aws

FlexMatch 개발자 가이드

아마존 GameLift

버전

아마존 GameLift: FlexMatch 개발자 가이드

Table of Contents

아마존이란 GameLift FlexMatch 무엇입니까?

GameLift FlexMatch Amazon은 멀티플레이어 게임을 위한 맞춤형 매치메이킹 서비스입니다. 를 사용하면 게임에 맞는 멀티플레이어 매치의 모습을 정의하고 각 매치에서 호환되는 플레이어를 평가하고 선택하는 방법을 결정하는 사용자 지정 규칙 세트를 만들 수 있습니다. FlexMatch 또한 매치메이킹 알고리즘의 주요 측면을 게임 요구 사항에 맞게 세밀하게 조정할 수 있습니다.

독립형 매치메이킹 FlexMatch 서비스로 사용하거나 Amazon GameLift 게임 호스팅 솔루션과 통합하여 사용할 수 있습니다. 예를 들어 peer-to-peer 아키텍처가 있는 게임이나 다른 클라우드 컴퓨팅 솔루션을 사용하는 게임에 독립형 기능으로 구현할 FlexMatch 수 있습니다. 또는 Amazon GameLift 관리형 EC2 호스팅 또는 FlexMatch Amazon을 통한 온프레미스 호스팅을 추가할 수도 있습니다. GameLift Anywhere 이 안내서는 특정 시나리오에 맞는 FlexMatch 매치메이킹 시스템을 구축하는 방법에 대한 자세한 정보를 제공합니다.

FlexMatch 게임 요구 사항에 따라 매치메이킹 우선 순위를 유연하게 설정할 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

- 매치 속도와 품질 사이의 균형을 찾습니다. 매치 규칙을 설정하여 충분히 좋은 매치를 빠르게 찾거나, 플레이어가 최적의 플레이어 경험을 위해 가능한 최상의 경기를 찾도록 조금 더 대기하게 합니다.
- 잘 매치되는 플레이어나 잘 매치된 팀을 기반으로 매치를 설정합니다. 모든 플레이어가 실력이나 경험 등 비슷한 특성을 가진 경기를 만들어 보세요. 또는 각 팀의 특성을 종합하여 공통 기준을 충족하는 매치를 구성하세요.
- 플레이어 지연 시간이 매치메이킹에 어떤 영향을 미치는지 우선순위를 정하세요. 모든 플레이어의 지연 시간을 엄격하게 제한하고 싶으신가요? 아니면 매치에 참여하는 모든 플레이어의 지연 시간이 비슷하다면 지연 시간을 늘려도 괜찮을까요?

> 작업을 시작할 준비가 되셨나요? FlexMatch
>
> 게임 설치 및 실행에 대한 step-by-step 지침은 다음 주제를 참조하십시오. FlexMatch
>
> - Amazon GameLift 호스팅을 사용한 FlexMatch 통합
> - 독립형 매치메이킹을 위한 Amazon GameLift FlexMatch 통합

주요 FlexMatch 기능

다음 기능은 독립형 서비스로 사용하든 Amazon GameLift 게임 호스팅과 함께 FlexMatch 사용하든 관계없이 모든 FlexMatch 시나리오에서 사용할 수 있습니다.

- 사용자 지정 플레이어 매칭. 플레이어에게 제공하는 모든 게임 모드에 적합한 매치메이커를 설계하고 구축합니다. 주요 플레이어 속성(예: 스킬 레벨 또는 역할)과 지리적 지연 시간 데이터를 평가하기 위한 사용자 지정 규칙 세트를 구축하고 게임에서 훌륭한 플레이어 매치를 형성합니다.
- 지연 시간 기반 매칭. 플레이어 지연 시간 데이터를 제공하고 매치에 참가한 플레이어들이 비슷한 응답 시간을 갖도록 하는 매치 규칙을 만들 수 있습니다. 이 기능은 플레이어 매치메이킹 풀이 여러 지리적 리전에 걸쳐 있는 경우에 유용합니다.
- 최대 200명의 플레이어까지 매치 규모 지원. 게임에 맞게 사용자 지정된 매치 규칙을 사용하여 최대 40명의 플레이어로 매치를 만들 수 있습니다. 간소화된 사용자 지정 매칭 프로세스와 플레이어의 대기 시간을 관리할 수 있는 매칭 프로세스를 사용하여 최대 200명의 플레이어로 구성된 매치를 생성합니다.
- 플레이어 수락. 매치를 마무리하고 게임 세션을 시작하기 전에 플레이어가 제안된 매치에 참여하도록 합니다. 이 기능을 사용하면 매치를 위한 새 게임 세션을 FlexMatch 시작하기 전에 사용자 지정 수락 워크플로를 시작하고 플레이어 응답을 보고할 수 있습니다. 모든 플레이어가 매치를 수락하지 않으면 제안된 매치는 실패하고 수락한 플레이어는 자동으로 매치메이킹 풀로 돌아갑니다.
- 플레이어 그룹 지원. 같은 팀에서 함께 플레이하고자 하는 플레이어 그룹을 위한 매치를 생성합니다. 필요에 따라 매치를 채울 추가 플레이어를 찾는 FlexMatch 데 사용합니다.
- 확장 가능한 매칭 규칙. 성공적인 매치를 찾지 못한 상태에서 일정 시간이 지나면 매치 요건이 점차 완화됩니다. 규칙 확장을 통해 초기 매치 규칙을 완화할 장소와 시기를 결정할 수 있으므로 플레이어가 더 빨리 플레이 가능한 게임에 참여할 수 있습니다.
- 매치 채우기. 기존 게임 세션에서 빈 플레이어 슬롯을 잘 매치된 새 플레이어로 채웁니다. 새 플레이어를 요청하는 시기와 방법을 사용자 지정하고 동일한 사용자 지정 매치 규칙을 사용하여 추가 플레이어를 찾을 수 있습니다.

FlexMatch 아마존 GameLift 호스팅으로

FlexMatch 은 Amazon에서 호스팅하는 게임에 사용할 수 있는 다음과 같은 추가 기능을 제공합니다 GameLift. 여기에는 사용자 지정 게임 서버 또는 실시간 서버가 있는 게임이 포함됩니다.

- 게임 세션 배치. 매치가 성공적으로 이루어지면 Amazon에 새 게임 세션 배치를 FlexMatch 자동으로 GameLift 요청합니다. 플레이어 ID 및 팀 배정을 포함하여 매치메이킹 중에 생성된 데이터는 게임 서

버에 제공되므로 해당 정보를 사용하여 매치를 위한 게임 세션을 시작할 수 있습니다. FlexMatch 그런 다음 게임 세션 연결 정보를 다시 전달하여 게임 클라이언트가 게임에 참여할 수 있도록 합니다. 매치에서 플레이어가 겪는 지연 시간을 최소화하기 위해 Amazon에서의 게임 세션 배치는 지역별 플레이어 지연 시간 데이터 (제공된 경우) 를 GameLift 사용할 수도 있습니다.

- 자동 매치 채우기. 이 기능을 활성화하면 플레이어 슬롯이 채워지지 않은 상태에서 새 게임 세션이 시작될 때 매치 백필 요청을 FlexMatch 자동으로 보냅니다. 매치메이킹 시스템은 최소 플레이어 수로 게임 세션 배치 프로세스를 시작한 다음 나머지 슬롯을 빠르게 채웁니다. 자동 채우기를 사용하여 매치된 게임 세션에서 탈락한 플레이어를 교체할 수는 없습니다.

Amazon GameLift Elastic Compute Cloud (Amazon EC2) 리소스로 호스팅되는 게임에 Amazon FleetiQ를 사용하는 경우 독립 실행형 서비스로 구현하십시오. FlexMatch

아마존 요금 GameLift FlexMatch

Amazon은 사용 기간별로 인스턴스 GameLift 요금을 부과하고 전송된 데이터 수량을 기준으로 대역폭을 청구합니다. Amazon GameLift 서버에서 게임을 호스팅하는 경우 Amazon 수수료에는 FlexMatch 사용량이 포함됩니다 GameLift. 다른 서버 솔루션에서 게임을 호스팅하는 경우 FlexMatch 사용 요금이 별도로 부과됩니다. Amazon의 전체 요금 및 가격 목록은 Amazon GameLift 가격을 참조하십시오. GameLift

Amazon에서의 게임 호스팅 또는 매치메이킹 비용 계산에 대한 자세한 내용은 사용 방법을 설명하는 Amazon 예상 GameLift 요금 생성을 참조하십시오. GameLift AWS Pricing Calculator

Amazon GameLift FlexMatch 작동 방식

이 주제에서는 FlexMatch 시스템의 핵심 구성 요소와 이러한 구성 요소가 상호 작용하는 방식을 포함하여 Amazon GameLift FlexMatch 서비스에 대한 개요를 제공합니다.

FlexMatch는 Amazon GameLift 관리형 호스팅을 사용하는 게임 또는 다른 호스팅 솔루션을 사용하는 게임에 사용할 수 있습니다. Realtime 서버를 포함하여 Amazon GameLift 서버에서 호스팅되는 게임은 통합된 Amazon GameLift 서비스를 사용하여 사용 가능한 게임 서버를 자동으로 찾고 매치를 위한 게임 세션을 시작합니다. Amazon GameLift FleetIQ를 포함하여 FlexMatch를 독립형 서비스로 사용하는 게임은 기존 호스팅 시스템과 협력하여 호스팅 리소스를 할당하고 매치를 위해 게임 세션을 시작해야 합니다.

게임에 FlexMatch를 설정하는 방법에 대한 자세한 지침은 FlexMatch 시작하기 섹션을 참조하세요.

매치메이킹 구성 요소

FlexMatch 매치메이킹 시스템은 다음 구성 요소 중 일부 또는 전체를 포함합니다.

Amazon GameLift 구성 요소

다음은 FlexMatch 서비스가 게임에 대한 매치메이킹을 수행하는 방식을 제어하는 Amazon GameLift 리소스입니다. 콘솔과 AWS CLI를 비롯한 Amazon GameLift 도구를 사용하거나 Amazon GameLift용 AWS SDK를 사용하여 프로그래밍 방식으로 생성 및 유지 관리합니다.

- FlexMatch 매치메이킹 구성(매치메이커라고도 함) - 매치메이커는 게임의 매치메이킹 프로세스를 사용자 지정하는 구성 값 세트입니다. 게임에는 매치메이커가 여러 개 있을 수 있으며, 각 매치메이커는 필요에 따라 서로 다른 게임 모드 또는 경험에 맞게 구성됩니다. 게임에서 FlexMatch에 매치메이킹 요청을 보낼 때 사용할 매치메이커를 지정합니다.
- FlexMatch 매치메이킹 규칙 세트 - 규칙 세트에는 잠재적 매치에 대해 플레이어를 평가하고 승인 또는 거부하는 데 필요한 모든 정보가 들어 있습니다. 규칙 세트는 매치의 팀 구조를 정의하고, 평가에 사용되는 플레이어 속성을 선언하며, 허용 가능한 매치의 기준을 설명하는 규칙을 제공합니다. 규칙은 개별 플레이어, 팀 또는 전체 매치에 적용될 수 있습니다. 예를 들어 규칙에 따라 매치에 참여하는 모든 플레이어가 같은 게임 맵을 선택하도록 하거나 모든 팀의 플레이어 평균 실력이 비슷해야 할 수 있습니다.
- Amazon GameLift 게임 세션 대기열(Amazon GameLift 관리형 호스팅만을 사용하는 FlexMatch의 경우) - 게임 세션 대기열은 사용 가능한 호스팅 리소스를 찾고 해당 매치를 위한 새 게임 세션을 시작합니다. 대기열의 구성에 따라 Amazon GameLift가 사용 가능한 호스팅 리소스를 찾는 위치와 매치에 가장 적합한 호스트를 선택하는 방법이 결정됩니다.

사용자 지정 구성 요소

다음 구성 요소에는 게임 아키텍처를 기반으로 구현해야 하는 완전한 FlexMatch 시스템에 필요한 기능이 포함되어 있습니다.

- 매치메이킹을 위한 플레이어 인터페이스 - 이 인터페이스를 통해 플레이어는 매치에 참가할 수 있습니다. 최소한 클라이언트 매치메이킹 서비스 구성 요소를 통해 매치메이킹 요청을 시작하고, 매치메이킹 프로세스에 필요한 경우 스킬 레벨 및 지연 시간 데이터와 같은 플레이어별 데이터를 제공합니다.

> **Note**
>
> FlexMatch 서비스와의 통신은 게임 클라이언트가 아닌 백엔드 서비스를 통해 수행하는 것이 가장 좋습니다.

- 클라이언트 매치메이킹 서비스 - 이 서비스는 플레이어 인터페이스에서 플레이어 참가 요청을 처리하고, 매치메이킹 요청을 생성하며, FlexMatch 서비스에 전송합니다. 요청이 처리되면 매치메이킹 이벤트를 모니터링하고, 매치메이킹 상태를 추적하며, 필요에 따라 조치를 취합니다. 게임에서 게임 세션 호스팅을 관리하는 방식에 따라, 이 서비스는 게임 세션 연결 정보를 플레이어에게 반환할 수 있습니다. 이 구성 요소는 Amazon GameLift API와 함께 AWS SDK를 사용하여 FlexMatch 서비스와 통신합니다.
- 매치 배치 서비스(독립형 서비스인 FlexMatch에만 해당) - 이 구성 요소는 기존 게임 호스팅 시스템과 함께 작동하여 사용 가능한 호스팅 리소스를 찾고 매치를 위한 새 게임 세션을 시작합니다. 구성 요소는 매치메이킹 결과를 가져와야 하며, 매치에 참여한 모든 플레이어의 플레이어 ID, 속성, 팀 배정 등 새 게임 세션을 시작하는 데 필요한 정보를 추출해야 합니다.

FlexMatch 매치메이킹 프로세스

이 주제에서는 기본 매치메이킹 시나리오와 다양한 게임 구성 요소와 FlexMatch 서비스 간의 상호 작용에 대해 설명합니다.

플레이어에 대해 매치메이킹 요청

게임 클라이언트를 사용하는 플레이어가 “게임 참여” 버튼을 클릭합니다. 이 작업을 수행하면 클라이언트 매치메이킹 서비스가 FlexMatch에 매치메이킹 요청을 보냅니다. 요청은 요청을 이행할 때 사용할 FlexMatch 매치메이커를 식별합니다. 요청에는 스킬 레벨, 플레이 선호도 또는 지리적 지연 시간 데이터 등 사용자 지정 매치메이커에 필요한 플레이어 정보도 포함됩니다. 한 명 또는 여러 명의 플레이어에 대해 매치메이킹을 요청할 수 있습니다.

매치메이킹 풀에 요청 추가

FlexMatch는 매치메이킹 요청을 받으면 매치메이킹 티켓을 생성하고 매치메이커의 티켓 풀에 추가합니다. 티켓은 매칭되거나 최대 시간 제한에 도달할 때까지 풀에 남아 있습니다. 클라이언트 매치메이킹 서비스는 티켓 상태 변경을 포함한 매치메이킹 이벤트에 대한 알림을 정기적으로 받습니다.

매치 구축

FlexMatch 매치메이커는 풀의 모든 티켓에 대해 다음 프로세스를 지속적으로 실행합니다.

1. 매치메이커는 티켓 연령별로 풀을 정렬한 다음 가장 오래된 티켓부터 시작하여 잠재적 매치를 구축하기 시작합니다.
2. 매치메이커는 잠재적 매치에 두 번째 티켓을 추가하고 사용자 지정 매치메이킹 규칙과 비교하여 결과를 평가합니다. 잠재적 매치가 평가를 통과하면 티켓의 플레이어가 팀에 배정됩니다.
3. 매치메이커는 다음 티켓을 순서대로 추가하고 평가 프로세스를 반복합니다. 플레이어 슬롯이 모두 채워지면 매치가 준비된 것입니다.

라지 매치(41~200명)의 매치메이킹은 합리적인 기간 내에 매치를 진행할 수 있도록 위에서 설명한 대로 수정한 프로세스 버전을 사용합니다. 각 티켓을 개별적으로 평가하는 대신에, 매치메이커는 미리 정렬된 티켓 풀을 잠재적 매치로 나눈 다음 지정한 플레이어 특성에 따라 각 매치의 균형을 맞춥니다. 예를 들어, 매치메이커는 지연 시간이 짧은 유사한 위치를 기준으로 티켓을 미리 정렬한 다음 경기 후 밸런싱을 사용하여 플레이어 스킬별로 팀이 균등하게 매칭되도록 할 수 있습니다.

매치메이킹 결과 보고

적절한 매치가 발견되면 매칭된 모든 티켓이 업데이트되고 매칭된 각 티켓에 대해 성공적인 매치메이킹 이벤트가 생성됩니다.

- 독립형 서비스로써의 FlexMatch: 성공적인 매치메이킹 이벤트에서 게임에 매치 결과가 전송됩니다. 결과 데이터에는 매칭된 모든 플레이어 목록과 팀 배정이 포함됩니다. 매치 요청에 플레이어 지연 시간 정보가 포함되어 있는 경우, 결과는 매치를 위한 최적의 지리적 위치도 추천합니다.
- Amazon GameLift 호스팅 솔루션을 사용한 FlexMatch: 매치 결과는 게임 세션 배치를 위해 Amazon GameLift 대기열에 자동으로 전달됩니다. 매치메이커는 게임 세션 배치에 사용할 대기열을 결정합니다.

매치를 위한 게임 세션 시작

제안된 매치가 성공적으로 구성되면 새 게임 세션이 시작됩니다. 매치를 위한 게임 세션을 설정할 때 게임 서버가 플레이어 ID 및 팀 배정을 포함한 매치메이킹 결과 데이터를 사용할 수 있어야 합니다.

- 독립형 서비스로써의 FlexMatch: 사용자 지정 매치 배치 서비스는 성공적인 매치메이킹 이벤트에서 매치 결과 데이터를 가져오고, 기존 게임 세션 배치 시스템에 연결하여 매치에 사용할 수 있는 호스팅 리소스를 찾습니다. 호스팅 리소스가 검색되면 매치 배치 서비스가 기존 호스팅 시스템과 협력하여 새 게임 세션을 시작하고 연결 정보를 획득합니다.
- Amazon GameLift 호스팅 솔루션을 사용한 FlexMatch: 게임 세션 대기열은 매치에 가장 적합한 게임 서버를 찾습니다. 대기열 구성 방식에 따라 비용이 가장 저렴한 비용의 리소스와 플레이어가 짧은 지연 시간을 경험할 수 있는 위치(플레이어 지연 시간 데이터가 제공되는 경우)로 게임 세션을 배치하도록 합니다. 게임 세션이 성공적으로 배치되면 Amazon GameLift 서비스는 게임

서버에 새 게임 세션을 시작하라는 메시지를 표시하고 매치메이킹 결과 및 기타 선택적 게임 데이터를 전달합니다.

플레이어를 매치에 연결

게임 세션이 시작된 후 플레이어는 세션에 접속하여 팀 배정을 신청하고 게임플레이를 시작합니다.

- 독립형 서비스로써의 FlexMatch: 게임은 기존 게임 세션 관리 시스템을 사용하여 플레이어에게 연결 정보를 다시 제공합니다.
- Amazon GameLift 호스팅 솔루션을 사용한 FlexMatch: 게임 세션을 성공적으로 배치하면 FlexMatch는 게임 세션 연결 정보와 플레이어 세션 ID를 사용하여 모든 매칭된 티켓을 업데이트합니다.

FlexMatch 지원됨 AWS 리전

Amazon GameLift 호스팅 FlexMatch 솔루션과 함께 사용하는 경우 게임을 호스팅하는 모든 위치에서 일치하는 게임 세션을 호스팅할 수 있습니다. Amazon GameLift 호스팅의 전체 목록 AWS 리전 및 위치를 참조하십시오.

모든 FlexMatch 사용자의 경우 다음과 같이 지원되는 방식으로 매치메이킹 구성 및 규칙 세트를 비롯한 FlexMatch 리소스를 호스팅할 수 있습니다. AWS 리전 매치메이커를 위한 위치 선택 섹션을 참조하십시오.

AWS 리전 이름	리전 코드
미국 동부(버지니아 북부)	us-east-1
미국 서부(오레곤)	us-west-2
아시아 태평양(서울)	ap-노스테스트-2
아시아 태평양(시드니)	ap-southeast-2
아시아 태평양(도쿄)	ap-northeast-1
유럽(프랑크푸르트)	eu-central-1
유럽(아일랜드)	eu-west-1
중국 (베이징 및 닝샤 지역)	

FlexMatch 설정

Amazon GameLift FlexMatch는 AWS 서비스이며 이 서비스를 사용하려면 AWS 계정이 있어야 합니다. AWS 계정 생성은 무료입니다. AWS 계정으로 수행할 수 있는 작업에 대한 정보는 AWS 시작하기를 참조하세요.

FlexMatch에 다른 Amazon GameLift 솔루션을 함께 사용하는 경우 다음 주제를 참조하세요.

- Amazon GameLift 호스팅 및 Realtime 서버에 대한 액세스 설정
- Amazon GameLift FleetIQ를 사용하여 Amazon EC2에서 호스팅하기 위한 액세스 설정

Amazon GameLift를 위한 계정을 설정하려면

1. 계정을 만듭니다. Amazon Web Services를 열고 콘솔에 로그인을 선택합니다. 메시지를 따라 새 계정을 만들거나 기존 계정으로 로그인합니다.
2. 관리 사용자 그룹을 설정합니다. AWS Identity and Access Management(IAM) 서비스 콘솔을 열고 단계에 따라 사용자 또는 사용자 그룹을 만들거나 업데이트합니다. IAM은 AWS 서비스와 리소스에 대한 액세스를 관리합니다. Amazon GameLift 콘솔을 사용하거나 Amazon GameLift API를 호출하여 FlexMatch 리소스에 액세스하는 모든 사용자에게 명시적인 액세스 권한을 부여해야 합니다. 콘솔(또는 AWS CLI나 기타 도구)을 사용하여 사용자 그룹을 설정하는 방법에 대한 자세한 지침은 IAM 사용자 생성을 참조하세요.
3. 사용자 또는 그룹에 권한 정책을 연결합니다. IAM 정책을 사용자 또는 사용자 그룹에 연결하여 AWS 서비스 및 리소스에 대한 액세스를 관리합니다. 권한 정책은 사용자가 액세스해야 하는 일련의 AWS 서비스와 작업을 지정합니다.

   Amazon GameLift의 경우 사용자 지정 권한 정책을 생성하여 각 사용자 또는 사용자 그룹에 연결해야 합니다. 정책은 JSON 문서입니다. 아래 예를 사용하여 정책을 생성합니다.

다음 예는 모든 Amazon GameLift 리소스 및 작업에 대한 관리자 권한이 포함된 인라인 권한 정책을 보여줍니다. FlexMatch 관련 항목만 지정하여 액세스를 제한하도록 선택할 수 있습니다.

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

```
    "Resource": "*"
  }
}
```

FlexMatch 시작하기

이 섹션의 리소스를 사용하여 FlexMatch로 매치메이킹 시스템 구축을 시작하는 방법을 시작할 수 있습니다.

주제

독립형 매치메이킹을 위한 Amazon GameLift FlexMatch 통합

이 주제에서는 FlexMatch를 독립형 매치메이킹 서비스로 구현하기 위한 전체 통합 프로세스를 간략하게 설명합니다. P2P, 사용자 지정 구성 온프레미스 하드웨어 또는 기타 클라우드 컴퓨팅 프리미티브를 사용하여 멀티플레이어 게임을 호스팅하는 경우 이 프로세스를 사용합니다. 이 프로세스는 Amazon EC2에서 호스팅되는 게임을 위한 호스팅 최적화 솔루션인 Amazon GameLift FleetIQ에서도 사용할 수 있습니다. Amazon GameLift 관리형 호스팅(Realtime 서버 포함)을 사용하여 게임을 호스팅하는 경우 Amazon GameLift 호스팅을 사용한 FlexMatch 통합 섹션을 참조하세요.

