unreal Engine용 Amazon GameLift Servers 플러그인](#), Amazon GameLift Servers 개발자 안내서
- [GitHub에서 플러그인 다운로드](#)
- [Amazon GameLift Servers 호스팅 정보](#)
- [Amazon GameLift Servers 포럼](#)

2023년 8월 17일: AWS Graviton 프로세서를 사용한 게임 서버 호스팅 Amazon GameLift Servers 제공

업데이트된 SDK 버전: AWS SDK 1.11.144

를 Amazon GameLift Servers 사용하면 이제 AWS Graviton 프로세서가 있는 EC2 인스턴스를 사용하여 클라우드에서 게임을 호스팅할 수 있습니다. Arm64-based 프로세서 AWS 를 사용하여에서 설계한 Graviton 인스턴스는 EC2를 사용하여 클라우드 워크로드에 최상의 가격 대비 성능을 제공하며 유사한 x86 기반 인스턴스에 비해 최대 40% 개선되었습니다. 최신 Graviton3 프로세서는 이전 버전보다 최대 25% 더 나은 컴퓨팅 성능을 제공합니다.

이제를 사용하여 AWS Graviton 패밀리와 새 인스턴스 중에서 선택할 Amazon GameLift Servers 수 있습니다.

- Graviton2 기반 인스턴스: c6g, c6gn, r6g, m6g, g5g
- Graviton3 기반 인스턴스: c7g, r7g, m7g

자세히 알아보기:

- [AWS Graviton 프로세서](#): Graviton 기반 EC2 인스턴스의 이점과 실제 사용에 대해 알아봅니다.
- [Graviton 시작하기](#): Graviton 기반 인스턴스에 대한 개요를 알아보고 운영 체제, 언어, 런타임에 따라 Graviton 기반 인스턴스에서 애플리케이션이 실행되는 방식에 대한 인사이트를 얻을 수 있습니다.

Note

Graviton Arm 인스턴스의 경우 Linux AMI에서 서버 빌드를 사용해야 합니다. 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.

2023년 7월 27일: Amazon GameLift Servers, Unity 개발에 대한 지원이 추가된 서버 SDK 5.1.0 출시

업데이트된 SDK 버전: C++, C#/Unity, Unreal 5.1.0용 Server SDK

Amazon GameLift Servers 서버 SDK의 최신 릴리스는 C++, C#, Unreal 플러그인에 대한 업데이트와 Unity 게임 엔진과 함께 사용할 수 있는 새로운 플러그인을 제공합니다. 게임 개발자는 Amazon

GameLift Servers 서버 SDK를 Amazon GameLift Servers에서 호스팅하기 위해 배포하는 게임 서버에 통합합니다.

최신 서버 SDK 버전에는 여러 고객 요청을 포함하는 다음과 같은 업데이트가 포함되어 있습니다.

- 언어별 SDK 패키지 다운로드 - 업데이트된 [Amazon GameLift Servers 다운로드 사이트](#)에는 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 UnityNuGet을 통한 필수 종속 항목이 포함되어 있습니다. 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 Servers 관리형 EC2 플릿에서 실행되는 게임 서버는 환경 변수에서 직접 서버 파라미터를 읽습니다. Amazon GameLift Servers Anywhere 플릿의 게임 서버는 서버 파라미터를 사용하여 InitSDK()를 호출해야 합니다.
 - [C++, C#, Unreal] Server SDK 호출을 통해 오류 메시지를 개선했습니다.
 - [C++ SDK] 서버 SDK 빌드 시간을 개선하기 위해 -DRUN_CLANG_FORMAT 빌드 플래그가 기본적으로 비활성화되어 있습니다. -DRUN_CLANG_FORMAT=1을 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.
 - [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)
 - [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#)
 - [Amazon GameLift Servers용 C++\(Unreal\) 서버 SDK 5.x - 작업](#)
 - Unity 및 Unreal 플러그인을 위한 Server SDK 5 통합 안내서의 새 버전
 - [Amazon GameLift Servers를 Unity 프로젝트에 통합](#)

- [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#)
- 추가 설명서 업데이트
 - 사용 중인 Amazon GameLift Servers 서버 SDK 버전을 기반으로 하는 원격 액세스 절차를 명확히 하기 위해 Amazon GameLift Servers 서비스 API 작업 [GetComputeAccess](#) 및 [GetInstanceAccess](#)에 대한 설명서를 수정했습니다.
 - “대기 중”인 상태일 때 게임 세션 정보가 일시적으로 표시되는 방식을 설명하기 위해 [GameSessionPlacement](#)의 설명을 수정했습니다.

2023년 7월 13일: Amazon GameLift Servers의 플릿 하드웨어 지표 추가

이제 Amazon GameLift Servers 관리형 EC2 플릿의 하드웨어 성능 지표를 추적할 수 있습니다. 지표에는 CPU 사용률, 네트워크 트래픽 볼륨, 디스크 읽기/쓰기 활동에 대한 EC2 인스턴스 지표가 포함됩니다. Amazon GameLift Servers의 경우 이러한 지표는 플릿 위치에 있는 모든 활성 인스턴스를 설명합니다. AWS Management Console의 Amazon CloudWatch 대시보드를 사용하여 이러한 플릿 하드웨어 지표를 볼 수 있습니다. Amazon GameLift Servers 콘솔의 플릿 세부 정보에서도 이를 확인할 수 있습니다.

자세히 알아보기:

- [Amazon CloudWatch를 사용하여 Amazon GameLift Servers 모니터링](#)(플릿에 대한 지표), Amazon GameLift Servers 개발자 안내서

2023년 6월 29일: Amazon GameLift Servers가 Amazon Linux 2023에 대한 지원 시작

업데이트된 SDK 버전: AWS SDK 1.11.111

Amazon GameLift Servers 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 리전을 AWS 리전 제외한 모든에서 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 2026년 6월 30일까지 계속됩니다.

자세히 알아보기:

- [Amazon GameLift Servers Linux 서버 FAQ](#)

- [Amazon Linux 2 및 Amazon Linux 2023 비교](#)
- Amazon GameLift Servers API 참조 링크:
 - [AWS SDK 작업 CreateBuild](#)
 - [CLI 명령 upload-build](#)
 - [CLI 명령 create-build](#)

2023년 5월 25일: 드레이닝 인스턴스에서 게임 세션 배치를 제외하는 필터 Amazon GameLift Servers FleetIQ 추가

업데이트된 SDK 버전: AWS SDK 1.11.87

Amazon GameLift Servers 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 ClaimGameServer를 호출할 때 이 기능을 사용합니다. 새 FilterOption 파라미터를 추가하고 허용된 인스턴스 상태를 ACTIVE로만 설정합니다. 이에 대해 Amazon GameLift Servers FleetIQ는 사용 가능한 게임 서버를 검색하고 요청할 때 활성 인스턴스만 살펴봅니다.

자세히 알아보기:

- Amazon GameLift Servers API 참조의 [ClaimGameServer](#)
- Amazon GameLift Servers FleetIQ 개발자 안내서의 [FleetIQ 작동 방식](#)

2023년 5월 16일: Amazon GameLift Servers는 플릿에 대한 비용 할당 태깅 지원

Amazon GameLift Servers 이제 고객은 AWS Billing 비용 할당 태그를 사용하여 게임 호스팅 비용을 구성할 수 있습니다. 개별 Amazon GameLift Servers EC2 플릿 리소스에 비용 할당 태그를 할당하여 플릿이 전체 호스팅 비용에 어떻게 기여하는지 추적할 수 있습니다.

자세히 알아보기:

- [리소스 비용 및 사용률 도구](#)
- AWS Billing 사용 설명서의 [AWS 비용 할당 태그 사용](#)

2023년 4월 20일: Amazon GameLift Servers가 Windows Server 2016의 지원 시작

업데이트된 SDK 버전: AWS SDK 1.11.63

Amazon GameLift Servers 고객은 이제 Windows Server 2016 운영 체제를 사용하여 게임 서버를 호스팅할 수 있습니다. 이 운영 체제는 모든에서 사용할 수 있습니다 AWS 리전. 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 Servers 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 Servers 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:

- Amazon GameLift Servers API 참조 링크:
 - [CLI 명령 `upload-build`](#)
 - [CLI 명령 `create-build`](#)
 - [AWS SDK 작업 `CreateBuild`](#)
- [Windows 2012용 Amazon GameLift Servers FAQ](#)

2023년 4월 13일: Amazon GameLift Servers가 Unreal용 서버 SDK 5.x 출시

업데이트된 SDK 버전: Unreal용 Server SDK 5.0.0

Unreal Engine용 Amazon GameLift Servers 경량 플러그인의 최신 버전은 이제 Amazon GameLift Servers 서버 SDK 5.x를 기반으로 합니다. Unreal Engine 환경과 Amazon GameLift Servers의 통합을 시작하려면 다음 링크를 참조하세요.

자세히 알아보기:

- [Amazon GameLift Servers를 Unreal Engine 프로젝트에 통합](#)
- [서버 SDK를 사용하여 게임 서버에 Amazon GameLift Servers를 추가합니다.](#)
- [Amazon GameLift Servers용 C++ 서버 SDK 5.x - 작업](#)

2023년 3월 14일: Amazon GameLift Servers 새 콘솔 경험 출시

새 Amazon GameLift Servers 콘솔에는 다음과 같은 개선 사항이 포함되어 있습니다.