통합을 시작하기 전에 AWS 계정이 있어야 하며 Amazon GameLift 서비스에 대한 액세스 권한을 설정해야 합니다. 자세한 내용은 FlexMatch 설정 섹션을 참조하세요. Amazon GameLift FlexMatch 매치메이커 및 규칙 세트의 생성 및 관리와 관련된 모든 필수 작업은 Amazon GameLift 콘솔을 사용하여 수행할 수 있습니다.

1. FlexMatch 매치메이킹 규칙 세트를 생성합니다. 사용자 지정 규칙 세트는 매치를 구성하는 방법에 대한 완전한 지침을 제공합니다. 여기에서 각 팀의 구조와 규모를 정의합니다. 또한 매치가 유효하기 위해 충족해야 하는 일련의 요구 사항을 제공하며, FlexMatch는 이를 사용하여 매치에 플레이어를 포함하거나 제외합니다. 이러한 요구 사항은 개별 플레이어에게 적용될 수 있습니다. 또한 규칙 세트에서 FlexMatch 알고리즘을 사용자 정의할 수 있습니다(예: 최대 200명의 플레이어가 참여하는 라지 매치를 구축할 수 있음). 다음 주제를 참조합니다.

   - FlexMatch 규칙 세트 설계
   - FlexMatch 규칙 세트 예제
2. 매치메이킹 이벤트 알림을 설정합니다. 알림을 사용하여 보류 중인 매치 요청 상태를 포함하여 FlexMatch 매치메이킹 활동을 추적할 수 있습니다. 이는 제안된 매치의 결과를 전달하는 데 사용되는 메커니즘입니다. 매치메이킹 요청은 비동기식이기 때문에 요청 상태를 추적할 방법이 필요합니다. 알림 사용은 선호 옵션입니다. 다음 주제를 참조합니다.

- FlexMatch 이벤트 알림 설정
- FlexMatch 매치메이킹 이벤트

3. FlexMatch 매치메이킹 구성을 설정합니다. 매치메이커라고도 하는 이 구성 요소는 매치메이킹 요청을 받아 처리합니다. 규칙 세트, 알림 대상, 최대 대기 시간을 지정하여 매치메이커를 구성합니다. 또한 옵션 기능을 활성화할 수 있습니다. 다음 주제를 참조합니다.

   - FlexMatch 매치메이커 디자인
   - 매치메이킹 구성 생성

4. 클라이언트 매치메이킹 서비스를 구축합니다. 매치메이킹 요청을 빌드하고 FlexMatch에 전송하는 기능을 갖춘 게임 클라이언트 서비스를 만들거나 확장합니다. 매치메이킹 요청을 생성하려면 이 구성 요소에 매치메이킹 규칙 세트에 필요한 플레이어 데이터와 선택적으로 지역별 지연 시간 정보를 가져오는 메커니즘이 있어야 합니다. 또한 각 요청에 대해 고유한 티켓 ID를 만들고 배정하는 메서드도 있어야 합니다. 플레이어가 제안된 매치에 참여하도록 요구하는 플레이어 수락 워크플로를 구축하도록 선택할 수도 있습니다. 또한 이 서비스는 매치메이킹 이벤트를 모니터링하여 매치 결과를 얻고 성공적인 매치를 위해 게임 세션 배치를 시작해야 합니다. 이 주제를 참조하세요.

   - FlexMatch를 게임 클라이언트에 추가

5. 매치 배치 서비스를 구축합니다. 기존 게임 호스팅 시스템과 연동되는 메커니즘을 만들어 사용 가능한 호스팅 리소스를 찾고 성공적인 매치를 위해 새 게임 세션을 시작합니다. 이 구성 요소를 통해 매치 결과 정보를 사용하여 사용 가능한 게임 서버를 확보하고 해당 매치를 위한 새 게임 세션을 시작할 수 있어야 합니다. 매치메이킹을 사용하여 이미 실행 중인 매칭 게임 세션의 빈 슬롯을 채우는 매치 채우기 요청을 만드는 워크플로를 구현할 수도 있습니다.

Amazon GameLift 호스팅을 사용한 FlexMatch 통합

FlexMatch는 사용자 지정 게임 서버 및 Realtime 서버에 대한 관리형 Amazon GameLift 호스팅과 함께 사용할 수 있습니다. 게임에 FlexMatch 매치메이킹을 추가하려면 다음 작업을 완료하십시오.

- 매치메이커를 설정합니다. 매치메이커는 플레이어에게 매치메이킹 요청을 수신하여 이를 처리합니다. 이 규칙은 정의된 규칙 세트를 기반으로 플레이어를 그룹화하며, 성공적인 매치마다 새로운 게임 세션과 플레이어 세션을 생성합니다. 매치메이커를 설정하려면 다음 단계를 따릅니다.
  - 규칙 세트를 생성합니다. 규칙 세트는 매치메이커에게 유효한 매치 생성 방법을 알려 줍니다. 팀 구성을 지정하며 매치 포함 여부에 대해 플레이어를 평가하는 방법을 지정합니다. 다음 주제를 참조합니다.

    - FlexMatch 규칙 세트 설계
    - FlexMatch 규칙 세트 예제
  - 게임 세션 대기열을 생성합니다. 대기열은 각 매치에 최선인 리전을 찾아 해당 리전에서 새 게임 세션을 생성합니다. 매치메이킹에 기존 대기열을 사용하거나 새 대기열을 생성합니다. 이 주제를 참조하세요.
    - 대기열 생성
  - 알림을 설정합니다(선택 사항). 매치메이킹 요청은 비동기식이기 때문에 요청 상태를 추적할 방법이 필요합니다. 알림은 기본 옵션입니다. 이 주제를 참조하세요.
    - FlexMatch 이벤트 알림 설정
  - 매치메이커를 구성합니다. 규칙 세트, 대기열 및 알림 대상이 있으면 매치메이커 구성을 생성합니다. 다음 주제를 참조합니다.
    - FlexMatch 매치메이커 디자인
    - 매치메이킹 구성 생성
- FlexMatch를 게임 클라이언트 서비스와 통합합니다. 게임 클라이언트 서비스에 매치메이킹과 함께 새 게임 세션을 시작하는 기능을 추가합니다. 매치메이킹에 대한 요청은 사용할 매치메이커를 지정하고 해당 매치에 필요한 플레이어 데이터를 제공합니다. 이 주제를 참조하세요.
  - FlexMatch를 게임 클라이언트에 추가
- FlexMatch를 게임 클라이언트 서비스와 통합합니다. 게임 서버에 매치메이킹을 통해 생성된 게임 세션을 시작하는 기능을 추가합니다. 이 게임 세션 유형에 대한 요청에는 플레이어 및 팀 배정을 포함한 매치 관련 정보가 포함됩니다. 게임 서버는 매치를 위해 게임 세션을 작성할 때 이 정보를 액세스하고 사용해야 합니다. 이 주제를 참조하세요.
  - FlexMatch를 Amazon GameLift 호스팅 게임 서버에 추가
- FlexMatch 채우기를 설정합니다(선택 사항). 기존 게임에서 열린 플레이어 슬롯을 채우기 위해 추가 플레이어 매치를 요청합니다. Amazon GameLift에서 채우기 요청을 관리하도록 자동 채우기를 설정할 수 있습니다. 또는 게임 클라이언트 서비스 또는 게임 서버에 매치 채우기 요청을 시작하는 기능을 추가하여 수동으로 채우기를 관리할 수 있습니다. 이 주제를 참조하세요.
  - FlexMatch로 기존 게임 채우기

> **Note**
>
> FlexMatch 채우기는 현재 Realtime 서버를 사용한 게임에서 사용할 수 없습니다.

Amazon GameLift FlexMatch 매치메이커 구축

FlexMatch 매치메이커 프로세스는 게임 매치를 구축하는 작업을 수행합니다. 접수된 매치메이킹 요청 풀을 관리하고, 매치를 위한 팀을 구성하며, 가능한 최고의 플레이어 그룹을 찾을 수 있도록 플레이어를 처리 및 선택하고, 매치를 위한 게임 세션을 배치하고 시작하는 프로세스를 시작합니다. 이 주제에서는 매치메이커의 주요 측면과 게임에 사용자 지정된 매치메이커를 구성하는 방법을 설명합니다.

FlexMatch 매치메이커가 수신한 매치메이킹 요청을 처리하는 방법에 대한 자세한 설명은 FlexMatch 매치메이킹 프로세스 섹션을 참조하세요.

주제

FlexMatch 매치메이커 디자인

이 주제에서는 게임에 맞는 매치메이커를 설계하는 방법에 대한 지침을 제공합니다.

기본 매치메이커 구성

매치메이커는 최소한 다음 요소를 필요로 합니다.

- 규칙 세트는 매치를 위한 팀의 규모 및 범위를 결정하고 매치에 참여할 플레이어를 평가할 때 적용할 규칙 세트를 정의합니다. 각 매치메이커는 한 가지 규칙 세트를 사용하도록 구성합니다. FlexMatch 규칙 세트 설계 및 FlexMatch 규칙 세트 예제를 참조합니다.
- 알림 대상은 모든 매치메이킹 이벤트 알림을 받습니다. Amazon Simple Notification Service(SNS) 주제를 설정한 다음 매치메이커에 주제 ID를 추가해야 합니다. 알림 설정에 대한 자세한 정보는 FlexMatch 이벤트 알림 설정 섹션을 참조하세요.
- 요청 타임아웃은 매치메이킹 요청이 요청 풀에 잔류하여 잠재적 매치의 평가 대상이 되는 기간을 결정합니다. 요청 시간이 초과되면 매치가 실패한 것이며 풀에서 제거됩니다.
- Amazon GameLift 관리형 FlexMatch 호스팅과 함께 사용하는 경우 게임 세션 대기열은 매치를 위한 게임 세션을 호스팅하는 데 사용할 수 있는 최상의 리소스를 찾아 새 게임 세션을 시작합니다. 각 대기열은 게임 세션을 배치할 수 있는 위치를 결정하는 위치 및 리소스 유형 (스팟 또는 온디맨드 인스

턴스 포함) 목록으로 구성됩니다. 대기열에 대한 자세한 내용은 다중 위치 대기열 사용을 참조하세요.

매치메이커를 위한 위치 선택

매치메이킹 활동을 어디에서 진행할지 결정하고 해당 위치에 매치메이킹 구성과 규칙 세트를 생성하십시오. GameLift Amazon은 게임의 경기 요청에 대한 티켓 풀을 관리하여 실행 가능한 매치를 분류하고 평가합니다. 매치가 완료되면 Amazon은 게임 세션 배치를 위한 경기 세부 정보를 GameLift 전송합니다. 호스팅 솔루션이 지원하는 모든 위치에서 매칭된 게임 세션을 실행할 수 있습니다.

FlexMatch 리소스를 생성할 수 있는 위치는 FlexMatch 지원됨 AWS 리전 참조하십시오.

AWS 리전매치메이커에 사용할 장소를 선택할 때는 위치가 경기력에 미치는 영향과 플레이어의 경기 경험을 최적화할 수 있는 방법을 고려하세요. 다음 모범 사례를 따르는 것이 좋습니다.

- 플레이어들과 가까운 위치에 매치메이커를 배치하고 매치메이킹 요청을 보내는 고객 서비스를 이용하세요. FlexMatch 이 방식을 사용하면 매치메이킹 요청 워크플로에 미치는 지연 시간이 줄어들어 효율성이 향상됩니다.
- 게임이 전 세계 시청자에게 도달한다면 여러 위치에 매치메이커를 만들어 플레이어와 가장 가까운 매치메이커로 매치 요청을 라우팅하는 것을 고려해 보세요. 이렇게 하면 효율성이 향상될 뿐만 아니라 지리적으로 서로 가까운 플레이어들로 티켓 풀이 형성되어, e지연 시간 요구 사항에 따라 플레이어들을 매칭할 수 있는 매치메이커의 능력이 향상됩니다.
- Amazon GameLift 관리형 FlexMatch 호스팅과 함께 사용하는 경우 매치메이커와 매치메이커가 사용하는 게임 세션 대기열을 같은 위치에 두십시오. 이렇게 하면 매치메이커와 대기열 간의 통신 지연을 최소화하는 데 도움이 됩니다.

선택적 요소 추가

이러한 최소 요건에 더해, 다음 추가 옵션으로 매치메이커를 구성할 수 있습니다. Amazon GameLift 호스팅 FlexMatch 솔루션과 함께 사용하는 경우 많은 기능이 내장되어 있습니다. 독립형 매치메이킹 FlexMatch 서비스로 사용하는 경우 이러한 기능을 시스템에 구축하는 것이 좋습니다.

플레이어 수락

매치에 선택된 모든 플레이어의 참여 수락을 요청하도록 매치메이커를 구성할 수 있습니다. 시스템에서 수락을 요구하는 경우, 모든 플레이어에게는 제안된 매치에 대해 수락하거나 거부할 수 있는 선택권이 주어져야 합니다. 매치는 제안된 매치에 속한 모든 플레이어로부터 수락을 접수해야만 성사됩니다.

임의의 플레이어가 매치를 거부하거나 수락하지 못한 경우 제안된 매치는 폐기되며 티켓은 다음과 같이 처리됩니다. 티켓에 있는 모든 플레이어가 매치를 수락한 티켓은 계속 처리될 수 있도록 매치메이킹 풀로 반환됩니다. 한 명 이상의 플레이어가 매치를 거부하거나 응답하지 않은 티켓은 실패 상태가 되며 더 이상 처리되지 않습니다. 플레이어 수락에는 시간 제한이 수반되며, 모든 플레이어가 제한 시간 내에 제안된 매치를 수락해야 매치가 계속됩니다.

채우기 모드

FlexMatch backfill을 사용하면 게임 세션 내내 잘 어울리는 신규 플레이어로 게임 세션을 가득 채울 수 있습니다. 백필 요청을 처리할 때는 원래 플레이어를 매칭할 때 사용한 것과 동일한 매치메이커를 FlexMatch 사용합니다. 새 매치 티켓의 우선 순위를 지정하여 채우기 티켓을 라인의 맨 앞이나 끝에 배치하는 방식으로 사용자 지정할 수 있습니다. 즉, 새 플레이어가 매치메이킹 풀에 들어오면 새로 구성된 게임에 배정될 확률보다 기존 게임에 배정될 확률이 더 높거나 낮아집니다.

수동 백필은 게임에서 관리형 Amazon GameLift 호스팅을 사용하든 다른 호스팅 FlexMatch 솔루션과 함께 사용하든 상관없이 사용할 수 있습니다. 수동 채우기를 사용하면 채우기 요청을 트리거할 시기를 유연하게 결정할 수 있습니다. 예를 들어 게임의 특정 단계 동안 또는 특정 조건이 존재할 때에 새 플레이어를 추가하려고 할 수 있습니다.

자동 채우기는 관리형 Amazon GameLift 호스팅을 사용하는 게임에만 사용할 수 있습니다. 이 기능을 활성화하면 열린 플레이어 슬롯으로 게임 세션이 시작되면 Amazon에서 자동으로 해당 슬롯에 대한 백필 요청을 GameLift 생성하기 시작합니다. 이 기능을 사용하면 최소 수의 플레이어로 새 게임을 시작한 다음 새 플레이어가 매치메이킹 풀에 들어오면 신속하게 채워지도록 매치메이킹을 설정할 수 있습니다. 게임 세션 수명 전반에 걸쳐 언제든지 자동 채우기를 끌 수 있습니다.

게임 속성

Amazon GameLift 관리형 FlexMatch 호스팅을 사용하는 게임의 경우 새 게임 세션이 요청될 때마다 게임 서버에 전달할 추가 정보를 제공할 수 있습니다. 이는 생성 중인 매치 유형에 맞게 게임 세션을 시작하는 데 필요한 게임 모드 구성을 전달하는 데 유용한 방법이 될 수 있습니다. 매치메이커가 생성한 매치의 모든 게임 세션에는 동일한 게임 속성 세트를 수신합니다. 다양한 매치메이킹 구성을 생성하여 게임 속성 정보를 변경할 수 있습니다.

예약된 플레이어 슬롯

각 매치에 있는 특정 플레이어 슬롯이 예약되었다가 나중에 채워지도록 지정할 수 있습니다. 이는 매치메이킹 구성의 "추가 플레이어 카운트" 속성을 구성함으로써 수행됩니다.

사용자 지정 이벤트 데이터

이 속성을 사용하여 매치메이커에 대한 모든 매치메이킹 관련 이벤트에 일단의 사용자 지정 정보를 포함시킵니다. 이 기능은 매치메이커의 성과 추적을 포함하여 게임에 대한 특정한 고유 활동을 추적하는 데 유용할 수 있습니다.

FlexMatch 규칙 세트 설계

모든 FlexMatch 매치메이커에는 규칙 세트가 있어야 합니다. 규칙 세트는 매치의 두 가지 핵심 요소인 게임의 팀 구조 및 크기와 최상의 매치를 위해 플레이어를 그룹화하는 방법을 결정합니다.

예를 들어, 규칙 세트에서 다음과 같이 매치를 기술할 수 있을 것입니다. 플레이어가 각각 다섯 명이고 한 팀은 수비이고 다른 한 팀은 공격인 두 팀으로 매치를 생성합니다. 한 팀에 초보 플레이어와 경험이 많은 플레이어가 공존할 수 있지만 두 팀의 평균 스킬 포인트가 서로 10포인트 이내여야 합니다. 30초 이후에 매치가 생성되지 않은 경우 스킬 요구 사항을 점진적으로 완화합니다.

이 단원의 주제에서는 매치메이킹 규칙 세트를 설계 및 빌드하는 방법을 설명합니다. 규칙 세트를 생성할 때 Amazon GameLift 콘솔 또는 AWS CLI를 사용할 수 있습니다.

주제

FlexMatch 규칙 세트 설계

이 주제에서는 규칙 세트의 기본 구조 및 최대 40명의 플레이어로 이루어진 소규모 매치에서 규칙 세트를 구축하는 방법에 대해 다룹니다. 매치메이킹 규칙 세트가 수행하는 작업은 두 가지입니다. 하나는 매치의 팀 구조 및 크기를 레이아웃하는 것이고, 다른 하나는 플레이어를 선택하여 최상의 매치를 구성하는 방법을 매치메이커에게 알리는 것입니다.

하지만 매치메이킹 규칙 세트를 사용하면 더 많은 작업을 수행할 수 있습니다. 예를 들어, 다음을 수행할 수 있습니다.

- 게임에 맞게 매치메이킹 알고리즘을 최적화합니다.
- 최소 플레이어 지연 시간 요구 사항을 설정하여 게임 플레이 품질을 보호합니다.

- 시간이 지남에 따라 팀 요구 사항과 매치 규칙을 점진적으로 완화하여 모든 활성 플레이어가 원할 때 적절한 매치를 찾을 수 있습니다.
- 그룹 집계를 사용하여 그룹 매치메이킹 요청에 대한 처리를 정의합니다.
- 40명 이상의 플레이어로 구성된 라지 매치를 처리합니다. 라지 매치 구축에 대한 자세한 내용은 FlexMatch 라지 매치 규칙 세트 설계 섹션을 참조하세요.

매치메이킹 규칙 세트를 만들 때는 다음과 같은 선택 및 필수 작업을 고려합니다.

- 규칙 세트 설명(필수)
- 매치 알고리즘 사용자 정의
- 플레이어 속성 선언
- 매치 팀 정의
- 플레이어 매칭에 대한 규칙 설정
- 시간이 지남에 따라 요구 사항이 완화되도록 허용

Amazon GameLift 콘솔 또는 `CreateMatchmakingRuleSet` 작업을 사용하여 규칙 세트를 구축할 수 있습니다.

규칙 세트 설명(필수)

규칙 세트에 대한 세부 정보를 제공합니다.

- 이름 (선택 사항) - 사용자가 직접 사용할 수 있는 설명 레이블입니다. 이 값은 Amazon GameLift로 규칙 세트를 생성할 때 지정하는 규칙 세트 이름과 관련이 없습니다.
- ruleLanguageVersion - FlexMatch 규칙을 생성하는 데 사용되는 속성 표현식 언어의 버전입니다. 값은 `1.0`여야 합니다.

매치 알고리즘 사용자 정의

FlexMatch는 대부분의 게임에 대한 기본 알고리즘을 최적화하여 플레이어가 최소한의 대기 시간으로 적절한 매치에 참여할 수 있도록 합니다. 알고리즘을 사용자 정의하고 게임에 맞게 매치메이킹을 조정할 수 있습니다.

다음은 기본 FlexMatch 매치메이킹 알고리즘입니다.

1. FlexMatch는 모든 오픈 매치메이킹 티켓과 채우기 티켓을 티켓 풀에 배치합니다.

2. FlexMatch는 풀의 티켓을 하나 이상의 배치로 무작위로 그룹화합니다. 티켓 풀이 더 커지면 FlexMatch는 최적의 배치 크기를 유지하기 위해 추가 배치를 구성합니다.
3. FlexMatch는 각 배치 내에서 연령별로 티켓을 정렬합니다.
4. FlexMatch는 각 배치에서 가장 오래된 티켓을 기반으로 매치를 구성합니다.

매치 알고리즘을 사용자 지정하려면 규칙 세트 스키마에 `algorithm` 구성 요소를 추가합니다. 전체 참조 문서에 대해서는 FlexMatch 규칙 세트 스키마 섹션을 참조하세요.

다음과 같은 선택적 사용자 지정을 사용하여 매치메이킹 프로세스의 여러 단계에 영향을 미칠 수 있습니다.

- 사전 배치 정렬 추가
- batchDistance 속성을 기반으로 배치 구성
- 채우기 티켓의 우선 순위 지정
- 확장이 있는 이전 티켓 선호

사전 배치 정렬 추가

배치를 구성하기 전에 티켓 풀을 정렬할 수 있습니다. 이러한 유형의 사용자 지정은 티켓 풀이 큰 게임에서 가장 효과적입니다. 사전 배치 정렬은 매치메이킹 프로세스를 가속화하고 정의된 특성에 대한 플레이어 균일성을 높이는 데 도움이 될 수 있습니다.

`batchingPreference` 알고리즘 속성을 사용하여 사전 배치 정렬 메서드를 정의합니다. 기본 설정은 `random`입니다.

사전 배치 정렬을 사용자 지정하는 옵션은 다음과 같습니다.

- 플레이어 속성별로 정렬합니다. 플레이어 속성 목록을 제공하여 티켓 풀을 사전 정렬합니다.

  플레이어 속성별로 정렬하려면 `batchingPreference`를 `sorted`로 설정하고 `sortByAttributes`에서 플레이어 속성 목록을 정의합니다. 속성을 사용하려면 먼저 규칙 세트의 `playerAttributes` 구성 요소에서 속성을 선언해야 합니다.

  다음 예제에서 FlexMatch는 플레이어가 선호하는 게임 맵과 플레이어 기술을 기준으로 티켓 풀을 정렬합니다. 결과 배치에는 동일한 맵을 사용하려는 비슷한 스킬을 갖춘 플레이어가 포함될 가능성이 더 큽니다.

```
"algorithm": {
```

```
    "batchingPreference": "sorted",
    "sortByAttributes": ["map", "player_skill"],
    "strategy": "exhaustiveSearch"
},
```

- 지연 시간별로 정렬합니다. 지연 시간이 가장 낮은 매치를 생성하거나 허용 가능한 지연 시간으로 빠르게 매치를 생성합니다. 이 사용자 지정은 40명 이상의 플레이어로 구성된 라지 매치를 구성하는 규칙 세트에 유용합니다.

  알고리즘 속성을 `strategy`에서 `balanced`로 설정합니다. 밸런스 전략은 사용 가능한 규칙 문 유형을 제한합니다. 자세한 내용은 FlexMatch 라지 매치 규칙 세트 설계 섹션을 참조하세요.

  FlexMatch는 다음 방법 중 하나로 플레이어가 보고한 지연 시간 데이터를 기반으로 티켓을 정렬합니다.

  - 가장 짧은 지연 시간 위치. 티켓 풀은 플레이어가 가장 짧은 지연 시간 값을 보고한 위치를 기준으로 사전 정렬됩니다. 그런 다음 FlexMatch는 동일한 위치에서 지연 시간이 짧은 티켓을 일괄 처리하여 더 나은 게임 플레이 경험을 제공합니다. 또한 각 배치의 티켓 수가 줄어들어 매치메이킹이 더 오래 걸릴 수 있습니다. 이 사용자 지정을 사용하려면 다음 예와 같이 `batchingPreference`를 `fastestRegion`으로 설정합니다.

    ```
    "algorithm": {
        "batchingPreference": "fastestRegion",
        "strategy": "balanced"
    },
    ```

  - 허용 가능한 지연 시간과 빠르게 일치합니다. 티켓 풀은 플레이어가 가장 짧은 지연 시간 값을 보고한 위치를 기준으로 사전 정렬됩니다. 이렇게 하면 더 많은 티켓이 포함된 배치 수가 줄어듭니다. 각 배치에 더 많은 티켓이 포함되므로 적합한 매치를 더 빨리 찾을 수 있습니다. 이 사용자 지정을 사용하려면 다음 예와 같이 `batchingPreference` 속성을 `largestPopulation`으로 설정합니다.

    ```
    "algorithm": {
        "batchingPreference": "largestPopulation",
        "strategy": "balanced"
    },
    ```

  > **Note**
  >
  > 밸런스 전략의 기본값은 `largestPopulation`입니다.

채우기 티켓의 우선 순위 지정

게임에서 자동 채우기 또는 수동 채우기를 구현하는 경우 요청 유형에 따라 FlexMatch가 매치메이킹 티켓을 처리하는 방법을 사용자 지정할 수 있습니다. 요청 유형은 새 매치 또는 채우기 요청일 수 있습니다. 기본적으로 FlexMatch는 두 유형의 요청을 동일하게 처리합니다.

채우기 우선 순위는 FlexMatch가 티켓을 일괄 처리한 후 처리하는 방식에 영향을 줍니다. 채우기 우선 순위를 지정하려면 철저한 검색 전략을 사용하는 규칙 세트가 필요합니다.

FlexMatch는 여러 채우기 티켓과 함께 매칭하지 않습니다.

채우기 티켓의 우선 순위를 변경하려면 `backfillPriority` 속성을 설정하세요.

- 채우기 티켓을 먼저 매칭합니다. 이 옵션은 새 매치를 만들기 전에 채우기 티켓을 매칭하려고 시도합니다. 즉, 들어오는 플레이어가 기존 게임에 참여할 확률이 더 높습니다.

  게임에서 자동 채우기를 사용하는 경우 이 방법을 사용하는 것이 가장 좋습니다. 자동 채우기는 게임 세션이 짧고 플레이어 교체 시간이 긴 게임에서 주로 사용됩니다. 자동 채우기는 FlexMatch가 빈 슬롯을 채울 더 많은 플레이어를 검색하는 동안 이러한 게임에 최소 실행 가능한 매치를 구성하고 시작할 수 있도록 도와줍니다.

  `backfillPriority`을 `high`로 설정합니다.

  ```
  "algorithm": {
      "backfillPriority": "high",
      "strategy": "exhaustiveSearch"
  },
  ```

- 채우기 티켓을 마지막에 매칭합니다. 이 옵션은 다른 모든 티켓을 평가하기 전까지는 채우기 티켓을 무시합니다. 즉, FlexMatch는 들어오는 플레이어를 새 게임에 매칭할 수 없을 때 기존 게임으로 다시 채웁니다.

  이 옵션은 새 매치를 구성할 플레이어가 충분하지 않은 경우와 같이 플레이어를 게임에 참여시키기 위한 마지막 기회로 채우기를 사용하려는 경우에 유용합니다.

  `backfillPriority`를 `low`로 설정합니다.

  ```
  "algorithm": {
      "backfillPriority": "low",
      "strategy": "exhaustiveSearch"
  },
  ```

확장이 있는 이전 티켓 선호

확장 규칙은 매치를 완료하기 어려운 경우 매치 기준을 완화시킵니다. Amazon GameLift는 부분적으로 완료된 매치의 티켓이 특정 연령에 도달하면 확장 규칙을 적용합니다. 티켓 생성 타임스탬프는 Amazon GameLift가 규칙을 적용하는 시기를 결정합니다. 기본적으로 FlexMatch는 가장 최근에 매칭된 티켓의 타임스탬프를 추적합니다.

FlexMatch가 확장 규칙을 적용하는 시기를 변경하려면 다음과 같이 `expansionAgeSelection` 속성을 설정합니다.

- 최신 티켓을 기준으로 확장합니다. 이 옵션은 잠재적 매치에 추가된 최신 티켓을 기반으로 확장 규칙을 적용합니다. FlexMatch가 새 티켓을 매칭할 때마다 시간 클록이 재설정됩니다. 이 옵션을 사용하면 매치의 품질이 더 높게 나오지만 매칭하는 데 시간이 더 오래 걸리는 경향이 있습니다. 매칭하는 데 시간이 너무 오래 걸리면 매치 요청이 완료되기까지 시간이 초과될 수 있습니다. 기본적으로 `expansionAgeSelection`은 `newest`, `newest`로 설정됩니다.
- 가장 오래된 티켓을 기준으로 확장합니다. 이 옵션은 잠재적 매치에 가장 오래된 티켓을 기반으로 확장 규칙을 적용합니다. 이 옵션을 사용하면 FlexMatch는 확장을 더 빠르게 적용하여 가장 먼저 매칭한 플레이어의 대기 시간을 개선하지만 모든 플레이어의 매치 품질을 낮춥니다. `expansionAgeSelection`를 `oldest`로 설정합니다.

```
"algorithm": {
    "expansionAgeSelection": "oldest",
    "strategy": "exhaustiveSearch"
},
```

플레이어 속성 선언

이 섹션에서는 매치메이킹 요청에 포함할 개별 플레이어 속성을 나열합니다. 규칙 세트에서 플레이어 속성을 선언하는 데에는 두 가지 이유가 있습니다.

- 규칙 세트에 플레이어 속성에 의존하는 규칙이 포함된 경우.
- 매치 요청을 통해 플레이어 속성을 게임 세션에 전달하려는 경우. 예를 들어 각 플레이어가 연결하기 전에 플레이어 캐릭터 선택을 게임 세션에 전달하고 싶을 수 있습니다.

플레이어 속성을 선언할 때 다음 정보를 포함합니다.

- name(필수) - 이 값은 규칙 세트에 대해 고유해야 합니다.

- type(필수) - 이는 속성 값의 데이터 유형입니다. 유효한 데이터 유형은 숫자, 문자열 또는 문자열 맵입니다.
- default(선택 사항) - 매치메이킹 요청이 속성 값을 제공하지 않는 경우 사용할 기본값을 입력합니다. 기본값이 선언되지 않고 요청에 값이 포함되지 않은 경우 FlexMatch는 요청을 처리할 수 없습니다.

매치 팀 정의

매치에 대한 팀의 구조 및 크기를 설명합니다. 각 매치에 팀이 하나 이상 있어야 하며 원하는 수의 팀을 정의할 수 있습니다. 팀의 플레이어 수가 동일할 수 있거나 팀이 비대칭일 수 있습니다. 예를 들어 플레이어가 한 명인 몬스터 팀과 플레이어가 10명인 헌터 팀을 정의할 수 있습니다.

FlexMatch에서는 규칙 세트가 팀 크기를 정의하는 방법에 따라 매치 요청을 스몰 매치 또는 라지 매치로 처리합니다. 플레이어 수가 최대 40명인 잠재적 매치는 스몰 매치이며 플레이어 수가 40을 초과하는 매치는 라지 매치입니다. 규칙 세트의 잠재적 매치 크기를 결정하려면 규칙 세트에 정의된 모든 팀에 대한 maxPlayer 설정을 합산합니다.

- name(필수 항목) - 각 팀에 고유한 이름을 할당합니다. 이 이름은 규칙과 확장, 및 게임 세션의 매치메이킹 데이터에 대한 FlexMatch 참조에서 사용합니다.
- maxPlayers(필수) - 팀에 할당할 수 있는 최대 플레이어 수를 지정합니다.
- minPlayers(필수) - 팀에 할당할 수 있는 최소 플레이어 수를 지정합니다.
- quantity(선택 사항) - 이 정의에 따라 구성할 팀 수를 지정합니다. FlexMatch가 매치를 생성하면 해당 팀에 제공된 이름과 함께 번호가 추가됩니다. 예: `Red-Team1`, `Red-Team2`, `Red-Team3`.

FlexMatch는 최대 플레이어 규모로 팀을 채우려고 하지만 플레이어 수가 적은 팀을 만듭니다. 매치에 참여하는 모든 팀의 크기를 동일하게 하려는 경우 해당 규칙을 생성할 수 있습니다. `EqualTeamSizes` 규칙의 예제는 FlexMatch 규칙 세트 예제 주제를 참조하세요.

플레이어 매칭에 대한 규칙 설정

매치에 대한 수락을 위해 플레이어를 평가하는 규칙 문 세트를 생성합니다. 규칙은 개별 플레이어, 팀 또는 전체 매치에 적용되는 요구 사항을 설정할 수 있습니다. Amazon GameLift에서 매치 요청을 처리할 때 사용 가능한 플레이어 풀에서 가장 오래된 플레이어부터 시작하고 해당 플레이어에 대한 매치를 빌드합니다. FlexMatch 규칙 생성에 대한 자세한 도움말은 FlexMatch 규칙 유형 섹션을 참조하세요.

- name(필수) - 규칙 세트에서 규칙을 고유하게 식별하는 유의미한 이름입니다. 규칙 이름은 이 규칙과 관련된 활동을 추적하는 이벤트 로그 및 측정치에서도 참조됩니다.
- description(선택 항목) - 이 요소는 자유로운 양식의 텍스트 설명을 연결하는 데 사용합니다.

- type(필수) - 이 type 요소는 규칙을 처리할 때 사용하는 연산을 식별합니다. 각 규칙 유형에는 추가 속성 세트가 필요합니다. FlexMatch 규칙 언어의 유효한 규칙 유형 및 속성 목록을 참조하세요.
- 규칙 유형 속성(필수 항목일 수 있음) - 정의되는 규칙 유형에 따라 특정 규칙 속성을 설정해야 할 수도 있습니다. FlexMatch 규칙 언어에서 속성 및 FlexMatch 속성 표현식 언어를 사용하는 방법에 대해 자세히 알아봅니다.

시간이 지남에 따라 요구 사항이 완화되도록 허용

확장을 통해 FlexMatch에서 매치를 찾을 수 없는 경우 시간이 지남에 따라 규칙 기준을 완화할 수 있습니다. 이 기능을 사용하면 완벽한 매칭을 할 수 없을 때 FlexMatch를 최대한 활용할 수 있습니다. 확장을 통해 규칙을 완화함으로써 적절한 매치가 있는 플레이어 풀을 점진적으로 확장합니다.

미완료 매치의 최신 티켓 연령이 확장팩 대기 시간과 일치할 때 확장이 시작됩니다. FlexMatch가 매치에 새 티켓을 추가하면 확장 대기 시간 클록이 재설정될 수 있습니다. 규칙 세트 `algorithm` 섹션에서 확장 시작 방식을 사용자 지정할 수 있습니다.

다음은 매치에 필요한 최소 스킬 레벨을 점진적으로 늘리는 확장의 예입니다. 규칙 세트는 SkillDelta라는 거리 규칙 문을 사용하여 한 매치에 참가하는 모든 플레이어가 서로 5스킬 레벨 이내여야 합니다. 15초 동안 새로운 매치가 이루어지지 않으면 이 확장에서는 10의 스킬 레벨 차이를 찾고 10초 후에는 20의 차이를 찾습니다.

```
"expansions": [{
        "target": "rules[SkillDelta].maxDistance",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 10
        }, {
            "waitTimeSeconds": 25,
            "value": 20
        }]
    }]
```

자동 채우기가 활성화되어 있는 매치메이커의 경우 플레이어 수 요구 사항을 너무 빨리 완화하지 않습니다. 새 게임 세션이 시작되고 자동 채우기를 시작할 때까지 몇 초 정도의 시간이 걸립니다. 보다 나은 접근 방식은 게임에 대한 자동 채우기가 시작될 조짐이 보인 후에만 확장을 시작하는 것입니다. 확장 타이밍은 팀 구성 요소에 따라 다르므로 테스트를 통해 게임에 가장 적합한 확장 전략을 찾습니다.

FlexMatch 라지 매치 규칙 세트 설계

규칙 세트에서 41~200명의 플레이어가 허용되는 매치를 만들 경우 규칙 세트 구성을 약간 조정해야 합니다. 이러한 조정을 통해 매치 알고리즘이 최적화되어 플레이어 대기 시간을 단축하는 동시에 라지 매치를 성공적으로 구축할 수 있습니다. 따라서 라지 매치 규칙 세트는 시간이 많이 걸리는 사용자 지정 규칙을 일반적인 매치메이킹 우선 순위에 최적화된 표준 솔루션으로 대체합니다.

라지 매치를 위해 규칙 세트를 최적화해야 하는지 여부를 결정하는 방법은 다음과 같습니다.

1. 규칙 세트에 정의된 각 팀에 대해 maxPlayer의 값을 얻습니다.
2. 모든 maxPlayer 값을 추가합니다. 합계가 40을 초과하는 경우 라지 매치 규칙 세트가 지정된 것입니다.

라지 매치에 맞게 규칙 세트를 최적화하려면 다음에 설명한 대로 조정합니다. 라지 매치에 대한 규칙 세트 스키마에서 라지 매치 규칙 세트에 대한 스키마 및 예제 7: 라지 매치 생성에서 규칙 세트 예를 참조하세요.

라지 매치를 위한 매치 알고리즘 사용자 지정

아직 없는 경우 알고리즘 구성 요소를 규칙 세트에 추가합니다. 다음 속성을 설정합니다.

- `strategy`(필수) - `strategy` 속성을 “balanced”로 설정합니다. 이 설정은 FlexMatch가 `balancedAttribute` 속성에 정의된 지정된 플레이어 속성을 기반으로 최적의 팀 밸런스를 찾기 위해 매치 후 추가 확인을 수행하도록 트리거합니다. 균형 잡힌 전략을 사용하면 균등하게 매칭되는 팀을 구성하기 위한 사용자 지정 규칙의 필요성을 대체합니다.
- `balancedAttribute`(필수) - 매치에서 팀 밸런스를 맞출 때 사용할 플레이어 속성을 식별합니다. 이 속성은 숫자 데이터 유형(Double 또는 정수)이어야 합니다. 예를 들어, 플레이어 스킬에 균형을 맞추기로 선택한 경우 FlexMatch는 모든 팀의 총 스킬 레벨이 최대한 균등하게 매칭하도록 플레이어를 배정하려고 합니다. 규칙 세트의 플레이어 속성에서 밸런싱 속성을 선언해야 합니다.
- `batchingPreference`(선택 사항) - 플레이어들이 최대한 지연 시간을 최소화할 수 있는 매치를 만드는 데 어느 정도 중점을 둘 것인지 선택합니다. 이 설정은 매치를 구성하기 전에 매치 티켓을 정렬하는 방식에 영향을 줍니다. 옵션에는 다음이 포함됩니다.
  - 가장 큰 모집단. FlexMatch를 사용하면 한 곳 이상의 공통 위치에서 허용 가능한 지연 시간이 있는 풀의 모든 티켓을 사용하여 매치를 진행할 수 있습니다. 따라서 잠재적 티켓 풀이 커지는 경향이 있어 매치를 더 빨리 채우기가 더 쉬워집니다. 플레이어들이 게임에 적절하게 배정될 수 있지만 항상 최적의 상태는 아니며 지연 시간이 있을 수 있습니다. `batchingPreference` 속성이 설정되지 않은 경우, `strategy`가 “balanced”로 설정된 때의 기본 동작입니다.

- 가장 빠른 위치. FlexMatch는 가장 낮은 지연 시간 값을 보고하는 위치를 기준으로 풀의 모든 티켓을 사전 정렬합니다. 따라서 동일한 위치에서 지연 시간이 짧다고 보고한 플레이어로 매치를 구성하는 경향이 있습니다. 동시에 각 매치의 잠재적 티켓 풀이 적기 때문에 매치를 채우는 데 필요한 시간이 늘어날 수 있습니다. 또한 지연 시간에 더 높은 우선 순위가 부여되기 때문에 매치에 참가하는 플레이어는 밸런싱 속성과 관련하여 더 폭넓게 달라질 수 있습니다.

다음 예에서는 다음과 같이 동작하도록 매치 알고리즘을 구성합니다. (1) 티켓 풀을 사전 정렬하여 지연 시간 값이 허용되는 위치별로 티켓을 그룹화하고, (2) 매칭을 위해 정렬된 티켓을 일괄 구성하며, (3) 티켓으로 매치를 만들고 팀의 균형을 조정하여 평균 플레이어 스킬을 균등하게 맞춥니다.

```
"algorithm": {
    "strategy": "balanced",
    "balancedAttribute": "player_skill",
    "batchingPreference": "largestPopulation"
},
```

플레이어 속성 선언

규칙 세트 알고리즘에서 밸런싱 속성으로 사용되는 플레이어 속성을 선언해야 합니다. 매치메이킹 요청의 각 플레이어마다 이 속성을 포함해야 합니다. 플레이어 속성에 기본값을 제공할 수 있지만 속성 밸런싱은 플레이어별 값을 제공할 때 가장 잘 작동합니다.

팀 정의

팀 크기 및 구조를 정의하는 프로세스는 스몰 매치의 경우와 동일하지만 FlexMatch에서 팀을 채우는 방식이 다릅니다. 이는 매치가 부분적으로만 채워졌을 때의 모양과 어떻게 가장 가까울 수 있는지에 영향을 줍니다. 응답에서 최소 팀 크기를 조정할 수도 있습니다.

FlexMatch는 팀에 플레이어를 할당할 때 다음 규칙을 사용합니다. 먼저 아직 해당 최소 플레이어 요구 사항을 충족하지 않은 팀을 찾습니다. 두 번째로, 이러한 팀 중 열려 있는 슬롯이 가장 많은 팀을 찾습니다.

크기가 동일하게 지정된 여러 팀을 정의하는 매치의 경우 채워질 때까지 각 팀에 순차적으로 플레이어가 추가됩니다. 따라서 매치에 참가한 팀의 플레이어 수는 항상 거의 같으며, 매치가 꽉 차지 않은 경우에도 마찬가지입니다. 현재는 라지 매치에서 크기가 동일하게 지정된 팀을 강제 적용할 방법은 없습니다. 비대칭적으로 크기가 지정된 팀이 있는 매치의 경우 이 프로세스는 약간 더 복잡합니다. 이 시나리오에서는 플레이어가 처음에는 열려 있는 슬롯이 가장 많은 가장 큰 팀에 할당됩니다. 열려 있는 슬롯 수가 모든 팀 간에 더 공평하게 분산됨에 따라 플레이어가 더 작은 팀에 슬롯됩니다.

예를 들어, 세 팀으로 구성된 규칙 세트가 있다고 가정해 보겠습니다. Red 팀과 Blue 팀 모두 maxPlayers=10, minPlayers=5로 설정되어 있으며, Green 팀은 maxPlayers =3, minPlayers =2로 설정되어 있습니다. 채우기 순서는 다음과 같습니다.

1. 아직 minPlayers에 도달한 팀이 없습니다. Red 팀과 Blue 팀에 열려 있는 슬롯이 10개 있는데 Green 팀에는 3개가 있습니다. 첫 10명의 플레이어(각각 5명)가 Red 팀과 Blue 팀에 할당됩니다. 이제 두 팀 모두 minPlayers에 도달했습니다.
2. Green 팀은 아직 minPlayers에 도달하지 않았습니다. 다음 2명의 플레이어가 Green 팀에 할당됩니다. 이제 Green 팀이 minPlayers에 도달했습니다.
3. 모든 팀이 minPlayers인 경우, 이제 열린 슬롯 수에 따라 추가 플레이어가 배정됩니다. Red 팀과 Blue 팀에 열려 있는 슬롯이 5개 있는데 Green 팀에는 1개가 있습니다. 다음 8명의 플레이어(각각 4명)가 Red 팀과 Blue 팀에 할당됩니다. 이제 모든 팀에 1개의 빈 슬롯이 있습니다.
4. 나머지 3명의 플레이어 슬롯은 특별한 순서 없이 팀(각 1명씩)에 배정됩니다.

라지 매치의 규칙 설정

라지 매치의 매치메이킹은 주로 밸런싱 전략과 지연 시간 배칭 최적화에 의존합니다. 대부분의 사용자 지정 규칙을 사용할 수 없습니다. 하지만 다음과 같은 규칙 유형을 통합할 수 있습니다.

- 플레이어 지연 시간을 엄격하게 제한하는 규칙. maxLatency 속성과 함께 latency 규칙 유형을 사용합니다. 지연 규칙 참조를 살펴보세요. 다음은 최대 플레이어 지연 시간을 200밀리초로 설정하는 예입니다.

```
"rules": [{
        "name": "player-latency",
        "type": "latency",
        "maxLatency": 200
    }],
```

- 지정된 플레이어 속성의 친밀도를 기반으로 플레이어를 배치하기 위한 규칙입니다. 이는 균등하게 매칭되는 팀을 구성하는 데 초점을 맞추는 라지 매치 알고리즘의 일부로 밸런싱 속성을 정의하는 것과는 다릅니다. 이 규칙은 초보자 또는 전문가 스킬과 같은 지정된 속성 값의 유사성을 기준으로 매치메이킹 티켓을 일괄 처리하며, 이로 인해 플레이어가 지정된 속성에 밀접하게 연계되는 경향이 있습니다. batchDistance 규칙 유형을 사용하여 수치 기반 속성을 식별하고 허용할 가장 넓은 범위를 지정합니다. 배치 거리 규칙 참조를 살펴보세요. 다음은 매치의 플레이어들이 서로 한 스킬 레벨 이내여야 하는 예제입니다.

```
"rules": [{
```

```
        "name": "batch-skill",
        "type": "batchDistance",
        "batchAttribute": "skill",
        "maxDistance": 1
```

라지 매치 요구 사항 완화

스몰 매치의 경우처럼 유효한 매치가 가능하지 않을 때 확장을 사용하여 시간 경과에 따라 매치 요구 사항을 완화할 수 있습니다. 라지 매치의 경우 지연 시간 규칙 또는 팀 플레이어 수를 완화하는 옵션이 있습니다.

라지 매치에 자동 매치 채우기를 사용하는 경우 팀 플레이어 수를 너무 빨리 완화하지 않습니다. FlexMatch는 게임 세션이 시작된 후에만 채우기 요청을 생성하기 시작합니다. 이 요청은 매치가 생성된 후 몇 초 동안 발생하지 않을 수 있습니다. 해당 시간 동안 FlexMatch는 특히 플레이어 수 규칙이 낮게 조정될 때 부분적으로 채워진 여러 새 게임 세션을 생성할 수 있습니다. 따라서 결국 필요한 수보다 더 많은 게임 세션이 생성되고 이러한 세션 간에 플레이어가 너무 촘촘하게 분산됩니다. 최상의 방법은 플레이어 수 확장의 첫 번째 단계에 더 긴 대기 시간(게임 세션이 시작하기에 충분히 긴 시간)을 지정하는 것입니다. 라지 매치의 경우 채우기 요청에 더 높은 우선 순위가 지정되므로 새 게임이 시작되기 전에 새로 들어오는 플레이어는 기존 게임에 배정됩니다. 게임에 이상적인 대시 시간을 찾아보는 연습을 해야 할 수도 있습니다.