- 탐색 개선 - 새 탐색 창을 사용하면 Amazon GameLift Servers 리소스 간의 탐색이 용이해집니다.
- Amazon GameLift Servers 랜딩 페이지 - 새 랜딩 페이지는 유용한 설명서로 연결되는 링크를 제공하고, Amazon GameLift Servers에 대한 개괄적인 개요를 표시하며, 설명서, 자주 묻는 질문, AWS re:Post에 대한 링크를 통해 지원을 제공합니다.
- Amazon CloudWatch 지표 개선 - 이제 Amazon GameLift Servers 콘솔과 CloudWatch 대시보드 모두에서 Amazon GameLift Servers 지표를 사용할 수 있습니다. 이 업데이트에는 성능, 사용률, 플레이어 세션에 대한 새로운 지표도 포함됩니다.

자세히 알아보기:

- [Amazon GameLift Servers를 사용하는 게임 호스팅 리소스 관리](#)
- [FlexMatch 매치메이커 빌드](#)

2023년 2월 14일: Amazon GameLift Servers는 이제 Amazon SNS 주제에 대한 서버 측 암호화 지원

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](#)
- [FlexMatch 매치메이킹 이벤트](#)
- [저장된 데이터 암호화](#)

2023년 2월 9일: Amazon GameLift Servers 서버 SDK는 C#10을 포함한 .NET 6 지원

업데이트된 SDK 버전: .NET 6용 Server SDK 5.0.0. SDK 업데이트는 필요하지 않습니다.

Unity Real-Time Development Platform을 사용하는 경우 Amazon GameLift Servers 서버 SDK 5.0.0 및 .NET 4.6을 계속 사용합니다. Unity는 .NET 6을 지원하지 않습니다.

자세히 알아보기:

- 최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.
- [Amazon GameLift Servers용 C# 서버 SDK 5.x - 작업](#)

2023년 1월 31일: Amazon GameLift Servers 서버 SDK는 Go 언어 지원

업데이트된 SDK 버전: Go용 Server SDK 5.0.0

자세히 알아보기:

- 최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.
- [Amazon GameLift Servers용 Go 서버 SDK - 작업](#)

2022년 12월 1일: Amazon GameLift Servers의 Amazon GameLift Servers Anywhere 및 Amazon GameLift Servers 서버 SDK 5.0 출시

업데이트된 SDK 버전: AWS SDK 1.10.21, C++ 및 C#용 Server SDK 5.0.0

Amazon GameLift Servers Anywhere는 게임 서버 리소스를 사용하여 Amazon GameLift Servers 게임 서버를 호스팅합니다. Amazon GameLift Servers Anywhere를 사용하여 자체 컴퓨팅 리소스를 Amazon GameLift Servers 관리형 EC2 컴퓨팅과 통합하여 게임 서버를 여러 컴퓨팅 유형에 분산할 수 있습니다. 또한 Amazon GameLift Servers Anywhere를 사용하면 반복할 때마다 Amazon GameLift Servers에 빌드를 업로드하지 않고도 게임 서버를 반복적으로 테스트할 수 있습니다.

주요 내용:

- 새 Amazon GameLift Servers Anywhere 플릿 및 컴퓨팅 유형
- Amazon GameLift Servers Anywhere 컴퓨팅 리소스 등록
- 테스트 반복 주기 개선

Amazon GameLift Servers 서버 SDK 5.0.0은 기존 서버 SDK와 새로운 리소스 유형인 컴퓨팅을 개선했습니다. 서버 SDK 5.0.0은 Amazon GameLift Servers Anywhere 및 게임 서버 호스팅을 위한 자체 컴퓨팅 리소스 사용을 지원합니다.

자세히 알아보기:

- [Amazon GameLift Servers용 서버 SDK 5.x](#)
- [지리적 위치](#)
- [관리형 플릿의 컴퓨팅 리소스 선택](#)
- [Amazon GameLift Servers Anywhere 플릿 생성](#)

2022년 8월 25일: Amazon GameLift Servers는 로컬 영역 지원 시작

업데이트된 SDK 버전: AWS SDK 1.9.333

Amazon GameLift Servers는 이제 미국의 8개 로컬 영역에서 사용할 수 있으므로 플릿을 플레이어 가까이 배치할 수 있습니다. 플릿에 로컬 영역을 추가하여 모든 관리형 Amazon GameLift Servers 기능을 로컬 영역에서 사용할 수 있습니다.

로컬 영역은 대규모 인구, 산업 및 정보 기술(IT) 센터와 가까운 클라우드 엣지로 AWS 리소스와 서비스를 확장합니다. 즉, 한 자릿수 밀리초 지연 시간이 필요한 애플리케이션을 최종 사용자나 온프레미스 데이터 센터 가까이 배포할 수 있게 되었습니다.

자세히 알아보기:

- [Amazon GameLift Servers 로컬 영역](#)
- [지리적 위치](#)
- [Amazon GameLift Servers 관리형 EC2 플릿 생성](#)

2022년 6월 28일: Amazon GameLift Servers가 새로운 옵트인 콘솔 환경 출시

새 Amazon GameLift Servers 콘솔에는 다음과 같은 개선 사항이 포함되어 있습니다.

- 탐색 개선 - 새 탐색 창을 사용하면 Amazon GameLift Servers 리소스 간의 탐색이 용이해집니다.
- Amazon GameLift Servers 랜딩 페이지 - 새 랜딩 페이지는 유용한 설명서로 연결되는 링크를 제공하고, Amazon GameLift Servers에 대한 개괄적인 개요를 표시하며, 설명서, 자주 묻는 질문, AWS re:Post에 대한 링크를 통해 지원을 제공합니다.

- Amazon CloudWatch 지표 개선 - 이제 Amazon GameLift Servers 콘솔과 CloudWatch 대시보드 모두에서 Amazon GameLift Servers 지표를 사용할 수 있습니다. 이 업데이트에는 성능, 사용률, 플레이어 세션에 대한 새로운 지표도 포함됩니다.

자세히 알아보기:

- [Amazon GameLift Servers를 사용하는 게임 호스팅 리소스 관리](#)
- [FlexMatch 매치메이커 빌드](#)

2022년 2월 15일: FlexMatch에 복합 규칙 및 개선 사항 추가

이제 FlexMatch 사용자는 다음 기능에 액세스할 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 Servers FlexMatch 개발자 안내서](#):
 - [FlexMatch 규칙 유형](#)
 - [FlexMatch 속성 표현식](#)
- [AttributeValue: SL](#)

2021년 10월 28일: 아시아 태평양(오사카) 리전의 다중 리전 플릿에 대한 지원 Amazon GameLift Servers 추가, AWS Graviton2 프로세서에 대한 지원 Amazon GameLift Servers FleetIQ 추가

업데이트된 SDK 버전: AWS SDK [1.9.133](#)

Amazon GameLift Servers(은)는 이제 아시아 태평양(오사카) 리전에서 사용할 수 있습니다. 이제 게임 개발자는 GameLift 다중 리전 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:

- Amazon GameLift Servers(은)는 이제 아시아 태평양(오사카) 리전에서 사용할 수 있습니다.
- 이제 Amazon GameLift Servers FleetIQ 게임 서버 그룹을 구성하여 Graviton2 인스턴스 패밀리 c6g, m6g, r6g를 관리할 수 있습니다.

자세히 알아보기:

- [Amazon GameLift Servers 다중 리전 플릿](#)
- [CreateGameServerGroup](#)
- [AWS graviton 프로세서](#)

2021년 9월 20일: Amazon GameLift Servers가 Unity용 플러그인 출시

라이브러리와 기본 UI가 포함된 Unity 버전 1.0.0용 Amazon GameLift Servers 플러그인을 사용하면 Amazon GameLift Servers 리소스에 쉽게 액세스하고 Amazon GameLift Servers를 Unity 게임에 통합할 수 있습니다. Unity용 Amazon GameLift Servers 플러그인을 사용하여 Amazon GameLift Servers APIs하고 일반적인 게임 시나리오를 위한 CloudFormation 템플릿을 배포할 수 있습니다. 플러그인에는 샘플 시나리오와 함께 작동하는 샘플 게임도 포함되어 있습니다. Amazon GameLift Servers Local을 사용하면 게임 클라이언트와 게임 서버 간에 전달되는 메시지를 확인하여 일반적인 게임이 Amazon GameLift Servers와 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 리전 플릿, 대기열 및 사용자 지정 매치메이커가 있는 다중 리전 플릿, 대기열 및 사용자 지정 매치메이커가 있는 스팟 플릿, 등 일반적인 게임 시나리오에 AWS CloudFormation 대한 샘플 시나리오를 배포합니다FlexMatch.