다음은 초기 대기 시간이 더 긴 Yellow 팀의 플레이어 수를 점진적으로 낮추는 예입니다. 규칙 세트 확장의 대기 시간은 복합 값이 아닌 절대 값임을 명심하십시오. 따라서 첫 번째 확장이 5초에 발생하고, 두 번째 확장이 5초 이후인 10초에 발생합니다.

```
"expansions": [{
        "target": "teams[Yellow].minPlayers",
        "steps": [{
            "waitTimeSeconds": 5,
            "value": 8
        }, {
            "waitTimeSeconds": 10,
            "value": 5
        }]
    }]
```

매치메이킹 규칙 세트 생성

Amazon 매치메이커용 GameLift FlexMatch 매치메이킹 규칙 세트를 생성하기 전에 규칙 세트 구문을 확인하는 것이 좋습니다. Amazon GameLift 콘솔 또는 AWS Command Line Interface (AWS CLI) 를 사용하여 규칙 세트를 생성한 후에는 변경할 수 없습니다.

AWS 리전에서 보유할 수 있는 최대 규칙 세트 수에 대한 Service Quotas가 있으므로 사용하지 않는 규칙 세트를 삭제하는 것이 좋습니다.

관련 주제

- FlexMatch 규칙 세트 설계
- FlexMatch 규칙 세트 예제
- FlexMatch 규칙 언어

Console

규칙 세트 생성

1. https://console.aws.amazon.com/gamelift/ 에서 아마존 GameLift 콘솔을 엽니다.
2. 규칙 세트를 생성하고자 하는 AWS 리전으로 전환합니다. 규칙 세트는 함께 사용하고자 하는 매치메이킹 구성과 동일한 리전에서 정의합니다.
3. 탐색 창에서 매치메이킹 규칙 세트를 선택합니다 FlexMatch.
4. 매치메이킹 규칙 세트 페이지에서 규칙 세트 생성을 선택합니다.
5. 매치메이킹 규칙 세트 생성 페이지에서 다음을 실행합니다.

    a. 규칙 세트 설정에서 이름에 목록이나 이벤트 및 지표 테이블에서 식별하는 데 사용할 수 있는 고유한 설명 이름을 입력합니다.

    b. 규칙 세트에는 규칙 세트를 JSON으로 입력합니다. 규칙 세트 설계에 대한 자세한 내용은 FlexMatch 규칙 세트 설계 섹션을 참조하세요. FlexMatch 규칙 세트 예제의 예제 규칙 세트 중 하나를 사용할 수도 있습니다.

    c. 검증을 선택하여 규칙 세트 구문이 정확한지 검증합니다. 규칙 세트를 만든 후에는 편집할 수 없으므로 먼저 검증하는 것이 좋습니다.

    d. (선택 사항) 태그에는 AWS 리소스를 관리하고 추적하는 데 도움이 되는 태그를 추가합니다.

6. 생성을 선택하세요. 생성을 성공적으로 마쳤으면 매치메이커가 규칙 세트를 사용할 수 있습니다.

AWS CLI

규칙 세트 생성

커맨드 라인 창을 열고 커맨드를 사용합니다. create-matchmaking-rule-set

이 예제 명령에서는 단일 팀을 설정하는 간단한 매치메이킹 규칙 세트를 생성합니다. 사용할 매치메이킹 구성과 동일한 AWS 리전에 규칙 세트를 생성해야 합니다.

```
aws gamelift create-matchmaking-rule-set \
    --name "SampleRuleSet123" \
    --rule-set-body '{"name": "aliens_vs_cowboys", "ruleLanguageVersion": "1.0",
 "teams": [{"name": "cowboys", "maxPlayers": 8, "minPlayers":  4}]}'
```

생성 요청이 성공하면 Amazon은 지정한 설정이 포함된 MatchmakingRuleSet객체를 GameLift 반환합니다. 이제 매치메이커가 새 규칙 세트를 사용할 수 있습니다.

Console

규칙 세트 삭제

1. https://console.aws.amazon.com/gamelift/ 에서 아마존 GameLift 콘솔을 엽니다.
2. 규칙 세트를 생성한 리전으로 전환합니다.
3. 탐색 창에서 매치메이킹 규칙 세트를 선택합니다 FlexMatch.
4. 매치메이킹 규칙 세트 페이지에서 삭제하려는 규칙 세트를 선택한 다음 삭제를 선택합니다.
5. 규칙 세트 삭제 대화 상자에서 삭제를 선택하여 삭제를 확인합니다.

> **Note**
>
> 매치메이킹 구성이 규칙 세트를 사용하는 경우 Amazon은 오류 메시지를 GameLift 표시합니다 (규칙 세트를 삭제할 수 없음). 이 경우 매치메이킹 구성을 다른 규칙 세트를 사용하도록 변경한 다음 다시 시도합니다. 규칙 세트를 사용 중인 매치메이킹 구성을 확인하려면 규칙 세트 이름을 선택하여 해당 세부 정보 페이지를 봅니다.

AWS CLI

규칙 세트 삭제

명령줄 창을 열고 명령을 사용하여 매치메이킹 규칙 세트를 삭제합니다. delete-matchmaking-rule-set

매치메이킹 구성이 규칙 세트를 사용하는 경우 Amazon은 오류 메시지를 GameLift 반환합니다. 이 경우 매치메이킹 구성을 다른 규칙 세트를 사용하도록 변경한 다음 다시 시도합니다. 규칙 세트를 사용하는 매치메이킹 구성의 목록을 가져오려면 명령을 describe-matchmaking-configurations사용하고 규칙 세트 이름을 지정하십시오.

이 예제 명령에서는 매치메이킹 규칙 세트의 사용 여부를 확인한 후 규칙 세트를 삭제합니다.

```
aws gamelift describe-matchmaking-rule-sets \
    --rule-set-name "SampleRuleSet123" \
    --limit 10

aws gamelift delete-matchmaking-rule-set \
    --name  "SampleRuleSet123"
```

FlexMatch 규칙 세트 예제

FlexMatch 규칙 세트는 다양한 매치메이킹 시나리오를 포함할 수 있습니다. 다음 예제는 FlexMatch 구성 구조 및 속성 표현식 언어를 준수합니다. 이들 규칙 전부를 복사하거나 필요한 구성 요소를 선택합니다.

FlexMatch 규칙 및 규칙 세트 사용에 대한 자세한 내용은 다음 항목을 참조하십시오.

- FlexMatch 규칙 세트 설계
- FlexMatch 규칙 세트 설계
- FlexMatch 규칙 세트 스키마
- FlexMatch 규칙 언어

> **Note**
>
> 복수 플레이어를 포함하는 매치메이킹 티켓을 평가할 때 요청에 포함된 모든 플레이어가 매치 요건을 충족해야 합니다.

예제 1: 플레이어가 동등하게 매칭된 두 팀 생성

이 예제에서는 다음과 같은 지침을 기준으로 플레이어가 동등하게 매치된 두 팀을 설정하는 방법을 설명합니다.

- 두 개의 플레이어 팀을 생성합니다.
  - 각 팀마다 4~8 명의 플레이어를 포함합니다.
  - 최종적으로 팀들의 플레이어 수가 동일해야 합니다.
- 플레이어의 스킬 레벨을 포함합니다(제공되지 않은 경우, 기본 설정 10).
- 스킬 레벨이 다른 플레이어와 비슷한지 여부를 바탕으로 플레이어를 선택합니다. 두 팀의 평균 플레이어 스킬이 10점 이내가 되도록 합니다.
- 매치가 신속하게 성사되지 않을 경우, 플레이어의 스킬 요건을 완화하여 합리적인 시간 내에서 매치가 성사되도록 합니다.
  - 5초 후에 검색을 확장하여 팀들의 평균 플레이어 스킬 차이를 50점 이내로 완화합니다.
  - 15초 후에 검색을 확장하여 팀들의 평균 플레이어 스킬 차이를 100점 이내로 완화합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 이 예제에서는 팀의 규모가 4명에서 8명 사이입니다(물론 팀 간에 플레이어 수는 동일해야 함). 유효한 규모의 범위에 속하는 팀에 대해 매치메이커는 허용되는 최대 수의 플레이어를 매치하기 위해 최선의 시도를 합니다.
- `FairTeamSkill` 규칙에 의해 팀들은 플레이어 스킬을 바탕으로 균등하게 매치됩니다. 새로운 각 잠재 플레이어에 대해 이 규칙을 평가하기 위해 FlexMatch 는 임시로 플레이어를 팀에 추가하고 평균을 계산합니다. 규칙에 부합하지 않은 경우 해당 잠재 플레이어는 매치에 추가되지 않습니다.
- 두 팀의 구조가 서로 일치하므로 팀 정의를 하나만 생성하고 팀 수량을 "2"로 설정하도록 선택할 수 있었습니다. 이 시나리오에서는 팀 이름을 "aliens"로 지정했으므로 팀에 "aliens_1" 및 "aliens_2" 이름이 할당되었습니다.

```
{
    "name": "aliens_vs_cowboys",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
        "default": 10
    }],
```

```
    "teams": [{
        "name": "cowboys",
        "maxPlayers": 8,
        "minPlayers": 4
    }, {
        "name": "aliens",
        "maxPlayers": 8,
        "minPlayers": 4
    }],
    "rules": [{
        "name": "FairTeamSkill",
        "description": "The average skill of players in each team is within 10 points
 from the average skill of all players in the match",
        "type": "distance",
        // get skill values for players in each team and average separately to produce
 list of two numbers
        "measurements": [ "avg(teams[*].players.attributes[skill])" ],
        // get skill values for players in each team, flatten into a single list, and
 average to produce an overall average
        "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10 // minDistance would achieve the opposite result
    }, {
        "name": "EqualTeamSizes",
        "description": "Only launch a game when the number of players in each team
 matches, e.g. 4v4, 5v5, 6v6, 7v7, 8v8",
        "type": "comparison",
        "measurements": [ "count(teams[cowboys].players)" ],
        "referenceValue": "count(teams[aliens].players)",
        "operation": "=" // other operations: !=, <, <=, >, >=
    }],
    "expansions": [{
        "target": "rules[FairTeamSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 5,
            "value": 50
        }, {
            "waitTimeSeconds": 15,
            "value": 100
        }]
    }]
}
```

예제 2: 불균일 팀 생성(헌터 대 몬스터)

이 예제에서는 플레이어 그룹이 단일 몬스터를 사냥하는 게임 모드를 설명합니다. 플레이어는 헌터와 몬스터 중 하나를 선택합니다. 헌터는 사냥하고자 하는 몬스터의 최소 스킬을 지정합니다. 매치 성사를 위해 헌터 팀의 최소 규모를 시간 경과에 따라 완화할 수 있습니다. 이 시나리오에서는 다음과 같은 지침을 설정합니다.

- 정확히 5명의 헌터로 구성되는 팀을 생성합니다.
- 정확히 1명의 몬스터로 구성되는 별도의 팀을 생성합니다.
- 다음 플레이어 속성을 포함합니다.
  - 플레이어의 스킬 레벨(제공되지 않은 경우, 기본 설정 10).
  - 플레이어가 선호하는 몬스터 스킬 레벨(제공되지 않은 경우, 기본 설정 10).
  - 플레이어가 몬스터가 되고 싶은지 여부(제공되지 않은 경우, 기본 설정 0 또는 F).
- 다음 기준에 따라 몬스터가 될 플레이어를 선택합니다.
  - 플레이어가 몬스터 역할을 요청해야 합니다.
  - 플레이어가 헌터 팀에 이미 속한 플레이어가 선호하는 최고 스킬 레벨 이상을 충족해야 합니다.
- 다음 기준에 따라 헌터 팀 플레이어를 선택합니다.
  - 몬스터 역할을 요청한 플레이어는 헌터 팀에 합류할 수 없습니다.
  - 몬스터 역할이 충원된 경우 플레이어가 원하는 몬스터의 스킬 레벨이 제안된 몬스터의 스킬보다 낮아야 합니다.
- 매치가 신속하게 이루어지지 않는 경우, 다음과 같이 헌터 팀의 최소 규모를 완화합니다.
  - 30초 후에는 네 명의 플레이어만 헌터 팀에 속해도 게임이 시작될 수 있도록 합니다.
  - 60초 후에는 세 명의 플레이어만 헌터 팀에 속해도 게임이 시작될 수 있도록 합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 헌터와 몬스터 두 개의 별도 팀을 사용함으로써 각기 다른 일단의 기준으로 바탕으로 회원을 평가할 수 있습니다.

```
{
    "name": "players_vs_monster_5_vs_1",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
```

```
        "default": 10
    },{
        "name": "desiredSkillOfMonster",
        "type": "number",
        "default": 10
    },{
        "name": "wantsToBeMonster",
        "type": "number",
        "default": 0
    }],
    "teams": [{
        "name": "players",
        "maxPlayers": 5,
        "minPlayers": 5
    }, {
        "name": "monster",
        "maxPlayers": 1,
        "minPlayers": 1
    }],
    "rules": [{
        "name": "MonsterSelection",
        "description": "Only users that request playing as monster are assigned to the
monster team",
        "type": "comparison",
        "measurements": ["teams[monster].players.attributes[wantsToBeMonster]"],
        "referenceValue": 1,
        "operation": "="
    },{
        "name": "PlayerSelection",
        "description": "Do not place people who want to be monsters in the players
team",
        "type": "comparison",
        "measurements": ["teams[players].players.attributes[wantsToBeMonster]"],
        "referenceValue": 0,
        "operation": "="
    },{
        "name": "MonsterSkill",
        "description": "Monsters must meet the skill requested by all players",
        "type": "comparison",
        "measurements": ["avg(teams[monster].players.attributes[skill])"],
        "referenceValue":
"max(teams[players].players.attributes[desiredSkillOfMonster])",
        "operation": ">="
    }],
```

```
    "expansions": [{
        "target": "teams[players].minPlayers",
        "steps": [{
            "waitTimeSeconds": 30,
            "value": 4
        },{
            "waitTimeSeconds": 60,
            "value": 3
        }]
    }]
}
```

예제 3: 팀 레벨 요구 사항 및 지연 시간 제한 설정

이 예제에서는 플레이어 팀을 설정하고 각 개별 플레이어 대신 각 팀에 규칙 세트를 적용하는 방법을 설명합니다. 여기서는 단일 정의를 사용하여 동등하게 매칭된 3개의 팀을 생성합니다. 또한 모든 플레이어에 대해 최대 지연 시간을 설정해 봅니다. 매치 성사를 위해 시간 경과에 따라 최대 지연 시간을 완화할 수 있습니다. 이 예제에서는 다음과 같은 지침을 설정합니다.

- 세 개의 플레이어 팀을 생성합니다.
  - 각 팀마다 3~5명의 플레이어를 포함합니다.
  - 최종적으로 팀들은 동일한 혹은 거의 동일한(차이: 1명 이내) 수의 플레이어를 포함해야 합니다.
- 다음 플레이어 속성을 포함합니다.
  - 플레이어의 스킬 레벨(제공되지 않은 경우, 기본 설정 10).
  - 플레이어의 캐릭터 역할(제공되지 않은 경우, 기본 설정 "농부").
- 매치에서 스킬 레벨이 다른 플레이어와 비슷한지 여부를 바탕으로 플레이어를 선택합니다.
  - 각 팀의 평균 플레이어 스킬이 10점 이내가 되도록 합니다.
- 팀의 "메딕" 캐릭터 수는 다음과 같이 제한합니다.
  - 매치 전체적으로 메딕의 최대 수는 5명입니다.
- 50밀리초 이하의 지연 시간을 보고하는 플레이어만 매칭합니다.
- 매치가 신속하게 이루어지지 않는 경우, 다음과 같이 플레이어 지연 시간 요건을 완화합니다.
  - 10초 후에는 플레이어 지연 시간 값을 최대 100ms로 완화합니다.
  - 20초 후에는 플레이어 지연 시간 값을 최대 150ms로 완화합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 규칙 세트에 의해 팀들은 플레이어 스킬을 바탕으로 균등하게 매치됩니다. `FairTeamSkill`규칙을 평가하기 위해 예비 선수를 팀에 FlexMatch 임시로 추가하고 팀 내 선수들의 평균 실력을 계산합니다. 그런 다음 해당 평균 스킬을 두 팀에 있는 플레이어의 평균 스킬과 비교합니다. 규칙에 부합하지 않은 경우 해당 잠재 플레이어는 매치에 추가되지 않습니다.
- 팀 및 매치 레벨 요건(총 메딕 수)은 수집 규칙을 통해 달성합니다. 이 규칙 유형은 모든 플레이어의 캐릭터 속성 목록을 취하고 최대 카운트에 대해 확인합니다. `flatten`을 사용하여 모든 팀의 모든 플레이어 목록을 생성합니다.
- 지연 시간을 기준으로 평가할 때 다음에 유의합니다.
  - 지연 시간 데이터는 플레이어 객체의 일부로 매치메이킹 요청에 제공됩니다. 플레이어 속성이 아니므로 나열될 필요가 없습니다.
  - 매치메이커는 리전 별로 지연 시간을 평가합니다. 리전을 불문하고 지연 시간의 최대치보다 긴 경우 무시됩니다. 매치에 대해 수락되기 위해서는 플레이어의 지연 시간이 적어도 한 리전에서 최대치보다 짧아야 합니다.
  - 매치메이킹 요청에서 하나 이상의 플레이어에 대한 지연 시간 데이터가 누락된 경우 모든 매치에 대해 요청이 거부됩니다.

```
{
    "name": "three_team_game",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
        "default": 10
    },{
        "name": "character",
        "type": "string_list",
        "default": [ "peasant" ]
    }],
    "teams": [{
        "name": "trio",
        "minPlayers": 3,
        "maxPlayers": 5,
        "quantity": 3
    }],
    "rules": [{
        "name": "FairTeamSkill",
        "description": "The average skill of players in each team is within 10 points
 from the average skill of players in the match",
```

```
        "type": "distance",
        // get players for each team, and average separately to produce list of 3
        "measurements": [ "avg(teams[*].players.attributes[skill])" ],
        // get players for each team, flatten into a single list, and average to
 produce overall average
        "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10 // minDistance would achieve the opposite result
    }, {
        "name": "CloseTeamSizes",
        "description": "Only launch a game when the team sizes are within 1 of each
 other.  e.g. 3 v 3 v 4 is okay, but not 3 v 5 v 5",
        "type": "distance",
        "measurements": [ "max(count(teams[*].players))"],
        "referenceValue": "min(count(teams[*].players))",
        "maxDistance": 1
    }, {
        "name": "OverallMedicLimit",
        "description": "Don't allow more than 5 medics in the game",
        "type": "collection",
        // This is similar to above, but the flatten flattens everything into a single
        // list of characters in the game.
        "measurements": [ "flatten(teams[*].players.attributes[character])"],
        "operation": "contains",
        "referenceValue": "medic",
        "maxCount": 5
    }, {
        "name": "FastConnection",
        "description": "Prefer matches with fast player connections first",
        "type": "latency",
        "maxLatency": 50
    }],
    "expansions": [{
        "target": "rules[FastConnection].maxLatency",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 100
        }, {
            "waitTimeSeconds": 20,
            "value": 150
        }]
    }]
}
```

예제 4: 최선의 매치 검색을 위한 명시적 정렬 사용

이 예제에서는 플레이어 수가 각 세 명인 두 팀으로 구성된 간단한 매치를 설정합니다. 가능한 최선의 매치를 최대한 빨리 찾기 위해 명시적 정렬 규칙을 사용하는 방법을 설명합니다. 이들 규칙은 모든 액티브 매치메이킹 티켓을 정렬하여 특정 키 요건을 기준으로 최선의 매치를 생성합니다. 이 예제는 다음 지침으로 구현됩니다.

- 두 개의 플레이어 팀을 생성합니다.
- 각 팀마다 정확히 3명의 플레이어를 포함합니다.
- 다음 플레이어 속성을 포함합니다.
  - 경험 레벨(제공되지 않은 경우, 기본 설정 50).
  - 선호 게임 모드(복수 값 나열 가능)(제공되지 않은 경우 기본 설정 "협동" 및 "데스매치").
  - 선호 게임 맵(맵 이름 및 선호 가중치 포함)(제공되지 않은 경우 기본 설정 `"defaultMap"` 및 가중치 100).
- 사전 정렬 설정:
  - 앵커 플레이어와 동일한 게임 맵에 대한 선호를 기준으로 플레이어를 정렬합니다. 플레이어가 복수의 희망 게임 맵을 가질 수 있으므로 이 예제에서는 선호 값을 사용합니다.
  - 경험 레벨이 앵커 플레이어와 얼마나 근접한지를 기준으로 플레이어를 정렬합니다. 이렇게 정렬하고 나면 모든 팀에 있는 모든 플레이어가 최대한 근접하는 경험 레벨을 가지게 될 것입니다.
- 모든 팀에 걸쳐 모든 플레이어가 선택한 게임 모드 중 공통 게임 모드가 하나 이상 있어야 합니다.
- 모든 팀에 걸쳐 모든 플레이어가 선택한 게임 맵 중 공통 맵이 하나 이상 있어야 합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 게임 맵 정렬은 mapPreference 속성 값을 비교하는 절대 정렬을 사용합니다. 이 정렬은 규칙 세트의 첫 번째 정렬이므로 맨 처음 수행됩니다.
- 경험 정렬은 거리 정렬을 사용하여 잠재 플레이어의 스킬 레벨을 앵커 플레이어의 스킬과 비교합니다.
- 정렬은 규칙 세트에 나열된 순서대로 수행됩니다. 이 시나리오에서는 플레이어를 게임 맵 선호도와 그 다음으로는 경험 레벨을 기준으로 정렬합니다.

```
{
    "name": "multi_map_game",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
```

```
        "name": "experience",
        "type": "number",
        "default": 50
    }, {
        "name": "gameMode",
        "type": "string_list",
        "default": [ "deathmatch", "coop" ]
    }, {
        "name": "mapPreference",
        "type": "string_number_map",
        "default": { "defaultMap": 100 }
    }, {
        "name": "acceptableMaps",
        "type": "string_list",
        "default": [ "defaultMap" ]
    }],
    "teams": [{
        "name": "red",
        "maxPlayers": 3,
        "minPlayers": 3
    }, {
        "name": "blue",
        "maxPlayers": 3,
        "minPlayers": 3
    }],
    "rules": [{
        // We placed this rule first since we want to prioritize players preferring the
same map
        "name": "MapPreference",
        "description": "Favor grouping players that have the highest map preference
aligned with the anchor's favorite",
        // This rule is just for sorting potential matches.  We sort by the absolute
value of a field.
        "type": "absoluteSort",
        // Highest values go first
        "sortDirection": "descending",
        // Sort is based on the mapPreference attribute.
        "sortAttribute": "mapPreference",
        // We find the key in the anchor's mapPreference attribute that has the highest
value.
        // That's the key that we use for all players when sorting.
        "mapKey": "maxValue"
    }, {
```

```
        // This rule is second because any tie-breakers should be ordered by similar
 experience values
        "name": "ExperienceAffinity",
        "description": "Favor players with similar experience",
        // This rule is just for sorting potential matches.  We sort by the distance
 from the anchor.
        "type": "distanceSort",
        // Lowest distance goes first
        "sortDirection": "ascending",
        "sortAttribute": "experience"
    }, {
        "name": "SharedMode",
        "description": "The players must have at least one game mode in common",
        "type": "collection",
        "operation": "intersection",
        "measurements": [ "flatten(teams[*].players.attributes[gameMode])"],
        "minCount": 1
    }, {
        "name": "MapOverlap",
        "description": "The players must have at least one map in common",
        "type": "collection",
        "operation": "intersection",
        "measurements": [ "flatten(teams[*].players.attributes[acceptableMaps])"],
        "minCount": 1
    }]
}
```

예제 5: 복수 플레이어 속성 간의 교집합 찾기

이 예제에서는 두 개 이상의 플레이어 속성 간에 교집합을 찾기 위해 수집 규칙을 사용하는 방법을 보여줍니다. 속성 모음을 처리할 때 단일 속성의 경우 `intersection` 작업을, 다중 속성의 경우 `reference_intersection_count` 작업을 사용할 수 있습니다.