자세히 알아보기:

- [Unity용 Amazon GameLift Servers 플러그인과 게임 통합](#)

2021년 6월 30일: FlexMatch에 batchDistance 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: Amazon GameLift Servers Realtime Client 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 Unreal Version 4.26용 Amazon GameLift Servers 서버 SDK를 빌드할 수도 있습니다. 이 업데이트에는 C++ 및 C# 버전의 Amazon GameLift Servers Server SDK, Amazon GameLift Servers Local, Unreal Engine 플러그인 등 Windows 또는 Linux 게임 서버와 통합되는 구성 요소가 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 Servers 개발자 안내서](#):
 - [DescribePlayerSessions\(\)](#)
 - [GetInstanceCertificate\(\)](#)

2021년 3월 23일: Amazon GameLift Servers가 게임 세션 배치에 알림 추가

업데이트된 SDK 버전: AWS SDK [1.8.168](#)

이제 이벤트를 사용하여 게임 세션 대기열의 게임 세션 배치 활동을 모니터링할 수 있습니다. Amazon Simple Notification Service(SNS) 주제를 생성하여 이벤트 알림을 게시하거나 CloudWatch 이벤트를 사용하여 이벤트 추적을 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 Servers 개발자 안내서:
 - [게임 세션 배치의 이벤트 알림 설정](#) (new)
 - [게임 세션 배치 이벤트](#) (new)
- [API 참조\(AWS SDK\)](#)
 - 새 게임 세션 대기열 파라미터 NotificationTarget 및 CustomEventData: [GameSessionQueue](#), [CreateGameSessionQueue](#), [UpdateGameSessionQueue](#)
- [Amazon GameLift Servers 포럼](#)

2021년 3월 16일: Amazon GameLift Servers는 다중 리전 플릿, 6개의 새로운 리전 추가

업데이트된 SDK 버전: AWS SDK [1.8.163](#)

Amazon GameLift Servers 이제 21 AWS 리전에서 관리형 호스팅을 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 Servers 다중 위치 플릿 기능을 사용하면 이제 20개의 Amazon GameLift Servers가 지원되는 리전 중 일부 또는 전체에서 게임 서버를 호스팅하도록 단일 플릿을 설정할 수 있습니다(베이징 리전 제외). 이 기능은 Amazon GameLift Servers 호스팅 리소스를 전 세계적으로 설정하고 유지하는 데 필요한 작업을 크게 줄이는 것을 목표로 합니다. 다중 위치 플릿은 (버지니아 북부), us-east-1 (오레곤), us-west-2 (eu-central-1프랑크푸르트), eu-west-1 (아일랜드), ap-southeast-2 (시드니), ap-northeast-1 (도쿄), ap-northeast-2 (서울) AWS 리전에서 생성할 수 있습니다. 다른 모든 리전에서는 필요에 따라 단일 위치 플릿을 계속 설정할 수 있습니다. 이번 릴리스 이전에 생성된 모든 플릿은 단일 위치 플릿입니다. 다중 위치 플릿을 사용하는 것은 호스팅 비용에 영향을 주지 않습니다. Amazon GameLift Servers 요금은 사용하는 인스턴스의 유형, 위치 및 볼륨을 기반으로 합니다. (자세한 내용은 [Amazon GameLift Servers 요금](#)을 참조하세요.)멀티 로케이션 플릿에 대한 CloudFormation 지원이 곧 제공될 예정입니다.

Note

중국 리전에서는 다중 위치 플릿을 사용할 수 없습니다. 중국 리전에 있는 Amazon GameLift Servers 리소스는 다른 Amazon GameLift Servers 리전의 리소스와 상호 작용하거나 사용할 수 없습니다.

주요 내용:

- 다중 위치 플릿을 사용하여 원격 위치 목록을 명시적으로 추가합니다. Amazon GameLift Servers는 빌드 및 런타임 구성을 포함하여 동일한 유형 및 구성의 인스턴스를 플릿의 홈 리전과 추가된 모든 위치에 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 Servers 콘솔에서 호스팅 지표를 가져옵니다. 이 지표는 플릿의 모든 위치에 대해 집계되거나 각 플릿 위치별로 분류되어 있습니다.

자세히 알아보기:

- [Amazon Game Tech 블로그](#)
- [API 참조\(AWS SDK\)](#)
 - 새로운 플릿 위치 운영: [CreateFleetLocations](#), [DescribeFleetLocationAttributes](#), [DescribeFleetLocationCapacity](#), [DescribeFleetLocationUtilization](#), [DeleteFleetLocations](#)
 - 새로운 다중 위치 지원이 있는 업데이트된 플릿 작업: [CreateFleet](#), [UpdateFleetCapacity](#), [DescribeEC2InstanceLimits](#), [DescribeInstances](#), [StopFleetActions](#), [StartFleetActions](#)
 - 새 우선 순위 및 필터링 기능이 있는 업데이트된 게임 세션 배치 작업 대기열: [CreateGameSessionQueue](#), [DescribeGameSessionQueues](#), [UpdateGameSessionQueue](#)
 - 새로운 위치 지원이 있는 업데이트된 게임 세션 생성 작업: [CreateGameSession](#), [DescribeGameSessions](#), [DescribeGameSessionDetails](#), [SearchGameSessions](#)
- [Amazon GameLift Servers 개발자 안내서](#):
 - [Amazon GameLift Servers 서비스 위치](#) (업데이트됨)
 - [리소스 사용자 지정 호스팅](#) (new)
 - [Amazon GameLift Servers를 사용하여 게임 호스팅 용량 확장](#) (업데이트됨)
 - [게임 세션 대기열 사용자 지정](#) (new)
 - [Amazon GameLift Servers 콘솔의 플릿 세부 정보](#) (업데이트됨)
- [Amazon GameLift Servers 포럼](#)