이러한 접근 방식을 보여주기 위해 이 예제에서는 선호하는 캐릭터를 토대로 매치에서 플레이어를 평가하고 있습니다. 예시 게임은 경기에 참가한 모든 플레이어가 상대가 되는 free-for-all "" 스타일 게임입니다. 각 플레이어는 (1) 자신의 캐릭터를 선택하고, (2) 게임 상대가 될 캐릭터를 선택하라는 요청 메시지를 받게 됩니다. 매치에 참여하는 모든 플레이어가 다른 모든 플레이어의 선호 상대 목록에 있는 캐릭터를 사용하도록 하는 규칙이 필요합니다.

다음은 다음과 같은 특성이 있는 매치를 설명하는 규칙 세트의 예제입니다.

- 팀 구성: 5명의 플레이어로 구성된 하나의 팀

- 플레이어 속성:
  - myCharacter: 플레이어가 선택한 캐릭터입니다.
  - preferredOpponents: 플레이어가 게임 상대로 선택하고 싶은 캐릭터의 목록입니다.
- 매치 규칙: 사용 중인 각 캐릭터가 모든 플레이어의 선호 상대 목록에 있는 경우에 잠재적 매치가 허용됩니다.

매치 규칙을 실행하기 위해 이 예제에서는 다음과 같은 속성 값을 가진 수집 규칙을 사용하고 있습니다.

- 작업 - `reference_intersection_count` 작업을 사용하여 측정 값의 각 문자열 목록이 참조 값의 문자열 목록과 어떻게 교집합을 이루는지 평가합니다.
- 측정 - `flatten` 속성 표현식을 사용하여 문자열 목록을 생성합니다. 각 문자열 목록에는 한 플레이어의 myCharacter 속성 값이 포함됩니다.
- 참조 값 - `set_intersection` 속성 표현식을 사용하여 매치에서 모든 플레이어에게 공통적인 preferredOpponents 속성 값 모두에 대한 문자열 목록을 생성합니다.
- 제한 - 각 플레이어가 선택한 캐릭터(측정 시 문자열 목록)는 모든 플레이어에 공통되는 선호 상대 중 최소 한 명과 매치될 수 있도록 `minCount`가 1로 설정됩니다.
- 확장 - 15초 내에 매치가 성사되지 않으면 최소한의 교집합 요구 사항을 완화합니다.

이 규칙의 프로세스 흐름은 다음과 같습니다.

1. 해당 매치에 플레이어가 추가됩니다. 새 플레이어가 선호하는 상대 목록과의 교집합을 포함시켜 참조 값(문자열 목록)이 다시 계산됩니다. 새 플레이어가 선택한 캐릭터를 새 문자열 목록으로 추가하여 측정 값(문자열 목록의 목록)이 다시 계산됩니다.
2. Amazon은 측정 값의 각 문자열 목록 (플레이어가 선택한 문자) 이 참조 값에 있는 하나 이상의 문자열 (플레이어가 선호하는 상대) 과 GameLift 교차하는지 확인합니다. 이 예제에서는 측정 값의 각 문자열 목록에 오직 하나의 값이 포함되어 있기 때문에 교집합은 0 또는 1입니다.
3. 측정 값의 어떤 문자열 목록도 참조 값 문자열 목록과 교집합을 이루지 않을 경우에는 규칙이 실패하고 새 플레이어가 해당 매치에서 제거됩니다.
4. 15초 내에 매치가 성사되지 않을 경우, 상대 매치 요건을 낮춰서 매치에서 나머지 플레이어 슬롯을 채웁니다.

```
{
    "name": "preferred_characters",
```

```
    "ruleLanguageVersion": "1.0",

    "playerAttributes": [{
        "name": "myCharacter",
        "type": "string_list"
    }, {
        "name": "preferredOpponents",
        "type": "string_list"
    }],

    "teams": [{
        "name": "red",
        "minPlayers": 5,
        "maxPlayers": 5
    }],

    "rules": [{
        "description": "Make sure that all players in the match are using a character
 that is on all other players' preferred opponents list.",
        "name": "OpponentMatch",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": ["flatten(teams[*].players.attributes[myCharacter])"],
        "referenceValue":
 "set_intersection(flatten(teams[*].players.attributes[preferredOpponents]))",
        "minCount":1
    }],
    "expansions": [{
        "target": "rules[OpponentMatch].minCount",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 0
        }]
    }]
}
```

예제 6: 모든 플레이어를 대상으로 속성 비교

이 예제에서는 플레이어 그룹에서 플레이어 속성을 비교하는 방법을 보여줍니다.

다음은 다음과 같은 특성이 있는 매치를 설명하는 규칙 세트의 예제입니다.

- 팀 구조: 두 개의 단일 플레이어 팀

- 플레이어 속성:
  - gameMode: 플레이어가 선택한 게임 유형입니다(제공되지 않는 경우 기본 설정 "순서 기반").
  - gameMap: 플레이어가 선택한 게임 세상입니다(제공되지 않는 경우 기본 설정 1).
  - character: 플레이어가 선택한 캐릭터입니다. 여기에 기본값이 없으면 플레이어가 캐릭터를 지정해야 합니다.
- 매치 규칙: 매치된 플레이어는 다음 요구 사항을 충족해야 합니다.
  - 플레이어들이 동일한 게임 모드를 선택해야 합니다.
  - 플레이어들이 동일한 게임 맵을 선택해야 합니다.
  - 플레이어들이 서로 다른 캐릭터를 선택해야 합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 매치 규칙을 실행하기 위해 이 예제에서는 비교 규칙을 사용하여 모든 플레이어의 속성 값을 확인하고 있습니다. 게임 모드 및 맵에서 규칙은 값들이 동일한지 확인합니다. 각 캐릭터에서 규칙은 값들이 서로 다른지 확인합니다.
- 이 예제에서는 하나의 플레이어 정의를 수량 속성과 함께 사용하여 두 플레이어 팀을 모두 생성합니다. 팀에 "player_1" 및 "player_2" 이름이 지정됩니다.

```
{
    "name": "",
    "ruleLanguageVersion": "1.0",

    "playerAttributes": [{
        "name": "gameMode",
        "type": "string",
        "default": "turn-based"
    }, {
        "name": "gameMap",
        "type": "number",
        "default": 1
    }, {
        "name": "character",
        "type": "number"
    }],

    "teams": [{
        "name": "player",
```

```
        "minPlayers": 1,
        "maxPlayers": 1,
        "quantity": 2
    }],

    "rules": [{
        "name": "SameGameMode",
        "description": "Only match players when they choose the same game type",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[gameMode])"]
    }, {
        "name": "SameGameMap",
        "description": "Only match players when they're in the same map",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[gameMap])"]
    }, {
        "name": "DifferentCharacter",
        "description": "Only match players when they're using different characters",
        "type": "comparison",
        "operation": "!=",
        "measurements": ["flatten(teams[*].players.attributes[character])"]
    }]
}
```

예제 7: 라지 매치 생성

이 예제에서는 플레이어 수가 40명을 초과할 수 있는 매치에 대한 규칙 세트를 설정하는 방법을 설명합니다. 규칙 세트에서 총 maxPlayer 수가 40보다 큰 팀을 설명할 때 라지 매치로 처리됩니다. FlexMatch 라지 매치 규칙 세트 설계에서 자세히 알아보세요.

이 예제 규칙 세트는 다음 지침에 따라 매치를 생성합니다.

- 최소 요구 사항이 175명의 플레이어인 최대 200명의 플레이어로 이루어진 팀 하나를 생성합니다.
- 밸런싱 기준: 유사한 스킬 레벨에 따라 플레이어를 선택합니다. 모든 플레이어는 매칭될 자신의 스킬 레벨을 보고해야 합니다.
- 일괄 처리 선호도: 매치를 생성할 때 유사한 밸런싱 기준에 따라 플레이어를 그룹화합니다.
- 지연 시간 규칙: 최대 허용 가능 플레이어 지연 시간을 150밀리초로 설정합니다.
- 매치가 신속하게 채워지지 않는 경우 적합한 시간 내에 매치를 완료할 수 있도록 요구 사항을 완화합니다.

- 10초 후에 플레이어가 150명인 팀을 수락합니다.
- 12초 후에 최대 허용 가능 지연 시간을 200밀리초로 늘립니다.
- 15초 후에 플레이어가 100명인 팀을 수락합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 이 알고리즘은 "가장 큰 모집단" 일괄 처리 선호도를 사용하므로 밸런싱 기준에 따라 플레이어가 먼저 정렬됩니다. 따라서 매치는 더 채워지려고 하며 스킬이 더 유사한 플레이어를 포함하려고 합니다. 모든 플레이어는 허용 가능한 지연 시간 요구 사항을 충족하지만 자신의 위치에 대해 최상의 지연 시간을 얻지 못할 수도 있습니다.
- 이 규칙 세트에서 사용되는 알고리즘 전략인 "가장 큰 모집단"이 기본 설정입니다. 기본값을 사용하기 위해 이 설정을 생략하도록 선택할 수 있습니다.
- 매치 채우기를 활성화한 경우에는 플레이어 수 요구 사항을 너무 빨리 완화하지 마십시오. 부분적으로 채워진 게임 세션이 너무 많아질 수 있습니다. 라지 매치 요구 사항 완화에서 자세히 알아보세요.

```
{
    "name": "free-for-all",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number"
    }],
    "algorithm": {
        "balancedAttribute": "skill",
        "strategy": "balanced",
        "batchingPreference": "largestPopulation"
    },
    "teams": [{
        "name": "Marauders",
        "maxPlayers": 200,
        "minPlayers": 175
    }],
    "rules": [{
        "name": "low-latency",
        "description": "Sets maximum acceptable latency",
        "type": "latency",
        "maxLatency": 150
    }],
    "expansions": [{
```

```
        "target": "rules[low-latency].maxLatency",
        "steps": [{
            "waitTimeSeconds": 12,
            "value": 200
        }],
    }, {
        "target": "teams[Marauders].minPlayers",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 150
        }, {
            "waitTimeSeconds": 15,
            "value": 100
        }]
    }]
}
```

예제 8: 복수 팀 라지 매치 생성

이 예제에서는 플레이어 수가 40명을 초과할 수 있는 복수 팀의 매치에 대한 규칙 세트를 설정하는 방법을 설명합니다. 이 예제에서는 하나의 정의로 동일한 복수 팀을 생성하는 방법 및 매치를 생성하는 동안 비대칭으로 크기가 지정된 팀을 채우는 방법을 설명합니다.

이 예제 규칙 세트는 다음 지침에 따라 매치를 생성합니다.

- 플레이어 수가 최대 15명인 동일한 "헌터" 팀 10개와 플레이어 수가 정확하게 5명인 "몬스터" 팀 한 개를 생성합니다.
- 밸런싱 기준: 몬스터 처리 수에 따라 플레이어를 선택합니다. 플레이어가 자신의 처리 수를 보고하지 않는 경우 기본값인 5를 사용합니다.
- 일괄 처리 선호도: 가장 빠른 플레이어 지연 시간을 보고하는 리전에 따라 플레이어를 그룹화합니다.
- 지연 시간 규칙: 최대 허용 가능 플레이어 지연 시간을 200밀리초로 설정합니다.
- 매치가 신속하게 채워지지 않는 경우 적합한 시간 내에 매치를 완료할 수 있도록 요구 사항을 완화합니다.
  - 15초 후에 플레이어가 10명인 팀을 수락합니다.
  - 20초 후에 플레이어가 8명인 팀을 수락합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 이 규칙 세트는 잠재적으로 최대 155명의 플레이어를 포함할 수 있는 팀을 정의합니다. 이로 인해 라지 매치가 됩니다.(10 x 15명의 헌터 + 5명의 몬스터 = 155)
- 이 알고리즘은 "가장 빠른 리전" 일괄 처리 선호도를 사용하므로 플레이어가 보다 빠른 지연 시간을 보고하는 리전에 배치되려고 하며 높지만 허용 가능한 지연 시간을 보고하는 리전에는 배치되지 않으려는 경향을 보입니다. 동시에 매치에 더 적은 수의 플레이어가 포함될 가능성이 높아지고 밸런싱 기준(몬스터 스킬 수)이 더 다양해질 수도 있습니다.
- 복수 팀 정의(수량 > 1)에 대한 확장이 정의되면 해당 확장이 해당 정의로 생성된 모든 팀에 적용됩니다. 따라서 헌터 팀의 최소 플레이어 설정을 완화함으로써 10개의 헌터 팀 모두가 동등한 영향을 받게 됩니다.
- 이 규칙 세트는 플레이어 지연 시간을 최소화하도록 최적화되어 있으므로 지연 시간 규칙이 허용 가능한 연결 옵션이 없는 플레이어를 제외하는 catch-all로 사용됩니다. 이 요구 사항을 완화할 필요가 없습니다.
- 확장이 적용되기 전에 이 규칙 세트의 일치 항목을 FlexMatch 채우는 방법은 다음과 같습니다.
  - 아직 minPlayers 수에 도달한 팀이 없습니다. 헌터 팀의 열린 슬롯은 15개이고 몬스터 팀의 열린 슬롯은 5개입니다.
    - 첫 100명의 플레이어(10명씩)가 10개의 헌터 팀에 할당됩니다.
    - 다음 22명의 플레이어가 순차적으로(2명씩) 헌터 팀과 몬스터 팀에 할당됩니다.
  - 헌터 팀이 각각 12명의 플레이어인 minPlayers 수에 도달했습니다. 몬스터 팀에 2명의 플레이어가 있으며 아직 minPlayers 수에 도달하지 않았습니다.
    - 다음 3명의 플레이어가 몬스터 팀에 할당됩니다.
  - 모든 팀이 minPlayers 수에 도달했습니다. 헌터 팀에는 각각 3개의 열린 슬롯이 있습니다. 몬스터 팀이 채워졌습니다.
    - 마지막 30명의 플레이어가 순차적으로 헌터 팀에 할당됩니다. 따라서 모든 헌터 팀의 크기가 거의 비슷해집니다(한 명의 플레이어가 더 많거나 더 적음).
- 이 규칙 세트로 생성한 매치에 대한 채우기를 활성화한 경우에는 플레이어 수 요구 사항을 너무 빨리 완화하지 마십시오. 부분적으로 채워진 게임 세션이 너무 많아질 수 있습니다. 라지 매치 요구 사항 완화에서 자세히 알아보세요.

```
{
    "name": "monster-hunters",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "monster-kills",
        "type": "number",
```

```
        "default": 5
    }],
    "algorithm": {
        "balancedAttribute": "monster-kills",
        "strategy": "balanced",
        "batchingPreference": "fastestRegion"
    },
    "teams": [{
        "name": "Monsters",
        "maxPlayers": 5,
        "minPlayers": 5
    }, {
        "name": "Hunters",
        "maxPlayers": 15,
        "minPlayers": 12,
        "quantity": 10
    }],
    "rules": [{
        "name": "latency-catchall",
        "description": "Sets maximum acceptable latency",
        "type": "latency",
        "maxLatency": 150
    }],
    "expansions": [{
        "target": "teams[Hunters].minPlayers",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 10
        }, {
            "waitTimeSeconds": 20,
            "value": 8
        }]
    }]
}
```

예제 9: 비슷한 특성을 가진 플레이어로 라지 매치 생성

이 예제는 batchDistance를 사용하여 두 팀이 참가하는 매치에 대한 규칙 세트를 설정하는 방법을 보여줍니다. 예제에서는:

- SimilarLeague 규칙은 매치에 참가하는 모든 플레이어가 다른 플레이어 2명 이내에 league를 갖도록 보장합니다.

- SimilarSkill 규칙은 매치에 참가하는 모든 플레이어가 다른 플레이어 10명 이내에 skill을 갖도록 보장합니다. 플레이어가 10초 동안 기다린 경우 거리는 20으로 확장됩니다. 플레이어가 20초 동안 기다린 경우 거리는 40으로 확장됩니다.
- SameMap 규칙은 매치에 참가한 모든 플레이어가 동일한 map를 요청하도록 보장합니다.
- SameMode 규칙은 매치에 참가한 모든 플레이어가 동일한 mode를 요청하도록 보장합니다.

```
{
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "name": "red",
        "minPlayers": 100,
        "maxPlayers": 100
    }, {
        "name": "blue",
        "minPlayers": 100,
        "maxPlayers": 100
    }],
    "algorithm": {
        "strategy":"balanced",
        "balancedAttribute": "skill",
        "batchingPreference":"fastestRegion"
    },
    "playerAttributes": [{
        "name": "league",
        "type": "number"
    },{
        "name": "skill",
        "type": "number"
    },{
        "name": "map",
        "type": "string"
    },{
        "name": "mode",
        "type": "string"
    }],
    "rules": [{
        "name": "SimilarLeague",
        "type": "batchDistance",
        "batchAttribute": "league",
        "maxDistance": 2
    }, {
```

```
        "name": "SimilarSkill",
        "type": "batchDistance",
        "batchAttribute": "skill",
        "maxDistance": 10
    }, {
        "name": "SameMap",
        "type": "batchDistance",
        "batchAttribute": "map"
    }, {
        "name": "SameMode",
        "type": "batchDistance",
        "batchAttribute": "mode"
    }],
    "expansions": [{
        "target": "rules[SimilarSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 20
        }, {
            "waitTimeSeconds": 20,
            "value": 40
        }]
    }]
}
```

예 10: 복합 규칙을 사용하여 비슷한 속성이나 비슷한 선택을 가진 플레이어로 매치 생성

이 예제는 compound를 사용하여 두 팀이 참가하는 매치에 대한 규칙 세트를 설정하는 방법을 보여줍니다. 예제에서는:

- SimilarLeagueDistance 규칙은 매치에 참가하는 모든 플레이어가 다른 플레이어 2명 이내에 league를 갖도록 보장합니다.
- SimilarSkillDistance 규칙은 매치에 참가하는 모든 플레이어가 다른 플레이어 10명 이내에 skill을 갖도록 보장합니다. 플레이어가 10초 동안 기다린 경우 거리는 20으로 확장됩니다. 플레이어가 20초 동안 기다린 경우 거리는 40으로 확장됩니다.
- SameMapComparison 규칙은 매치에 참가한 모든 플레이어가 동일한 map를 요청하도록 보장합니다.
- SameModeComparison 규칙은 매치에 참가한 모든 플레이어가 동일한 mode를 요청하도록 보장합니다.

- 다음 조건 중 하나에 해당하면 CompoundRuleMatchmaker 규칙이 일치하는지 확인합니다.
  - 매치에 참가한 플레이어도 동일한 map 및 mode 요청을 수행했습니다.
  - 매치에 참가한 플레이어는 비슷한 skill 및 league 속성을 가지고 있습니다.

```
{
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "name": "red",
        "minPlayers": 10,
        "maxPlayers": 20
    }, {
        "name": "blue",
        "minPlayers": 10,
        "maxPlayers": 20
    }],
    "algorithm": {
        "strategy":"balanced",
        "balancedAttribute": "skill",
        "batchingPreference":"fastestRegion"
    },
    "playerAttributes": [{
        "name": "league",
        "type": "number"
    },{
        "name": "skill",
        "type": "number"
    },{
        "name": "map",
        "type": "string"
    },{
        "name": "mode",
        "type": "string"
    }],
    "rules": [{
        "name": "SimilarLeagueDistance",
        "type": "distance",
        "measurements": ["max(flatten(teams[*].players.attributes[league]))"],
        "referenceValue": "min(flatten(teams[*].players.attributes[league]))",
        "maxDistance": 2
    }, {
        "name": "SimilarSkillDistance",
        "type": "distance",
```

```
        "measurements": ["max(flatten(teams[*].players.attributes[skill]))"],
        "referenceValue": "min(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10
    }, {
        "name": "SameMapComparison",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[map])"]
    }, {
        "name": "SameModeComparison",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[mode])"]
    }, {
        "name": "CompoundRuleMatchmaker",
        "type": "compound",
        "statement": "or(and(SameMapComparison, SameModeComparison),
 and(SimilarSkillDistance, SimilarLeagueDistance))"
    }],
    "expansions": [{
        "target": "rules[SimilarSkillDistance].maxDistance",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 20
        }, {
            "waitTimeSeconds": 20,
            "value": 40
        }]
    }]
}
```

예제 11: 플레이어의 차단 목록을 사용하는 규칙 생성

이 예제는 플레이어가 다른 특정 플레이어와 매칭되는 것을 피할 수 있는 규칙 세트를 보여줍니다. 플레이어는 차단 목록을 생성할 수 있으며, 매치메이커는 매치에 참가할 플레이어를 선택할 때 이를 평가합니다. 차단 목록 또는 금지 목록 기능을 추가하는 방법에 게임 블로그용 AWS를 참조하세요.

이 예제에서는 다음과 같은 지침을 설정합니다.

- 정확히 5명의 플레이어로 구성된 두 팀을 만듭니다..
- 플레이어 ID 목록(최대 100개)인 플레이어 차단 목록을 전달합니다.

- 모든 플레이어를 각 플레이어의 차단 목록과 비교하고 차단된 플레이어 ID가 발견되면 제안된 매치를 거부합니다.

이 규칙 세트 사용에 대한 참고 사항:

- 제안된 매치에 추가할 (또는 기존 매치의 한 스팟을 채우기 위해) 새 플레이어를 평가할 때 다음 이유 중 하나로 플레이어가가 거부될 수 있습니다.
  - 매치에 이미 선택된 플레이어의 차단 목록에 새 플레이어가 포함된 경우.
  - 새 플레이어의 차단 목록에 매치에 이미 선택된 플레이어가 포함된 경우.
- 그림에 표시된 대로 이 규칙 세트는 플레이어를 차단 목록에 있는 플레이어와 매칭하는 것을 방지합니다. 규칙 확장을 추가하고 `maxCount` 값을 높이면 이 요구 사항을 기본 설정(“방지” 목록이라고도 함)으로 변경할 수 있습니다.

```
{
    "name": "Player Block List",
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "red"
    }, {
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "blue"
    }],
    "playerAttributes": [{
        "name": "BlockList",
        "type": "string_list",
        "default": []
    }],
    "rules": [{
        "name": "PlayerIdNotInBlockList",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": "flatten(teams[*].players.attributes[BlockList])",
        "referenceValue": "flatten(teams[*].players[playerId])",
        "maxCount": 0
    }]
}
```

매치메이킹 구성 생성

Amazon GameLift FlexMatch 매치메이커가 매치메이킹 요청을 처리하도록 설정하려면 매치메이킹 구성을 생성합니다. Amazon GameLift 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용합니다. 매치메이커 생성에 대한 자세한 내용은 FlexMatch 매치메이커 디자인 섹션을 참조하세요.

Amazon GameLift 호스팅을 위한 매치메이커 생성

매치메이킹 구성을 생성하기 전에 매치메이커와 함께 사용할 규칙 세트를 생성하고 Amazon GameLift 게임 세션 대기열을 생성합니다.

Console

1. Amazon GameLift 콘솔의 탐색 창에서 매치메이킹 구성을 선택합니다.
2. 매치메이커를 생성하고자 하는 AWS 리전으로 전환합니다.
3. 매치메이킹 구성 페이지에서 매치메이킹 구성 생성을 선택합니다.
4. 구성 세부 정보 정의 페이지의 매치메이킹 구성 세부 정보에서 다음을 수행합니다.

    a. 이름에는 목록과 지표에서 매치메이커를 식별하는 데 도움이 되는 매치메이커 이름을 입력합니다. 매치메이커 이름은 리전 내에서 고유해야 합니다. 매치메이킹 요청은 이름 및 리전으로 사용할 매치메이커를 식별합니다.

    b. (선택 사항) 설명에는 매치메이커를 식별하는 데 도움이 되는 설명을 추가합니다.

    c. 규칙 세트에는 목록에서 매치메이커와 함께 사용할 규칙 세트를 선택합니다. 목록은 현재 리전에 생성된 모든 규칙 세트를 포함합니다.

    d. FlexMatch 모드의 경우 Amazon GameLift 관리형 호스팅에 대해 관리형을 선택합니다. 이 모드에서는 FlexMatch가 성공적인 매치를 지정된 게임 세션 대기열로 전달하라는 메시지를 표시합니다.

    e. AWS 리전에는 매치메이커와 함께 사용할 게임 세션 대기열을 구성한 리전을 선택합니다.

    f. 대기열에는 매치메이커와 함께 사용하려는 게임 세션 대기열을 선택합니다.
5. 다음을 선택합니다.
6. 설정 구성 페이지의 매치메이킹 설정에서 다음을 수행합니다.

    a. 요청 제한 시간에는 매치메이커가 각 요청의 매치를 성사시키는 데 대한 최대 시간(초)을 설정합니다. FlexMatch는 이 시간을 초과하는 매치메이킹 요청을 취소합니다.

    b. 채우기 모드에는 매치 채우기를 처리하기 위한 모드를 선택합니다.

    - 자동 채우기 기능을 설정하려면 자동을 선택합니다.

- 자체 채우기 요청 관리를 생성하거나 채우기 기능을 사용하지 않으려면 수동을 선택합니다.

c. (선택 사항) 추가 플레이어 수에는 매치에 계속 열어 둘 플레이어 슬롯 수를 설정합니다. FlexMatch는 이러한 슬롯을 향후에 플레이어로 채울 수 있습니다.

d. (선택 사항) 매치 수락 선택 사항의 수락 필수에서 제안된 매치의 각 플레이어가 매치 참가를 적극적으로 수락하도록 요구하려면 필수를 선택합니다. 이 선택 사항을 선택하는 경우 수락 제한 시간에 대해 매치메이커가 매치를 취소하기 전에 플레이어 수락을 기다리는 시간(초)을 설정합니다.

7. (선택 사항) 이벤트 알림 설정에서 다음을 수행합니다.

a. (선택 사항) SNS 주제의 경우 매치메이킹 이벤트 알림을 수신할 Amazon Simple Notification Service(SNS) 주제를 선택합니다. 아직 SNS 주제를 설정하지 않았다면 나중에 매치메이킹 구성 편집을 통해 이를 선택할 수 있습니다. 자세한 내용은 FlexMatch 이벤트 알림 설정 섹션을 참조하세요.

b. (선택 사항) 사용자 지정 이벤트 데이터에는 이벤트 메시징에서 이 매치메이커와 연결하고자 하는 임의의 사용자 지정 데이터를 입력합니다. FlexMatch는 매치메이커와 연결된 모든 이벤트에 이 데이터를 포함합니다.

8. (선택 사항) 추가 게임 데이터를 확장한 후 다음을 수행합니다.

a. (선택 사항) 게임 세션 데이터에는 FlexMatch가 이 매치메이킹 구성을 사용하여 만든 매치로 시작하는 새 게임 세션에 전달할 추가 게임 관련 정보를 입력합니다.

b. (선택 사항) 게임 속성의 경우 새 게임 세션에 대한 정보가 포함된 키-값 쌍 속성을 추가합니다.

9. (선택 사항) 태그에는 AWS 리소스를 관리하고 추적하는 데 도움이 되는 태그를 추가합니다.

10. 다음을 선택합니다.

11. 검토 및 생성 페이지에서 선택 사항을 검토한 다음 생성을 선택합니다. 성공적으로 생성되었으면 매치메이커가 매치메이킹 요청을 접수할 준비가 된 것입니다.

AWS CLI

AWS CLI로 매치메이킹 구성을 생성하려면 명령줄 창을 열고 create-matchmaking-configuration 명령을 이용해 새 매치메이커를 정의합니다.

이 예제 명령은 플레이어 수락이 필요한 새로운 매치메이킹 구성을 생성하고 자동 채우기를 활성화합니다. 또한 나중에 플레이어를 추가할 수 있도록 FlexMatch에 두 개의 플레이어 슬롯을 예약하고 일부 게임 세션 데이터를 제공합니다.

```
aws gamelift create-matchmaking-configuration \
    --name "SampleMatchamker123" \
    --description "The sample test matchmaker with acceptance" \
    --flex-match-mode WITH_QUEUE \
    --game-session-queue-arns "arn:aws:gamelift:us-
west-2:111122223333:gamesessionqueue/MyGameSessionQueue" \
    --rule-set-name "MyRuleSet" \
    --request-timeout-seconds 120 \
    --acceptance-required \
    --acceptance-timeout-seconds 30 \
    --backfill-mode AUTOMATIC \
    --notification-target "arn:aws:sns:us-
west-2:111122223333:My_Matchmaking_SNS_Topic" \
    --additional-player-count 2 \
    --game-session-data "key=map,value=winter444"
```

매치메이킹 구성 생성 요청이 성공하면 Amazon GameLift가 사용자가 요청한 매치메이커 설정과 함께 MatchmakingConfiguration 객체를 반환합니다. 이제 새 매치메이커가 매치메이킹 요청을 접수할 준비가 되었습니다.

독립형 FlexMatch를 위한 매치메이커 생성

매치메이킹 구성을 생성하려면 먼저 매치메이커에 사용할 규칙 세트를 생성합니다.

Console

1. https://console.aws.amazon.com/gamelift/home에서 Amazon GameLift SNS 콘솔을 엽니다.
2. 매치메이커를 생성하고자 하는 AWS 리전으로 전환합니다. FlexMatch 매치메이킹 구성을 지원하는 리전 목록은 매치메이커를 위한 위치 선택 섹션을 참조하세요.
3. 탐색 창에서 FlexMatch, 매치메이킹 구성을 선택합니다.
4. 매치메이킹 구성 페이지에서 매치메이킹 구성 생성을 선택합니다.
5. 구성 세부 정보 정의 페이지의 매치메이킹 구성 세부 정보에서 다음을 수행합니다.

    a. 이름에는 목록과 지표에서 매치메이커를 식별하는 데 도움이 되는 매치메이커 이름을 입력합니다. 매치메이커 이름은 리전 내에서 고유해야 합니다. 매치메이킹 요청은 이름 및 리전으로 사용할 매치메이커를 식별합니다.

    b. (선택 사항) 설명에는 매치메이커를 식별하는 데 도움이 되는 설명을 추가합니다.

c. 규칙 세트에는 목록에서 매치메이커와 함께 사용할 규칙 세트를 선택합니다. 목록은 현재 리전에 생성된 모든 규칙 세트를 포함합니다.

d. FlexMatch 모드의 경우 독립형을 선택합니다. 이는 Amazon GameLift 외부의 호스팅 솔루션에서 새 게임 세션을 시작하기 위한 사용자 지정 메커니즘이 있음을 나타냅니다.

6. 다음을 선택합니다.

7. 설정 구성 페이지의 매치메이킹 설정에서 다음을 수행합니다.

a. 요청 제한 시간에는 매치메이커가 각 요청의 매치를 성사시키는 데 대한 최대 시간(초)을 설정합니다. 이 제한 시간을 초과하는 매치메이킹 요청은 거부됩니다.

b. (선택 사항) 매치 수락 선택 사항의 수락 필수에서 제안된 매치의 각 플레이어가 매치 참가를 적극적으로 수락하도록 요구하려면 필수를 선택합니다. 이 선택 사항을 선택하는 경우 수락 제한 시간에 대해 매치메이커가 매치를 취소하기 전에 플레이어 수락을 기다리는 시간(초)을 설정합니다.

8. (선택 사항) 이벤트 알림 설정에서 다음을 수행합니다.

a. (선택 사항) SNS 주제의 경우 매치메이킹 이벤트 알림을 수신할 Amazon SNS 주제를 선택합니다. 아직 SNS 주제를 설정하지 않았다면 나중에 매치메이킹 구성 편집을 통해 이를 선택할 수 있습니다. 자세한 내용은 FlexMatch 이벤트 알림 설정 섹션을 참조하세요.

b. (선택 사항) 사용자 지정 이벤트 데이터에는 이벤트 메시징에서 이 매치메이커와 연결하고자 하는 임의의 사용자 지정 데이터를 입력합니다. FlexMatch는 매치메이커와 연결된 모든 이벤트에 이 데이터를 포함합니다.

9. (선택 사항) 태그에는 AWS 리소스를 관리하고 추적하는 데 도움이 되는 태그를 추가합니다.

10. 다음을 선택합니다.

11. 검토 및 생성 페이지에서 선택 사항을 검토한 다음 생성을 선택합니다. 성공적으로 생성되었으면 매치메이커가 매치메이킹 요청을 접수할 준비가 된 것입니다.

AWS CLI

AWS CLI로 매치메이킹 구성을 생성하려면 명령줄 창을 열고 create-matchmaking-configuration 명령을 이용해 새 매치메이커를 정의합니다.

이 예제 명령은 플레이어 수락이 필요한 독립형 매치메이커에 대해 새로운 매치메이킹 구성을 생성합니다.

```
aws gamelift create-matchmaking-configuration \
    --name "SampleMatchamker123" \
```

```
    --description "The sample test matchmaker with acceptance" \
    --flex-match-mode STANDALONE \
    --rule-set-name "MyRuleSetOne" \
    --request-timeout-seconds 120 \
    --acceptance-required \
    --acceptance-timeout-seconds 30 \
    --notification-target "arn:aws:sns:us-
west-2:111122223333:My_Matchmaking_SNS_Topic"
```

매치메이킹 구성 생성 요청이 성공하면 Amazon GameLift가 사용자가 요청한 매치메이커 설정과 함께 MatchmakingConfiguration 객체를 반환합니다. 이제 새 매치메이커가 매치메이킹 요청을 접수할 준비가 되었습니다.

매치메이킹 구성 편집

매치메이킹 구성을 편집하려면 탐색 모음에서 매치메이킹 구성을 선택하고 편집하려는 구성을 선택합니다. 이름을 제외한 기존 구성의 모든 필드를 업데이트할 수 있습니다.

구성 규칙 세트를 업데이트할 때 기존 활성 매치메이킹 티켓이 있는 경우 다음과 같은 이유로 새 규칙 세트가 호환되지 않을 수 있습니다.

- 새 팀 이름 또는 다른 팀 이름 또는 팀 수
- 새 플레이어 속성
- 기존 플레이어 속성 유형 변경

규칙 세트에 이러한 변경을 적용하려면 업데이트된 규칙 세트를 사용하여 새 매치메이킹 구성을 생성합니다.

FlexMatch 이벤트 알림 설정

이벤트 알림을 사용하여 개별 매치메이킹 요청 상태를 추적할 수 있습니다. 프로덕션 중인 게임이나 대량의 매치메이킹 활동이 있는 프로덕션 전 게임에서는 이벤트 알림을 사용해야 합니다.

이벤트 알림을 설정하는 데는 두 가지 옵션이 있습니다.

- 매치메이커는 Amazon Simple Notification Service(SNS) 주제에 이벤트 알림을 게시하도록 합니다.
- 자동으로 게시된 Amazon EventBridge 이벤트 및 이벤트 관리 도구 모음을 사용합니다.

Amazon GameLift에서 발생하는 FlexMatch 이벤트 목록은 FlexMatch 매치메이킹 이벤트 섹션을 참조하세요.

EventBridge 이벤트 설정

Amazon GameLift는 모든 매치메이킹 이벤트를 Amazon EventBridge에 자동으로 게시합니다. EventBridge를 사용하면 매치메이킹 이벤트를 처리 대상으로 라우팅하도록 규칙을 설정할 수 있습니다. 예를 들어, “PotentialMatchCreated” 이벤트를 플레이어 수락을 처리하는 AWS Lambda 함수로 라우팅하도록 규칙을 설정할 수 있습니다. 자세한 내용은 Amazon EventBridge란 무엇인가요? 섹션을 참조하세요.

> **Note**
>
> 매치메이커를 구성할 때 알림 대상 필드를 비워 두거나 EventBridge와 Amazon SNS를 모두 사용하려면 SNS 주제를 참조하세요.

Amazon SNS 주제 설정

Amazon GameLift가 FlexMatch 매치메이커가 생성하는 모든 이벤트를 Amazon SNS 주제에 게시하도록 할 수 있습니다.

Amazon GameLift 이벤트 알림에 대한 SNS 주제를 생성하려면

1. Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제 생성을 선택합니다.
4. 콘솔에서 주제를 생성합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 AWS Management Console을 사용하여 주제를 생성하려면을 참조하세요.
5. 주제에 대한 세부 정보 페이지에서 편집을 선택합니다.
6. (선택 사항) 해당 주제의 편집 페이지에서 액세스 정책을 확장한 후, 다음 AWS Identity and Access Management (IAM) 정책문에서 굵은 구문을 기존 정책 끝에 추가합니다. (명확성을 위해 전체 정책이 여기에 표시됩니다.) 자체 SNS 주제 및 Amazon GameLift 매치메이킹 구성에 대해서는 반드시 Amazon 리소스 이름(ARN) 세부 정보를 사용합니다.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
```

```
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
        }
      }
    }
  ]
}
```

7. 변경 사항 저장을 선택합니다.

SNS 주제에 서버 측 암호화 설정

서버 측 암호화(SSE)를 사용하면 암호화된 주제에서 민감한 데이터를 저장할 수 있습니다. SSE는 AWS Key Management Service(AWS KMS)에서 관리되는 키를 사용하여 Amazon SNS 주제의 메시지 내용을 보호합니다. Amazon SNS를 포함한 서버 측 암호화에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 저장 시 암호화를 참조하세요.

SNS 주제에 서버 측 암호화를 설정하려면 다음 주제를 검토합니다.

- AWS Key Management Service 개발자 가이드의 키 생성
- Amazon Simple Notification Service 개발자 가이드의 주제 SSE 활성화

KMS 키를 생성할 때는 다음 KMS 키 정책을 사용합니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "gamelift.amazonaws.com"
  },
  "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
      "ArnLike": {
        "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
      },
      "StringEquals": {
        "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:your_account:your_sns_topic_name"
      }
  }
}
```

Lambda 함수를 호출하기 위해 주제 구독 구성

Amazon SNS 주제에 게시된 이벤트 알림을 사용하여 Lambda 함수를 호출할 수 있습니다. 매치메이커를 구성할 때는 알림 대상을 SNS 주제의 ARN으로 설정해야 합니다.

다음 AWS CloudFormation 템플릿은 이름이 `FlexMatchEventHandlerLambdaFunction`인 Lambda 함수를 호출하도록 `MyFlexMatchEventTopic` SNS 주제에 대한 구독을 구성합니다. 템플릿은 Amazon GameLift가 SNS 주제에 글을 쓸 수 있도록 허용하는 IAM 권한 정책을 생성합니다. 그런 다음 템플릿은 SNS 주제에 Lambda 함수를 호출할 권한을 추가합니다.

```
FlexMatchEventTopic:
  Type: "AWS::SNS::Topic"
  Properties:
    KmsMasterKeyId: alias/aws/sns #Enables server-side encryption on the topic using an
 AWS managed key
    Subscription:
      - Endpoint: !GetAtt FlexMatchEventHandlerLambdaFunction.Arn
        Protocol: lambda
    TopicName: MyFlexMatchEventTopic

FlexMatchEventTopicPolicy:
  Type: "AWS::SNS::TopicPolicy"
  DependsOn: FlexMatchEventTopic
  Properties:
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service: gamelift.amazonaws.com
          Action:
            - "sns:Publish"
          Resource: !Ref FlexMatchEventTopic
    Topics:
      - Ref: FlexMatchEventTopic

FlexMatchEventHandlerLambdaPermission:
  Type: "AWS::Lambda::Permission"
  Properties:
    Action: "lambda:InvokeFunction"
    FunctionName: !Ref FlexMatchEventHandlerLambdaFunction
    Principal: sns.amazonaws.com
    SourceArn: !Ref FlexMatchEventTopic
```

FlexMatch를 위한 게임 준비

Amazon GameLift FlexMatch를 사용하여 게임에 플레이어 매치메이킹 기능을 추가할 수 있습니다. FlexMatch는 사용자 지정 게임 서버 및 Realtime 서버에 대한 관리형 Amazon GameLift 솔루션과 함께 사용할 수 있습니다.

FlexMatch는 매치메이킹 서비스를 사용자 지정 가능한 규칙 엔진과 연결합니다. 그렇게 하면 플레이어 속성 및 게임 모드를 바탕으로 게임에 적합하게 플레이어들을 서로 매칭하고 FlexMatch를 사용하여 플레이어 그룹 형성의 기본 구조와 플레이어의 게임 내 배치를 관리하는 방식을 설계할 수 있습니다. 사용자 지정 매치메이킹에 대한 자세한 내용은 FlexMatch 규칙 세트 예제 단원을 참조하십시오.

FlexMatch는 대기열 기능을 바탕으로 빌드됩니다. 매치가 형성되면 FlexMatch가 매치 세부 정보를 선택한 대기열에 전달합니다. 대기열은 Amazon GameLift 플릿에서 사용할 수 있는 호스팅 리소스를 검색하여 매치를 위한 새로운 게임 세션을 시작합니다.

이 단원의 주제에서는 게임 서버 및 게임 클라이언트에 매치메이킹 지원을 추가하는 방법을 설명합니다. 게임에 대한 매치메이커를 생성하려면 Amazon GameLift FlexMatch 매치메이커 구축 단원을 참조하십시오. FlexMatch 작동 방식에 대한 자세한 내용은 Amazon GameLift FlexMatch 작동 방식 섹션을 참조하세요.

FlexMatch를 게임 클라이언트에 추가

이 주제에서는 클라이언트 측 게임 서비스에 FlexMatch 매치메이킹 지원을 추가하는 방법에 대해 설명합니다. FlexMatch를 Amazon GameLift 관리형 호스팅과 함께 사용하든 다른 호스팅 솔루션에서 사용하든 프로세스는 기본적으로 동일합니다. FlexMatch 및 게임에 대해 사용자 지정 매치메이커를 설정하는 방법에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- Amazon GameLift 호스팅을 사용한 FlexMatch 통합
- Amazon GameLift FlexMatch 작동 방식
- Amazon GameLift FlexMatch 매치메이커 구축
- FlexMatch 규칙 세트 예제

게임에 대해 FlexMatch 매치메이킹을 활성화하려면 다음 기능을 추가합니다.

- 한 명 또는 여러 명의 플레이어에 대한 매치메이킹 요청 준비(필수).
- 매치메이킹 요청 상태 추적(필수).
- 제안된 매치에 대한 플레이어 수락 요청(선택 사항).

- 새로운 매치에 대한 게임 세션이 생성되면 플레이어 연결 정보를 얻고 게임에 참여합니다.

플레이어에 대해 매치메이킹 요청 준비

게임 클라이언트에서 클라이언트 측 게임 서비스를 통해 매치메이킹 요청을 수행하는 것이 좋습니다. 신뢰할 수 있는 소스를 사용하면 해킹 시도 및 가짜 플레이어 데이터를 보다 쉽게 막을 수 있습니다. 게임에 세션 디렉터리 서비스가 있는 경우 매치메이킹 요청을 처리하기 위한 좋은 옵션이 됩니다.

클라이언트 서비스를 준비하려면 다음 작업을 수행합니다.

- Amazon GameLift API를 추가합니다. 클라이언트 서비스는 AWS SDK에 속한 Amazon GameLift API의 기능을 사용합니다. 클라이언트 서비스용 Amazon GameLift SDK 섹션을 참조하여 AWS SDK에 대해 자세히 알아보고 최신 버전을 다운로드하세요. 이 SDK를 게임 클라이언트 서비스 프로젝트에 추가합니다.
- 매치메이킹 티켓 시스템을 설정합니다. 모든 매치메이킹 요청에 고유 티켓 ID를 할당해야 합니다. 고유 ID를 생성한 후 새 매치 요청에 할당하는 메커니즘이 필요합니다. 티켓 ID는 최대 128자까지 어떤 문자열 형식도 사용 가능합니다.
- 매치메이커 정보를 가져옵니다. 사용할 계획인 매치메이킹 구성 이름을 가져옵니다. 또한 매치메이커의 규칙 세트에 정의된 필수 플레이어 속성에 대한 매치메이커의 목록이 필요합니다.
- 플레이어 데이터를 가져옵니다. 각 플레이어에 대한 관련 데이터를 가져오는 방법을 설정합니다. 여기에는 플레이어가 게임에 배정될 가능성이 있는 각 리전에 대한 플레이어 ID, 속성 값 및 업데이트된 지연 시간 데이터가 포함됩니다.
- (선택 사항) 매치 채우기를 활성화합니다. 기존 매칭된 게임을 채울 방법을 결정합니다. 매치메이커의 채우기 모드가 “수동”으로 설정된 경우 게임에 채우기 지원을 추가할 수 있습니다. 채우기 모드가 “자동”으로 설정된 경우 개별 게임 세션에 대해 채우기 모드를 해제하는 방법이 필요할 수 있습니다. FlexMatch로 기존 게임 채우기에서 매치 채우기를 관리하는 방법에 대해 자세히 알아보세요.

플레이어에 대해 매치메이킹 요청

FlexMatch 매치메이커에 대한 매치메이킹 요청을 생성하고 관리하기 위해 클라이언트 서비스에 코드를 추가합니다. FlexMatch 매치메이킹을 요청하는 프로세스는 FlexMatch를 Amazon GameLift 관리형 호스팅과 함께 사용하는 게임과 FlexMatch를 독립형 솔루션으로 사용하는 게임의 경우 동일합니다.

매치메이킹 요청 생성:

- Amazon GameLift API StartMatchmaking을 호출합니다. 각 요청에는 다음 정보가 포함되어야 합니다.

매치메이커

요청에 사용할 매치메이킹 구성의 이름입니다. FlexMatch가 각 요청을 지정된 매치메이커에 대한 풀에 배치하며 요청은 매치메이커의 구성 방법에 따라 처리됩니다. 여기에는 시간 제한 적용, 매치에 대한 플레이어 수락 요청 여부, 결과 게임 세션을 배치할 때 사용할 대기열 등이 포함됩니다. FlexMatch 매치메이커 디자인에서 매치메이커 및 규칙 세트에 대해 자세히 알아보세요.

티켓 ID

요청에 할당된 고유 티켓 ID입니다. 이벤트 및 알림 등 요청과 관련된 모든 항목이 티켓 ID를 참조합니다.

플레이어 데이터

생성할 매치의 대상인 플레이어 목록입니다. 요청에 속한 플레이어 중 한 명이라도 매치 규칙 및 최소 지연 시간에 따라 매치 요구 사항을 충족하지 않는 경우, 매치메이킹을 요청해도 성공적으로 매칭되지 않습니다. 매치 요청에 최대 10명의 플레이어를 포함시킬 수 있습니다. 요청에 여러 플레이어가 속해 있는 경우, FlexMatch는 단일 매치를 생성하여 모든 플레이어를 동일한 팀에 할당하려고 시도합니다(무작위 선택). 매치 팀 중 하나에 맞추기에는 요청에 플레이어가 너무 많이 포함되어 있는 경우, 요청이 매칭되지 않습니다. 예를 들어 2v2 매치(2명의 플레이어가 있는 2개의 팀)를 생성하도록 매치메이커를 설정하는 경우, 플레이어가 셋 이상 포함된 매치메이킹 요청을 전송할 수 없습니다.

> **Note**
>
> 한 플레이어(플레이어 ID로 식별)는 한 번에 하나의 액티브 매치메이킹 요청에만 포함될 수 있습니다. 플레이어에 대해 새 요청을 생성할 때 동일한 플레이어 ID와 연결된 모든 활성 매치메이킹 티켓이 자동으로 취소됩니다.

나열된 각 플레이어에 대해 다음 데이터를 포함합니다.