2021년 2월 9일: Amazon GameLift Servers는 AMD 인스턴스, 독립형 FlexMatch에 대한 지원 확대

업데이트된 SDK 버전: AWS SDK [1.8.139](#)

다음은 이 릴리스의 업데이트 사항입니다.

- 이제 Amazon GameLift Servers FleetIQ 게임 서버 그룹을 구성하여 AMD 인스턴스 패밀리 C5a, M5a, R5a를 관리할 수 있습니다. GameServerGroup [InstanceDefinition](#)에 나열된 대로 지원되는 Amazon EC2 인스턴스 유형은 이제 다음을 포함합니다.

- 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: 용 AMD 인스턴스FleetIQ는 현재 중국(베이징) AWS 리전에서 사용할 수 없습니다. 중국의 [기능 가용성 및 구현 차이](#)를 참조하세요.

- 이제 Amazon GameLift Servers 관리형 게임 호스팅은 Sinnet이 운영하는 중국(베이징) 리전에서 AMD 인스턴스를 지원합니다. 새 AMD 인스턴스 패밀리에는 M5a 및 R5a가 포함됩니다. [InstanceType](#)에 나열된 대로 지원되는 EC2 인스턴스 유형은 이제 다음을 포함합니다.
 - 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
- 이제 Sinnet이 운영하는 중국(베이징) 리전에서 Amazon GameLift Servers FlexMatch를 독립형 매치메이킹 솔루션으로 사용할 수 있습니다. 고객은 베이징 리전에서 FlexMatch 매치메이커를 생성하고 [FlexMatchMode](#) 파라미터를 STANDALONE으로 구성할 수 있습니다. Amazon GameLift Servers 관리형 호스팅 또는 비 Amazon GameLift Servers 호스팅 솔루션을 사용하는 FlexMatch에 대한 자세한 내용은 [Amazon GameLift Servers FlexMatch 개발자 안내서](#)를 참조하세요.
- Amazon GameLift Servers FlexMatch에 대한 이벤트 알림을 설정할 때 이제 Amazon SNS FIFO 주제를 알림 대상으로 지정할 수 있습니다. 자세한 내용은 다음을 참조하세요.
 - [MatchmakingConfiguration NotificationTarget](#), Amazon GameLift Servers API 참조
 - [FlexMatch 이벤트 알림 설정](#), Amazon GameLift Servers FlexMatch 개발자 안내서
 - AWS 뉴스 블로그의 [Amazon SNS FIFO 소개 - 선착순 Pub/Sub 메시징](#)

2020년 12월 22일: Amazon GameLift Servers 서버 SDK는 Unreal Engine 4.25 및 Unity 2020 지원

업데이트된 SDK 버전: Amazon GameLift Servers 서버 SDK 4.0.2, Unreal 플러그인 버전 3.3.3

최신 버전의 Amazon GameLift Servers 서버 SDK에는 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.

최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.

2020년 11월 24일: 이제 Amazon GameLift Servers FlexMatch는 어디에서나 호스팅되는 게임으로 사용 가능

업데이트된 SDK 버전: AWS SDK [1.8.95](#)

Amazon GameLift Servers FlexMatch는 멀티플레이어 게임을 위한 사용자 지정 가능한 매치메이킹 서비스입니다. 처음에는 Amazon GameLift Servers 관리형 호스팅 사용자를 위해 설계된 FlexMatch를 이제 P2P, 전용 온프레미스 컴퓨팅, 클라우드 컴퓨팅 프리미티브 유형을 포함한 다른 호스팅 시스템을 사용하는 게임에 통합할 수 있습니다. Amazon EC2에서 게임을 호스팅하기 위해 Amazon GameLift Servers FleetIQ를 사용하는 게임은 이제 FlexMatch를 사용하여 매치메이킹을 구현할 수 있습니다.