- 플레이어 ID - 각 플레이어는 사용자가 생성한 고유 플레이어 ID를 가지고 있어야 합니다. 플레이어 ID 생성을 참조하세요.
- 플레이어 속성 - 사용 중인 매치메이커가 플레이어 속성을 요청하는 경우 해당 요청은 각 플레이어에게 해당 속성을 제공해야 합니다. 필수 플레이어 속성은 매치메이커의 규칙 세트에 정의되어 있으며, 이 규칙 세트는 속성에 대한 데이터 형식도 지정합니다. 플레이어 속성은 규칙 세트가 해당 속성에 대해 기본값을 지정할 때에만 선택 사항입니다. 매치 요청이 모든

플레이어에게 필수 플레이어 속성을 제공하지 않는 경우 매치메이킹 요청이 성공할 수 없습니다. FlexMatch 규칙 세트 설계 및 FlexMatch 규칙 세트 예제에서 매치메이커 규칙 세트 및 플레이어 속성에 대해 자세히 알아보십시오.

- 플레이어 지연 시간 - 사용 중인 매치메이커에 플레이어 지연 시간 규칙이 있는 경우 요청은 각 플레이어에 대한 지연 시간을 보고해야 합니다. 플레이어 지연 시간 데이터는 플레이어당 하나 이상의 값 목록입니다. 이 데이터는 매치메이커 대기열에서 플레이어가 리전에 대해 경험하는 지연 시간을 나타냅니다. 요청에 플레이어에 대한 지연 시간 값이 포함되어 있지 않는 경우 플레이어가 매칭될 수 없으며 요청이 실패합니다.

매치 요청 세부 정보 검색:

- 매치 요청이 전송되면 요청의 티켓 ID로 DescribeMatchmaking을 호출하여 요청 세부 정보를 볼 수 있습니다. 이 요청은 현재 상태를 포함한 요청 정보를 반환합니다. 요청이 성공적으로 완료되면 티켓에도 게임 클라이언트가 매치에 연결하는 데 필요한 정보가 포함됩니다.

매치 요청 취소:

- 요청의 티켓 ID로 StopMatchmaking을 호출하여 언제든지 매치메이킹 요청을 취소할 수 있습니다.

매치메이킹 이벤트 추적

매치메이킹 프로세스에 대해 Amazon GameLift가 방출하는 이벤트를 추적하도록 알림을 설정합니다. 직접 알림을 설정하거나 SNS 주제를 생성하거나 Amazon EventBridge를 사용하여 알림을 설정할 수 있습니다. 알림 설정에 대한 자세한 내용은 FlexMatch 이벤트 알림 설정 섹션을 참조하세요. 알림을 설정하면 이벤트를 감지하고 필요할 경우 응답하기 위해 클라이언트 서비스에 리스너를 추가합니다.

또한 알림 없이 상당한 시간이 경과하면 상태 업데이트를 정기적으로 폴링하여 알림을 백업하는 것이 좋습니다. 매치메이킹 성능에 미치는 영향을 최소화하려면 매치메이킹 티켓을 제출한 후 30초 이상 기다린 후 또는 마지막으로 알림을 받은 후에만 폴링해야 합니다.

요청의 티켓 ID로 DescribeMatchmaking을 호출하여 현재 상태를 포함한 매치메이킹 요청 티켓을 검색합니다. 폴링은 10초에 1회가 넘지 않도록 하는 것이 좋습니다. 이 접근 방식은 소량 개발 시나리오에서만 사용할 수 있습니다.

> **Note**
>
> 프로덕션 전 부하 테스트와 같이 대량의 매치메이킹을 사용하기 전에 이벤트 알림을 사용하여 게임을 설정해야 합니다. 공개 버전의 모든 게임은 볼륨에 관계없이 알림을 사용해야 합니다. 지속적인 폴링 접근 방식은 매치메이킹 사용량이 낮은 개발 중인 게임에만 적합합니다.

플레이어 수락 요청

플레이어 수락이 설정되어 있는 매치메이커를 사용 중인 경우 플레이어 수락 프로세스를 관리하기 위해 코드를 클라이언트 서비스에 추가합니다. 플레이어 수락을 관리하는 프로세스는 FlexMatch를 Amazon GameLift 관리형 호스팅과 함께 사용하는 게임과 FlexMatch를 독립형 솔루션으로 사용하는 게임의 경우 동일합니다.

제안된 매치에 대한 플레이어 수락 요청:

1. 제안된 매치에 플레이어 수락이 필요할 때를 감지합니다. 매치메이킹 티켓을 모니터링하여 상태가 `REQUIRES_ACCEPTANCE`로 변경되는 때를 감지합니다. 이 상태가 변경되면 `MatchmakingRequiresAcceptance` FlexMatch 이벤트가 트리거됩니다.
2. 모든 플레이어로부터 수락을 취득합니다. 매치메이킹 티켓의 모든 플레이어에게 제안된 매치 세부 정보를 제공하는 메커니즘을 생성합니다. 플레이어는 제안된 매치를 수락할지 아니면 거부할지를 나타낼 수 있어야 합니다. DescribeMatchmaking 호출을 통해 매치 세부 정보를 검색할 수 있습니다. 매치메이커가 제안된 매치를 철회하고 나가기 전에 플레이어가 응답할 수 있는 시간에는 제한이 있습니다.
3. FlexMatch에 대한 플레이어 반응을 보고합니다. 수락 또는 거부로 AcceptMatch를 호출하여 플레이어 응답을 보고합니다. 매치가 성사되려면 매치메이킹 요청에 포함된 모든 플레이어가 매치를 수락해야 합니다.
4. 수락이 안 된 티켓을 처리합니다. 제안된 매치의 플레이어가 매치를 거부하거나 수락 시간 제한 내에 응답하지 못하면 요청이 실패합니다. 매치를 수락한 플레이어의 티켓은 자동으로 티켓 풀로 반환됩니다. 매치를 수락하지 않은 플레이어의 티켓은 FAILURE 상태로 바뀌고 더 이상 처리되지 않습니다. 플레이어가 여러 명인 티켓의 경우 티켓에 있는 플레이어가 매치를 수락하지 않으면 티켓 전체가 실패합니다.

매치 연결

성공적으로 구성된 매치(상태 COMPLETED 또는 이벤트 MatchmakingSucceeded)를 처리하기 위한 코드를 클라이언트 서비스에 추가합니다. 여기에는 매치의 플레이어에게 알리고 해당 게임 클라이언트에게 연결 정보를 전달하는 작업이 포함됩니다.

Amazon GameLift 관리형 호스팅을 사용하는 게임의 경우 매치메이킹 요청이 성공적으로 처리되면 게임 세션 연결 정보가 매치메이킹 티켓에 추가됩니다. DescribeMatchmaking을 호출하여 완료된 매치메이킹 티켓을 검색합니다. 연결 정보에는 게임 세션의 IP 주소 및 포트는 물론 각 플레이어 ID에 대한 플레이어 세션 ID도 포함됩니다. GameSessionConnectionInfo에서 자세히 알아보세요. 게임 클라이언트는 이 정보를 사용하여 매치의 게임 세션에 직접 연결합니다. 연결 요청에는 플레이어 세션 ID와 플레이어 ID가 포함되어야 합니다. 이 데이터는 연결된 플레이어를 팀 할당을 포함한 게임 세션의 매치 데이터에 연결합니다(GameSession 참조).

Amazon GameLift FleetIQ를 비롯한 다른 호스팅 솔루션을 사용하는 게임의 경우 매치 플레이어가 적절한 게임 세션에 연결할 수 있는 메커니즘을 구축해야 합니다.

샘플 매치메이킹 요청

다음 코드 조각은 서로 다른 여러 매치메이커에 대한 매치메이킹 요청을 빌드합니다. 설명에 따라 요청은 매치메이커의 규칙 세트에 정의된 대로 사용 중인 매치메이커가 필요로 하는 플레이어 속성을 제공해야 합니다. 제공된 속성은 규칙 세트에 정의된 동일한 데이터 형식인 숫자(N) 또는 문자열(S)을 사용해야 합니다.

```
# Uses matchmaker for two-team game mode based on player skill level
def start_matchmaking_for_cowboys_vs_aliens(config_name, ticket_id, player_id, skill,
 team):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill}
            },
            "PlayerId": player_id,
            "Team": team
        }],
        TicketId=ticket_id)

# Uses matchmaker for monster hunter game mode based on player skill level
def start_matchmaking_for_players_vs_monster(config_name, ticket_id, player_id, skill,
 is_monster):
```

```
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill},
                "desiredSkillOfMonster": {"N": skill},
                "wantsToBeMonster": {"N": int(is_monster)}
            },
            "PlayerId": player_id
        }],
        TicketId=ticket_id)

# Uses matchmaker for brawler game mode with latency
def start_matchmaking_for_three_team_brawler(config_name, ticket_id, player_id, skill,
 role):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill},
                "character": {"S": [role]},
            },
            "PlayerId": player_id,
            "LatencyInMs": { "us-west-2": 20}
        }],
        TicketId=ticket_id)

# Uses matchmaker for multiple game modes and maps based on player experience
def start_matchmaking_for_multi_map(config_name, ticket_id, player_id, skill, maps,
 modes):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "experience": {"N": skill},
                "gameMode": {"SL": modes},
                "mapPreference": {"SL": maps}
            },
            "PlayerId": player_id
        }],
        TicketId=ticket_id)
```

FlexMatch를 Amazon GameLift 호스팅 게임 서버에 추가

이 주제에서는 Amazon GameLift 관리형 호스팅을 사용하는 사용자 지정 게임 서버에 FlexMatch 매치메이킹 지원을 추가하는 방법에 대해 설명합니다. 게임에 FlexMatch를 추가하는 방법에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- Amazon GameLift FlexMatch 작동 방식
- Amazon GameLift 호스팅을 사용한 FlexMatch 통합

이 주제의 정보는 Amazon GameLift를 게임 서버에 추가에 설명된 바와 같이 Amazon GameLift Server SDK를 게임 서버 프로젝트에 성공적으로 통합했다고 가정합니다. 이 작업이 완료되면 사용자에게 필요한 대부분의 메커니즘을 갖게 됩니다. 이 주제의 섹션에서는 FlexMatch로 설정된 게임을 처리하는 데 필요한 나머지 작업에 대해 설명합니다.

매치메이킹을 위한 게임 서버 설정

게임 서버를 설정하여 매치 게임을 처리하려면 다음 작업을 완료합니다.

1. 매치메이킹으로 생성한 게임 세션을 시작합니다. 새 게임 세션을 요청하기 위해 Amazon GameLift는 게임 세션 객체를 사용하여 게임 서버에 `onStartGameSession()` 요청을 전송합니다(GameSession 참조). 게임 서버는 사용자 지정 게임 데이터를 비롯한 게임 세션 정보를 사용하여 요청된 게임 세션을 시작합니다. 자세한 내용은 게임 세션 시작 섹션을 참조하세요.

   매칭된 게임의 경우 게임 세션 객체에는 매치메이커 데이터 세트도 포함됩니다. 매치메이커 데이터에는 게임 서버가 매치를 위해 새 게임 세션을 시작해야 한다는 정보가 포함됩니다. 여기에는 매치의 팀 구조, 팀 배정 및 게임에 적합할 수 있는 특정 플레이어 속성이 포함됩니다. 예를 들어 게임이 평균 플레이어 스킬 레벨을 기반으로 특정 기능 또는 수준을 잠금 해제하거나 플레이어의 선호에 따라 맵을 선택할 수도 있습니다. 매치메이커 데이터를 사용하는 작업에서 자세히 알아보세요.
2. 플레이어 연결 처리. 매칭된 게임에 연결할 때 게임 클라이언트는 플레이어 ID 및 플레이어 세션 ID를 참조합니다(새 플레이어 확인 참조). 게임 서버는 플레이어 ID를 사용하여 들어오는 플레이어를 매치메이커 데이터의 플레이어 정보과 연결합니다. 매치메이커 데이터는 플레이어의 팀 배정을 식별하며 게임에서 플레이어를 올바르게 나타내는 데 필요한 기타 정보를 제공할 수 있습니다.
3. 플레이어가 게임을 중단할 때 보고합니다. 게임 서버가 Server API `RemovePlayerSession()`을 호출하여 중단된 플레이어를 보고하는지 확인합니다(플레이어 세션 종료 보고 참조). 이 단계는 FlexMatch 채우기를 사용하여 기존 게임에서 빈 슬롯을 채우는 경우 중요합니다. 게임이 클라이언트 측 게임 서비스를 통해 채우기 요청을 시작하는 경우 반드시 필요합니다. FlexMatch로 기존 게임 채우기에서 FlexMatch 채우기를 구현하는 방법에 대해 자세히 알아보세요.

4. 기존 매치 게임 세션에 대해 새 플레이어를 요청합니다(선택 사항). 기존 매칭된 게임을 채울 방법을 결정합니다. 매치메이커의 채우기 모드가 “수동”으로 설정된 경우 게임에 채우기 지원을 추가할 수 있습니다. 채우기 모드가 “자동”으로 설정된 경우 개별 게임 세션에 대해 채우기 모드를 해제하는 방법이 필요할 수 있습니다. 예를 들어 게임의 특정 지점에 도달하면 게임 세션 채우기를 중지하고 싶을 수도 있습니다. FlexMatch로 기존 게임 채우기에서 매치 채우기를 관리하는 방법에 대해 자세히 알아보세요.

매치메이커 데이터를 사용하는 작업

게임 서버는 GameSession 객체의 게임 정보를 인식하고 이를 사용할 수 있어야 합니다. Amazon GameLift 서비스는 게임 세션이 시작되거나 업데이트될 때마다 이러한 객체를 게임 서버에 전달합니다. 핵심 게임 세션 정보에는 게임 세션 ID 및 이름, 최대 플레이어 수, 연결 정보 및 사용자 지정 게임 데이터(제공되는 경우)가 포함됩니다.

FlexMatch를 사용하여 생성된 게임 세션의 경우 `GameSession` 객체에는 매치메이커 데이터 세트도 포함됩니다. 이는 고유한 매치 ID와 더불어 매치를 생성한 매치메이커를 식별하고 팀, 팀 배정 및 플레이어를 설명합니다. 원본 매치메이킹 요청의 플레이어 속성도 포함합니다(플레이어 객체 참조). 플레이어 지연 시간은 포함하지 않습니다. 매치 채우기에 필요한 경우처럼 현재 플레이어의 지연 시간 데이터가 필요한 경우 새로운 데이터를 가져오는 것이 좋습니다.

> **Note**
>
> 매치메이커 데이터는 전체 매치메이킹 구성 ARN을 지정하며, 이 ARN은 구성 이름, AWS 계정, 리전을 식별합니다. 게임 클라이언트 또는 서비스에서 매치 채우기를 요청할 때 구성 이름만 필요합니다. ":matchmakingconfiguration/" 다음에 오는 문자열을 구문 분석하여 구성 이름을 추출할 수 있습니다. 표시된 예에서 매치메이킹 구성 이름은 "MyMatchmakerConfig"입니다.

다음 JSON은 매치메이커 데이터의 일반적인 집합을 보여 줍니다. 이 예는 두 명의 플레이어를 대상으로 하는 게임을 설명하며, 여기서 플레이어는 스킬 등급과 획득한 최고 레벨에 따라 매치됩니다. 또한 매치메이커는 캐릭터에 따라 매치되며, 매치된 플레이어는 공통적으로 하나 이상의 맵 기본 설정을 갖습니다. 이 시나리오에서는 게임 서버가 가장 선호되는 맵을 확인하여 게임 세션에서 사용하는 것입니다.

```
{
 "matchId":"1111aaaa-22bb-33cc-44dd-5555eeee66ff",
 "matchmakingConfigurationArn":"arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
```

```
 "teams":[
    {"name":"attacker",
  "players":[
           {"playerId":"4444dddd-55ee-66ff-77aa-8888bbbb99cc",
   "attributes":{
    "skills":{
     "attributeType":"STRING_DOUBLE_MAP",
     "valueAttribute":{"Body":10.0,"Mind":12.0,"Heart":15.0,"Soul":33.0}}
   }
  }]
 },{
  "name":"defender",
  "players":[{
   "playerId":"3333cccc-44dd-55ee-66ff-7777aaaa88bb",
   "attributes":{
    "skills":{
     "attributeType":"STRING_DOUBLE_MAP",
     "valueAttribute":{"Body":11.0,"Mind":12.0,"Heart":11.0,"Soul":40.0}}
   }
  }]
 }]
}
```

FlexMatch로 기존 게임 채우기

매치 채우기는 FlexMatch 메커니즘을 사용하여 기존 매치 게임 세션에 대한 신규 플레이어를 찾습니다. 언제든지 플레이어를 모든 게임에 추가할 수 있지만(게임 세션에 플레이어 참여 참조), 매치 채우기 기능은 새로운 플레이어가 현재 플레이어와 동일한 매치 기준을 충족하도록 보장합니다. 또한 매치 채우기 기능은 새 플레이어를 팀에 할당하고, 플레이어의 수락을 관리하며, 업데이트된 매치 정보를 게임 서버에 전송합니다. FlexMatch 매치메이킹 프로세스에서 매치 채우기 기능에 대해 자세히 알아봅니다.

> **Note**
>
> FlexMatch 채우기는 현재 Realtime 서버를 사용한 게임에서 사용할 수 없습니다.

채우기 메커니즘에는 다음과 같은 두 가지 유형이 있습니다.

- 허용되는 최대 플레이어 수 미만으로 시작하는 게임 세션을 채우려면 자동 채우기를 활성화합니다.
- 진행 중인 게임 세션에서 빠진 플레이어를 교체하려면 채우기 요청을 보내는 기능을 게임 서버에 추가합니다.

자동 채우기 설정

자동 매치 채우기를 사용하면 채워지지 않은 플레이어 슬롯이 하나 이상 있는 채로 게임 세션이 시작할 때마다 Amazon GameLift에서 채우기 요청을 자동으로 트리거합니다. 이 기능을 사용할 경우 최소 매치 플레이어 수가 충족되면 즉시 게임을 시작하고 남은 슬롯은 나중에 추가 플레이어가 매치될 때 채웁니다. 자동 채우기는 언제든지 중단하도록 선택할 수 있습니다.

예를 들어, 6~10명의 플레이어를 보유할 수 있는 게임을 고려해 봅니다. FlexMatch는 처음에 6명의 플레이어를 찾아 매치를 구성하고 새 게임 세션을 시작합니다. 자동 채우기를 사용하면 새 게임 세션에서 4명의 추가 플레이어를 즉시 요청할 수 있습니다. 게임 스타일에 따라 신규 플레이어가 게임 세션 중에 언제든지 참여할 수 있도록 허용할 수 있습니다. 또는 초기 설정 단계 이후 및 게임 플레이 시작 전에 자동 채우기를 중지할 수 있습니다.

게임에 자동 채우기를 추가하려면 게임을 다음과 같이 업데이트합니다.

1. 자동 채우기 활성화. 자동 채우기는 매치메이킹 구성에서 관리됩니다. 활성화되면 해당 매치메이커로 만든 모든 매칭 게임 세션에 사용됩니다. Amazon GameLift는 게임 서버에서 게임 세션이 시작되는 즉시 비전체 게임 세션에 대한 채우기 요청을 생성하기 시작합니다.

   자동 채우기를 설정하려면 매치 구성을 열고 채우기 모드를 “자동”으로 설정합니다. 자세한 내용은 매치메이킹 구성 생성 섹션을 참조하세요.
2. 채우기 우선 순위 지정을 활성화합니다. 매치메이킹 프로세스를 사용자 지정하여 새 매치를 생성하기 전에 채우기 요청 처리의 우선 순위를 지정합니다. 매치메이킹 규칙 세트에서 알고리즘 구성 요소를 추가하고 채우기 우선 순위를 “높음”으로 설정합니다. 자세한 내용은 매치 알고리즘 사용자 정의 섹션을 참조하세요.
3. 게임 세션을 새로운 매치 데이터로 업데이트. Amazon GameLift는 Server SDK 콜백 함수 `onUpdateGameSession`을 사용하여 게임 서버에 매치 정보를 업데이트합니다(서버 프로세스 초기화 참조). 업데이트된 게임 세션 객체를 채우기 활동의 결과로 처리하도록 게임 서버에 코드를 추가합니다. 게임 서버의 매치 데이터 업데이트에서 자세히 알아봅니다.
4. 게임 세션에 대한 자동 채우기 해제. 개별 게임 세션 동안 임의의 시점에서 자동 채우기를 중지하도록 선택할 수 있습니다. 자동 채우기를 중지하려면 게임 클라이언트 또는 게임 서버에 코드를 추가하여 Amazon GameLift API 호출 StopMatchmaking을 만듭니다. 이러한 호출의 경우 티켓 ID가 필요합니다. 최신 채우기 요청의 채우기 티켓 ID를 사용하십시오. 이전 단계의 설명에 따라 업데이트된 게임 세션 매치메이킹 데이터에서 이 정보를 가져올 수 있습니다.

채우기 요청 전송(게임 서버에서)

게임 세션을 호스팅하는 게임 서버 프로세스에서 바로 매치 채우기 요청을 시작할 수 있습니다. 서버 프로세스에는 게임에 연결된 현재 플레이어 및 비어 있는 플레이어 슬롯의 상태에 대한 최신 정보가 있습니다.

이 주제에서는 필요한 FlexMatch 구성 요소를 이미 빌드했고 게임 서버 및 클라이언트 측 게임 서비스에 매치메이킹 프로세스를 성공적으로 추가했다고 가정합니다. FlexMatch 설정에 대한 자세한 내용은 Amazon GameLift 호스팅을 사용한 FlexMatch 통합 섹션을 참조하세요.

게임에 대해 매치 채우기를 활성화하려면 다음 기능을 추가합니다.

- 매치메이킹 채우기 요청을 매치메이커에게 전송하고 요청 상태를 추적합니다.
- 게임 세션에 대한 매치 정보를 업데이트합니다. 게임 서버의 매치 데이터 업데이트 섹션을 참조하세요.

다른 서버 기능과 마찬가지로 게임 서버는 Amazon GameLift Server SDK를 사용합니다. 이 SDK는 C++ 및 C#에서 사용할 수 있습니다.

게임 서버에서 매치 채우기를 요청하려면 다음 작업을 완료하십시오.

1. 매치 채우기 요청을 트리거합니다. 일반적으로, 매치 게임에 하나 이상의 빈 플레이어 슬롯이 있을 때마다 채우기 요청을 시작하려고 합니다. 중요한 캐릭터 역할을 수행하거나 팀의 균형을 맞추는 것과 같은 특정 상황에 채우기 요청을 연결하고 싶을 수 있습니다. 또한 게임 세션의 기간에 근거하여 채우기 활동을 제한하려고 할 수도 있습니다.
2. 채우기 요청을 생성합니다. 매치 채우기 요청을 생성하여 FlexMatch 매치메이커에 전송하는 코드를 추가합니다. 채우기 요청은 다음 서버 API를 사용하여 처리됩니다.

   - StartMatchBackfill()
   - StopMatchBackfill()

   채우기 요청을 생성하려면 다음 정보를 이용해 `StartMatchBackfill`을 호출합니다. 채우기 요청을 취소하려면 채우기 요청의 티켓 ID를 이용해 `StopMatchBackfill`을 호출합니다.

   - 티켓 ID - 매치메이킹 티켓 ID를 제공합니다(또는 자동 생성되도록 선택). 동일한 메커니즘을 사용하여 매치메이킹 및 채우기 요청에 티켓 ID를 할당할 수 있습니다. 매치메이킹 및 채우기용 티켓은 동일한 방식으로 처리됩니다.