FlexMatch는 주요 플레이어 특성과 보고된 지연 시간을 기반으로 플레이어가 서로 매치되도록 매치메이킹 프로세스를 사용자 정의할 수 있는 폭넓은 위도를 제공하는 강력한 매치메이킹 알고리즘과 규칙 언어를 제공합니다. 또한 FlexMatch는 플레이어 그룹, 플레이어 수락, 매치 채우기와 같은 기능을 지원하는 매치메이킹 요청 워크플로를 제공합니다. FlexMatch를 Amazon GameLift Servers 관리형 호스팅 또는 Amazon GameLift Servers Realtime과 함께 사용하는 경우 매치메이커는 자동으로 Amazon GameLift Servers를 사용하여 호스팅 리소스를 찾고 새로 구성된 매치를 위한 새 게임 세션을 시작합니다. FlexMatch를 독립형 서비스로 사용하는 경우 매치메이커는 매치 결과를 게임에 전달하면, 호스팅 솔루션을 사용하여 새 게임 세션을 시작할 수 있습니다.

에 대한 API 작업은 AWS SDK 및 AWS Command Line Interface ()에 포함된 Amazon GameLift Servers 서비스 API의 FlexMatch 일부입니다. 이번 릴리스에는 독립형 매치메이킹을 지원하는 다음과 같은 업데이트가 포함되어 있습니다.

- API 리소스 MatchmakingConfiguration에는 다음 변경 사항이 있습니다.
 - 새 속성 FlexMatchMode는 매치메이커가 Amazon GameLift Servers 관리형 호스팅과 함께 사용되는지, 독립형 매치메이킹으로 사용되는지 여부를 나타냅니다.
 - FlexMatchMode가 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 Amazon GameLift Servers에서 AMD 인스턴스 사용 가능

업데이트된 SDK 버전: AWS SDK [1.8.95](#)

Amazon GameLift Servers에서 지원하는 Amazon EC2 인스턴스 유형 목록에는 이제 C5a, M5a, R5a 의 세 가지 새로운 인스턴스 패밀리가 포함됩니다. 이러한 패밀리는 최대 3.3의 주파수에서 실행되는 AMD EPYC 프로세서로 구동되는 AMD 컴퓨팅 최적화 인스턴스로 구성됩니다. 사용합니다. AMD 인스턴스는 x86과 호환되므로, 현재 Amazon GameLift Servers에서 실행 중인 게임을 변경 없이 AMD 인스턴스 유형에 배포할 수 있습니다. 새 인스턴스는 미국 동부(버지니아 북부 및 오하이오), 미국 서부(오레곤 및 캘리포니아 북부), 캐나다 중부(몬트리올), 남아메리카(상파울루), EU 중앙부(프랑크푸르트), EU 서부(런던 및 아일랜드), 아시아 태평양 남부(뭄바이), 아시아 태평양 북동부(서울 및 도쿄), 아시아 태평양 남동부(싱가포르 및 시드니) AWS 리전에서 사용할 수 있습니다.

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](#)
- [Amazon GameLift Servers 인스턴스 요금](#)
- [Amazon EC2 인스턴스 기능 AMD EPYC 프로세서](#)
- [Amazon GameLift Servers 포럼](#)

2020년 11월 11일: Amazon GameLift Servers 서버 SDK에 버전 업데이트

업데이트된 SDK 버전: Amazon GameLift Servers Server 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.

최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.

2020년 11월 5일: 새로운 FlexMatch 알고리즘 사용자 지정

FlexMatch 사용자는 이제 매치메이킹 프로세스에 대해 다음과 같은 기본 동작을 조정할 수 있습니다. 이러한 사용자 지정은 매치메이킹 규칙 세트에서 설정됩니다. Amazon GameLift Servers 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: Amazon GameLift Servers의 서버 SDK 업데이트

업데이트된 SDK 버전: Amazon GameLift Servers Server 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.

- 이제 [GetTerminationTime\(\)](#) 작업은 AwsDateTimeOutcome 데이터 형식의 값을 반환합니다.
- 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 OnUpdateGameSession이 [FProcessParameters](#)에 추가되었습니다.

최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.

2020년 8월 27일: Amazon EC2를 사용한 게임 호스팅용 Amazon GameLift Servers FleetIQ(일반 공급)

업데이트된 SDK 버전: AWS SDK [1.8.36](#)

Amazon EC2에서 저렴한 클라우드 기반 게임 호스팅을 위한 Amazon GameLift Servers FleetIQ 솔루션을 이제 정식 버전으로 사용할 수 있습니다. Amazon GameLift Servers FleetIQ는 게임 호스팅의 실행 가능성을 최적화하여 Amazon EC2 스팟 인스턴스에서 직접 게임 서버를 호스팅할 수 있는 기능을 개발자에게 제공합니다. 게임 개발자는 Amazon GameLift Servers 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 가용성 릴리스에는 Amazon GameLift Servers FleetIQ 솔루션에 대한 다음과 같은 업데이트가 포함되어 있습니다.

- 새 API 작업은 Amazon GameLift Servers FleetIQ 게임 서버 그룹의 모든 활성 인스턴스에 대한 상태를 포함한 정보를 DescribeGameServerInstances 반환합니다.
- 새로운 밸런싱 전략 ON_DEMAND_ONLY는 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: Amazon GameLift Servers에서 Unity 및 Unreal Engine에 대한 서버 SDK를 업데이트함

업데이트된 SDK 버전: Amazon GameLift Servers 서버 SDK 4.0.0, Amazon GameLift Servers Local 1.0.5

최신 버전의 Amazon GameLift Servers Server SDK에는 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.
- C# 서버 SDK 버전 4.0.0을 사용하는 통합을 테스트하도록 업데이트된 Amazon GameLift Servers Local 버전 1.0.5.

최신 버전의 Amazon GameLift Servers 서버 SDK는 [Amazon GameLift Servers 시작하기](#)에서 다운로드합니다.

2020년 4월 2일: EC2에서 게임 호스팅에 Amazon GameLift Servers FleetIQ 사용 가능 (공개 미리 보기)

업데이트된 SDK 버전: AWS SDK [1.7.310](#)

Amazon GameLift Servers FleetIQ 기능은 게임 호스팅에 사용할 수 있도록 저렴한 스팟 인스턴스의 실행 가능성을 최적화합니다. 이제 관리형 Amazon GameLift Servers 서비스를 이용하지 않고 호스팅 리소스를 직접 관리하려는 고객을 위해 이 기능을 확장했습니다. 이 솔루션은 컨테이너 또는 AWS Shield 및 Amazon Elastic Container Service(Amazon ECS)와 같은 기타 AWS 서비스의 사용을 지원합니다.

자세히 알아보기:

[Amazon GameLift Servers FleetIQ의 GameTech 블로그 게시물](#)

2019년 12월 19일: AWS 리소스에 대한 Amazon GameLift Servers 리소스 관리 개선

업데이트된 SDK 버전: AWS SDK [1.7.249](#)

이제 AWS 리소스와 함께 리소스 관리 도구를 활용할 수 Amazon GameLift Servers 있습니다. 특히 빌드, 스크립트, 플릿, 게임 세션 대기열, 매치메이킹 구성, 매치메이킹 규칙 세트 등 모든 주요 Amazon GameLift Servers 리소스에 Amazon 리소스 이름(ARN) 값이 할당됩니다. 리소스 ARN은 모든 AWS 리전에서 고유한 일관된 식별자를 제공합니다. 리소스별 AWS Identity and Access Management (IAM) 권한 정책을 생성하는 데 사용할 수 있습니다. 이제 리소스에는 ARN과 리전별로 고유하지 않은 기존 리소스 식별자가 할당됩니다.

또한 Amazon GameLift Servers 리소스는 이제 태그 지정을 지원합니다. 태그를 사용하여 리소스를 구성하고, IAM 권한 정책을 생성하여 리소스 그룹에 대한 액세스를 관리하고, AWS 비용 내역을 사용자 지정할 수 있습니다. Amazon GameLift Servers 리소스에 대한 태그를 관리할 때는 Amazon GameLift Servers API 작업 `TagResource()`, `UntagResource()` 및 `ListTagsForResource()`를 사용합니다.

자세히 알아보기:

- Amazon GameLift Servers API 참조의 [TagResource](#)
- AWS 일반 참조의 [AWS 리소스 태깅](#)
- AWS AWS General Reference의 [Amazon 리소스 이름](#)

2019년 11월 14일: 중국(베이징) 리전의 새 CloudFormation 템플릿, 업데이트

업데이트된 SDK 버전: AWS SDK [1.7.210](#)

CloudFormation 용 템플릿 Amazon GameLift Servers

Amazon GameLift Servers 이제 리소스를 생성하고 관리할 수 있습니다 CloudFormation. 기존 CloudFormation 빌드 및 플릿 템플릿이 현재 리소스에 맞게 업데이트되었으며 이제 스크립트, 대기열, 매치메이킹 구성 및 매치메이킹 규칙 세트에 새 템플릿을 사용할 수 있습니다. CloudFormation 템플릿은 특히 여러 리전에 게임을 배포할 때 관련 AWS 리소스 그룹을 관리하는 작업을 크게 간소화합니다.

자세히 알아보기:

- AWS CloudFormation 사용 설명서의 [Amazon GameLift Servers 리소스 유형 참조](#)
- Amazon GameLift Servers 개발자 안내서의 [를 사용하여 Amazon GameLift Servers 호스팅 리소스 관리 CloudFormation](#)

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.