- 매치메이커 - 채우기 요청에 사용할 매치메이커를 식별합니다. 일반적으로, 원본 매치를 만드는 데 사용된 매치메이커를 사용하려고 할 것입니다. 이 요청에는 매치메이킹 구성 ARN이 필요합니다. 이 정보는 게임 세션 객체(GameSession)에 저장되며, 해당 객체는 게임 세션을 활성화할 때 Amazon GameLift에 의해 서버 프로세스에 제공되었습니다. 매치메이킹 구성 ARN은 `MatchmakerData` 속성에 포함됩니다.
- 게임 세션 ARN - 채우고 있는 게임 세션을 식별합니다. 게임 세션 ARN은 서버 API GetGameSessionId()를 호출하면 얻을 수 있습니다. 매치메이킹 프로세스에서 새 요청 티켓에는 게임 세션 ID가 없는 반면 채우기 요청 티켓에는 게임 세션 ID가 있습니다. 게임 세션 ID의 존재는 새로운 매치 티켓과 채우기 티켓의 차이를 알 수 있는 방법 중 하나입니다.
- 플레이어 데이터 - 채우고 있는 게임 세션의 모든 기존 플레이어에 대한 플레이어 정보(플레이어)를 포함합니다. 이 정보를 통해 매치메이커는 현재 게임 세션에 있는 플레이어에게 가장 적합한 플레이어 매치를 찾을 수 있습니다. 모든 플레이어의 팀 멤버십을 포함해야 합니다. 채우기를 사용하지 않는 경우 팀을 지정하지 마세요. 게임 서버가 플레이어 연결 상태를 정확하게 보고한 경우 다음과 같이 이 데이터를 획득할 수 있어야 합니다.

  1. 게임 세션을 호스팅하는 서버 프로세스에는 현재 게임 세션에 연결되어 있는 플레이어에 대한 최신 정보가 있어야 합니다.
  2. 플레이어 ID, 속성 및 팀 배정을 받으려면 게임 세션 객체(GameSession), `MatchmakerData` 속성에서 플레이어 데이터를 가져옵니다(매치메이커 데이터를 사용하는 작업 참조). 매치메이커 데이터에는 게임 세션과 일치하는 모든 플레이어가 포함되므로 현재 연결된 플레이어에 대해서만 플레이어 데이터를 가져와야 합니다.
  3. 플레이어 지연 시간의 경우, 매치메이커가 지연 시간 데이터를 호출하면 모든 현재 플레이어의 새로운 지연 시간 값을 수집하고 해당 값을 각 `Player` 객체에 포함합니다. 지연 시간 데이터가 생략되고 매치메이커에 지연 시간 규칙이 있는 경우 요청이 일치되지 않습니다. 채우기 요청은 현재 게임 세션이 있는 리전에 대해서만 지연 시간 데이터를 필요로 합니다. `GameSession` 객체의 `GameSessionId` 속성에서 게임 세션의 리전을 얻을 수 있습니다. 이 값은 리전을 포함하는 ARN입니다.

3. 채우기 요청 상태를 추적합니다. Amazon GameLift는 Server SDK 콜백 함수 `onUpdateGameSession`을 사용하여 게임 서버에 매치 정보를 업데이트합니다(서버 프로세스 초기화 참조). 게임 서버의 매치 데이터 업데이트에서 상태 메시지를 처리하는 코드를 추가합니다. 이 코드는 채우기 요청 성공에 따라 업데이트된 게임 세션 객체 또한 처리합니다.

   매치메이커는 한 번에 한 게임 세션의 매치 채우기 요청만 처리할 수 있습니다. 요청을 취소해야 하는 경우 StopMatchBackfill()을 호출합니다. 요청을 변경해야 하는 경우 `StopMatchBackfill`을 호출한 다음 업데이트한 요청을 제출합니다.

채우기 요청 전송(클라이언트 서비스에서)

게임 서버에서 채우기 요청을 보내는 대신에 클라이언트 측 게임 서비스에서 채우기 요청을 보낼 수도 있습니다. 이 옵션을 사용하려면 클라이언트 측 서비스가 게임 세션 활동 및 플레이어 연결에 대한 현재 데이터에 액세스할 수 있어야 합니다. 게임에서 세션 디렉터리 서비스를 사용하는 경우 이 옵션은 적절한 선택일 수 있습니다.

이 주제에서는 필요한 FlexMatch 구성 요소를 이미 빌드했고 게임 서버 및 클라이언트 측 게임 서비스에 매치메이킹 프로세스를 성공적으로 추가했다고 가정합니다. FlexMatch 설정에 대한 자세한 내용은 Amazon GameLift 호스팅을 사용한 FlexMatch 통합 섹션을 참조하세요.

게임에 대해 매치 채우기를 활성화하려면 다음 기능을 추가합니다.

- 매치메이킹 채우기 요청을 매치메이커에게 전송하고 요청 상태를 추적합니다.
- 게임 세션에 대한 매치 정보를 업데이트합니다. 게임 서버의 매치 데이터 업데이트 섹션 참조

다른 클라이언트 기능과 마찬가지로 클라이언트 측 게임 서비스는 Amazon GameLift API를 통해 AWS SDK를 사용합니다. 이 SDK는 C++, C# 및 기타 여러 언어로 제공됩니다. 클라이언트 API의 일반적인 설명은 Amazon GameLift Service API Reference를 참조하십시오. 여기에는 Amazon GameLift 관련 작업에 대한 하위 수준 서비스 API가 설명되어 있으며 언어별 참조 가이드의 링크가 포함되어 있습니다.

클라이언트 측 게임 서비스를 설정하여 매치 게임을 채우려면 다음 작업을 완료합니다.

1. 채우기 요청을 트리거합니다. 일반적으로 게임은 매치 게임에 하나 이상의 빈 플레이어 슬롯이 있을 때마다 채우기 요청을 시작합니다. 중요한 캐릭터 역할을 수행하거나 팀의 균형을 맞추는 것과 같은 특정 상황에 채우기 요청을 연결하고 싶을 수 있습니다. 또한 게임 세션의 기간에 근거하여 채우기 작업을 제한하려고 할 수도 있습니다. 트리거에 어떤 것을 사용하든 최소한 다음 정보가 필요합니다. 이 정보는 게임 세션 ID로 DescribeGameSessions을 호출하여 게임 세션 객체(GameSession)에서 가져올 수 있습니다.

   - 현재 비어 있는 플레이어 슬롯 수. 이 값은 게임 세션의 최대 플레이어 한도와 현재 플레이어 수에서 계산할 수 있습니다. 현재 플레이어 수는 게임 서버가 Amazon GameLift 서비스에 접속해서 새로운 플레이어 연결을 확인하거나 중단된 플레이어를 보고할 때마다 업데이트됩니다.

   - 정책 생성. 이 설정은 게임 세션이 현재 새 플레이어를 수락하는지 여부를 나타냅니다.

게임 세션 객체에는 게임 세션 시작 시간, 사용자 지정 게임 속성 및 매치메이커 데이터를 포함하여 잠재적으로 유용한 기타 정보가 포함됩니다.

2. 채우기 요청을 생성합니다. 매치 채우기 요청을 생성하여 FlexMatch 매치메이커에 전송하는 코드를 추가합니다. 채우기 요청은 다음 클라이언트 API를 사용하여 처리됩니다.

   - StartMatchBackfill
   - StopMatchmaking

   채우기 요청을 생성하려면 다음 정보를 이용해 `StartMatchBackfill`을 호출합니다. 채우기 요청은 매치메이킹 요청(플레이어에 대해 매치메이킹 요청 참조)과 유사하지만 기존 게임 세션을 식별하기도 합니다. 채우기 요청을 취소하려면 채우기 요청의 티켓 ID를 이용해 `StopMatchmaking`을 호출합니다.

   - 티켓 ID - 매치메이킹 티켓 ID를 제공합니다(또는 자동 생성되도록 선택). 동일한 메커니즘을 사용하여 매치메이킹 및 채우기 요청에 티켓 ID를 할당할 수 있습니다. 매치메이킹 및 채우기용 티켓은 동일한 방식으로 처리됩니다.
   - 매치메이커 - 사용할 매치메이킹 구성 이름을 식별합니다. 일반적으로 원본 매치를 만드는 데 사용한 매치메이커를 채우기용으로 사용하려고 할 것입니다. 이 정보는 매치메이킹 구성 ARN 아래 게임 세션 객체(GameSession)인 `MatchmakerData` 속성에 있습니다. 이름 값은 ""matchmakingconfiguration/" 다음의 문자열입니다. (예를 들어, ARN 값 "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4"에서 매치메이킹 구성 이름은 "MM-4v4"임.)
   - 게임 세션 ARN - 채우고 있는 게임 세션을 지정합니다. 게임 세션 객체의 `GameSessionId` 속성을 사용합니다. 이 ID는 사용자에게 필요한 ARN 값을 사용합니다. 채우기 요청에 대한 매치메이킹 티켓(MatchmakingTicket)은 처리되는 동안 게임 세션 ID를 갖습니다. 새로운 매치메이킹 요청 티켓은 매치가 완료될 때까지 게임 세션 ID를 갖지 않습니다. 게임 세션 ID의 존재는 새로운 매치 티켓과 채우기 티켓의 차이를 알 수 있는 방법 중 하나입니다.
   - 플레이어 데이터 - 채우고 있는 게임 세션의 모든 기존 플레이어에 대한 플레이어 정보(플레이어)를 포함합니다. 이 정보를 통해 매치메이커는 현재 게임 세션에 있는 플레이어에게 가장 적합한 플레이어 매치를 찾을 수 있습니다. 모든 플레이어의 팀 멤버십을 포함해야 합니다. 채우기를 사용하지 않는 경우 팀을 지정하지 마세요. 게임 서버가 플레이어 연결 상태를 정확하게 보고한 경우 다음과 같이 이 데이터를 획득할 수 있어야 합니다.

1. 게임 세션 ID를 통해 DescribePlayerSessions()를 호출하여 게임 세션에 현재 연결되어 있는 모든 플레이어를 파악합니다. 각 플레이어 세션에는 플레이어 ID가 포함됩니다. 상태 필터를 추가하여 활성 플레이어 세션만 검색할 수 있습니다.
2. 게임 세션 객체(GameSession), `MatchmakerData` 속성에서 플레이어 데이터를 가져옵니다(매치메이커 데이터를 사용하는 작업 참조). 이전 단계에서 획득한 플레이어 ID를 사용하여 현재 연결되어 있는 플레이어의 데이터만 가져옵니다. 플레이어가 삭제되면 매치메이커 데이터가 업데이트되지 않으므로 현재 플레이어의 데이터만 추출해야 합니다.
3. 플레이어 지연 시간의 경우 매치메이커가 지연 시간 데이터를 호출하면 모든 현재 플레이어의 새로운 지연 시간 값을 수집하고 해당 값을 `Player` 객체에 포함합니다. 지연 시간 데이터가 생략되고 매치메이커에 지연 시간 규칙이 있는 경우 요청이 일치되지 않습니다. 채우기 요청은 현재 게임 세션이 있는 리전에 대해서만 지연 시간 데이터를 필요로 합니다. `GameSession` 객체의 `GameSessionId` 속성에서 게임 세션의 리전을 얻을 수 있습니다. 이 값은 리전을 포함하는 ARN입니다.

3. 채우기 요청의 상태를 추적합니다. 매치메이킹 티켓 상태 업데이트를 수신하기 위한 코드를 추가합니다. 이 메커니즘 설정을 이용하면 이벤트 알림(기본) 또는 폴링을 통해 새로운 매치메이킹 요청 티켓을 추적할 수 있습니다(매치메이킹 이벤트 추적 참조). 채우기 요청으로 플레이어 수락 활동을 트리거할 필요가 없고 게임 서버에서 플레이어 정보가 업데이트되더라도, 계속 티켓 상태를 모니터링하여 요청 실패 및 재제출 작업을 처리해야 합니다.

   매치메이커는 한 번에 한 게임 세션의 매치 채우기 요청만 처리할 수 있습니다. 요청을 취소해야 하는 경우 StopMatchmaking을 호출합니다. 요청을 변경해야 하는 경우 `StopMatchmaking`을 호출한 다음 업데이트한 요청을 제출합니다.

   매치 채우기 요청이 성공하면 게임 서버는 업데이트된 `GameSession` 객체를 수신하고 새 플레이어를 게임 세션에 참여시키는 데 필요한 작업을 처리합니다. 자세한 내용은 게임 서버의 매치 데이터 업데이트에서 확인하십시오.

게임 서버의 매치 데이터 업데이트

게임에서 매치 채우기 요청을 시작하는 방법에 관계없이, 게임 서버는 매치 채우기 요청의 결과로 Amazon GameLift가 제공하는 게임 세션 업데이트를 처리할 수 있어야 합니다.

Amazon GameLift는 성공 여부에 관계없이 매치 채우기 요청을 완료하면 콜백 함수 `onUpdateGameSession`을 사용하여 게임 서버를 호출합니다. 이 호출에는 세 가지 입력 파라미터가 있는데, 매치 채우기 티켓 ID, 상태 메시지, 플레이어 정보를 비롯한 최신 매치메이킹 데이터가 포함된 GameSesson 객체가 있습니다. 게임 서버 통합의 일환으로 게임 서버에 다음 코드를 추가해야 합니다.

1. onUpdateGameSession 함수를 구현합니다. 이 함수는 다음 상태 메시지(updateReason)를 처리할 수 있어야 합니다.
   - MATCHMAKING_DATA_UPDATED - 새로운 플레이어가 성공적으로 게임 세션에 매치되었습니다. GameSession 객체에는 기존 플레이어 및 새로 일치된 플레이어에 대한 플레이어 데이터를 포함하여 업데이트된 매치메이커 데이터가 포함됩니다.
   - BACKFILL_FAILED - 내부 오류로 인해 매치 채우기 시도가 실패했습니다. GameSession 객체는 변경되지 않습니다.
   - BACKFILL_TIMED_OUT - 매치메이커가 제한 시간 내에 채워진 매치를 찾지 못했습니다. GameSession 객체는 변경되지 않습니다.
   - BACKFILL_CANCELLED - 매치 채우기 요청은 StopMatchmaking(클라이언트) 또는 StopMatchBackfill(서버) 호출에 따라 취소되었습니다. GameSession 객체는 변경되지 않습니다.
2. 매치를 성공적으로 채우려면 새 플레이어가 게임 세션에 연결될 때 업데이트된 매치메이커 데이터를 사용하여 새 플레이어를 처리해야 합니다. 최소한 새 플레이어에 대한 팀 배정뿐만 아니라 플레이어가 게임을 시작하는 데 필요한 다른 플레이어 속성을 사용해야 합니다.
3. Server SDK 작업 ProcessReady()에 대한 게임 서버의 호출에서, 프로세스 파라미터로 onUpdateGameSession 콜백 메서드를 추가합니다.

Amazon GameLift FlexMatch 참조

이 섹션에는 Amazon GameLift FlexMatch를 매치메이킹하기 위한 참조 문서가 포함되어 있습니다.

주제

- Amazon GameLift FlexMatch API 참조(AWS SDK)
- FlexMatch 규칙 언어
- FlexMatch 매치메이킹 이벤트

Amazon GameLift FlexMatch API 참조(AWS SDK)

이 주제에서는 Amazon GameLift FlexMatch의 작업 기반 API 작업 목록을 제공합니다. Amazon GameLift FlexMatch Service API는 `aws.gamelift` 네임스페이스의 AWS SDK에 패키징되어 있습니다. AWS를 다운로드하거나 Amazon GameLift API 참조 문서를 확인합니다.

Amazon GameLift FlexMatch는 Amazon GameLift 호스팅 솔루션(사용자 지정 게임 서버 또는 Realtime 서버의 관리형 호스팅, Amazon GameLift FleetIQ를 통한 Amazon EC2 호스팅 포함)으로 호스팅되는 게임은 물론 P2P, 온프레미스 또는 클라우드 컴퓨팅 프리미티브와 같은 다른 호스팅 시스템에서도 사용할 수 있는 매치메이킹 서비스를 제공합니다. Amazon GameLift 호스팅 옵션에 대한 자세한 내용은 Amazon GameLift 개발자 가이드를 참조하세요.

매치메이킹 규칙 및 프로세스 설정

이러한 작업을 호출하여 FlexMatch 매치메이커를 만들고, 게임의 매치메이킹 프로세스를 구성하며, 매치와 팀을 생성하기 위한 일련의 사용자 지정 규칙을 정의합니다.

매치메이킹 구성

- CreateMatchmakingConfiguration - 플레이어 그룹을 평가하고 새 게임 세션에 구축하기 위한 지침을 사용하여 매치메이킹 구성을 만듭니다. 호스팅에 Amazon GameLift를 사용하는 경우 매치에 사용할 새 게임 세션을 생성하는 방법도 지정합니다.
- DescribeMatchmakingConfigurations - Amazon GameLift 리전에 정의된 매치메이킹 구성을 검색합니다.
- UpdateMatchmakingConfiguration - 매치메이킹 구성에 대한 설정을 변경합니다.
- DeleteMatchmakingConfiguration - 리전에서 매치메이킹 구성을 제거합니다.

매치메이킹 규칙 세트

- CreateMatchmakingRuleSet - 플레이어 매치를 검색할 때 사용할 규칙 집합을 만듭니다.
- DescribeMatchmakingRuleSets - Amazon GameLift 리전에 정의된 매치메이킹 규칙 집합을 검색합니다.
- ValidateMatchmakingRuleSet - 매치메이킹 규칙 집합에 대한 구문을 확인합니다.
- DeleteMatchmakingRuleSet - 리전에서 매치메이킹 규칙 집합을 제거합니다.

한 명 또는 여러 명의 플레이어를 위한 매치 요청

게임 클라이언트 서비스에서 이러한 작업을 호출하여 플레이어 매치메이킹 요청을 관리합니다.

- StartMatchmaking - 동일한 매치에서 플레이하고자 하는 플레이어 한 명 또는 그룹에 대해 매치메이킹을 요청합니다.
- DescribeMatchmaking - 상태를 포함하여 매치메이킹 요청에 대한 세부 정보를 가져옵니다.
- AcceptMatch - 플레이어의 수락이 필요한 매치인 경우, 플레이어가 제안된 매치를 수락할 때 Amazon GameLift에 알립니다.
- StopMatchmaking - 매치메이킹 요청을 취소합니다.
- StartMatchBackfill - 기존 게임 세션의 빈 슬롯을 채우기 위해 추가적인 플레이어 매치를 요청합니다.

사용 가능한 프로그래밍 언어

Amazon GameLift를 지원하는 AWS SDK는 다음 언어로 제공됩니다. 개발 환경 지원에 대한 자세한 내용은 각 언어로 된 설명서를 참조하세요.

- C++(SDK docs)(Amazon GameLift)
- Java(SDK docs)(Amazon GameLift)
- .NET(SDK docs)(Amazon GameLift)
- Go(SDK docs)(Amazon GameLift)
- Python(SDK docs)(Amazon GameLift)
- Ruby(SDK docs)(Amazon GameLift)
- PHP(SDK docs)(Amazon GameLift)
- JavaScript/Node.js(SDK docs)(Amazon GameLift)

FlexMatch 규칙 언어

이 섹션의 참조 주제에서는 Amazon GameLift FlexMatch와 함께 사용할 매치메이킹 규칙을 구축하는 데 사용되는 구문과 의미를 설명합니다. 매치메이킹 규칙 및 규칙 세트 작성에 대한 자세한 도움말은 FlexMatch 규칙 세트 설계 섹션을 참조하세요.

주제

- FlexMatch 규칙 세트 스키마
- FlexMatch 규칙 세트 속성 정의
- FlexMatch 규칙 유형
- FlexMatch 속성 표현식

FlexMatch 규칙 세트 스키마

FlexMatch 규칙 세트는 스몰 매치 및 라지 매치 규칙에 표준 스키마를 사용합니다. 각 섹션에 대한 자세한 설명은 FlexMatch 규칙 세트 속성 정의 섹션을 참조하세요.

스몰 매치에 대한 규칙 세트 스키마

다음 스키마는 최대 40명의 플레이어로 구성된 매치를 구성하는 데 사용되는 규칙 세트의 가능한 모든 속성과 허용된 값을 문서화합니다.

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{
         "name": "string,
         "type": <"string", "number", "string_list", "string_number_map">,
         "default": "string"
    }],
    "algorithm": {
        "strategy": "exhaustiveSearch",
        "batchingPreference": <"random", "sorted">,
        "sortByAttributes": [ "string" ],
        "expansionAgeSelection": <"newest", "oldest">,
        "backfillPriority": <"normal", "low", "high">
    },
    "teams": [{
        "name": "string",
        "maxPlayers": number,
```

```
        "minPlayers": number,
        "quantity": integer
    }],
    "rules": [{
        "type": "distance",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "maxDistance": number,
        "minDistance": number,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "comparison",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"<", "<=", "=", "!=", ">", ">=">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "collection",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"intersection", "contains", "reference_intersection_count">,
        "maxCount": number,
        "minCount": number,
        "partyAggregation": <"union", "intersection">
        },{
        "type": "latency",
        "name": "string",
        "description": "string",
        "maxLatency": number,
        "maxDistance": number,
        "distanceReference": number,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "distanceSort",
        "name": "string",
        "description": "string",
        "sortDirection": <"ascending", "descending">,
        "sortAttribute": "string",
```

```
        "mapKey": <"minValue", "maxValue">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "absoluteSort",
        "name": "string",
        "description": "string",
        "sortDirection": <"ascending", "descending">,
        "sortAttribute": "string",
        "mapKey": <"minValue", "maxValue">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "compound",
        "name": "string",
        "description": "string",
        "statement": "string"
        }
    }],
    "expansions": [{
        "target": "string",
        "steps": [{
            "waitTimeSeconds": number,
            "value": number
        }, {
            "waitTimeSeconds": number,
            "value": number
        }]
    }]
}
```

라지 매치에 대한 규칙 세트 스키마

다음 스키마는 40명 초과인 플레이어로 구성된 매치를 구성하는 데 사용되는 규칙 세트의 가능한 모든 속성과 허용된 값을 문서화합니다. 규칙 세트의 모든 팀에 대한 총 `maxPlayers` 값이 40을 초과하면 FlexMatch에서 이 규칙 세트를 사용하는 매치 요청을 라지 매치 지침에 따라 처리합니다.

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{
         "name": "string,
         "type": <"string", "number", "string_list", "string_number_map">,
         "default": "string"
    }],
```

```
    "algorithm": {
        "strategy": "balanced",
        "batchingPreference": <"largestPopulation", "fastestRegion">,
        "balancedAttribute": "string",
        "expansionAgeSelection": <"newest", "oldest">,
        "backfillPriority": <"normal", "low", "high">
    },
    "teams": [{
        "name": "string",
        "maxPlayers": number,
        "minPlayers": number,
        "quantity": integer
    }],
    "rules": [{
        "name": "string",
        "type": "latency",
        "description": "string",
        "maxLatency": number,
        "partyAggregation": <"avg", "min", "max">
    }, {
        "name": "string",
        "type": "batchDistance",
        "batchAttribute": "string",
        "maxDistance": number
    }],
    "expansions": [{
        "target": "string",
        "steps": [{
            "waitTimeSeconds": number,
            "value": number
        }, {
            "waitTimeSeconds": number,
            "value": number
        }]
    }]
}
```

FlexMatch 규칙 세트 속성 정의

이 섹션에서는 규칙 세트 스키마의 각 속성을 정의합니다. 규칙 세트 생성에 대한 추가 도움말은 FlexMatch 규칙 세트 설계 섹션을 참조하세요.

name

규칙 세트를 설명하는 레이블. 이 값은 Amazon GameLift MatchmakingRuleSet 리소스에 할당된 이름과는 관련이 없습니다. 이 값은 완료된 매치를 설명하는 매치메이킹 데이터에 포함되지만 Amazon GameLift 프로세스에서는 사용되지 않습니다.

허용된 값: 문자열

필수? 아니요

ruleLanguageVersion

사용 중인 FlexMatch 속성 표현식 언어의 버전입니다.

허용된 값: “1.0”

필수? 예

playerAttributes

매치메이킹 요청에 포함되고 매치메이킹 프로세스에 사용되는 플레이어 데이터 모음입니다. 또한 매치메이킹 프로세스에 데이터가 사용되지 않더라도, 여기에서 속성을 선언하여 게임 서버로 전달되는 매치메이킹 데이터에 플레이어 데이터가 포함되도록 할 수 있습니다.

필수? 아니요

name

매치메이커가 사용하는 플레이어 속성의 고유한 이름입니다. 이 이름은 매치메이킹 요청에서 참조되는 플레이어 속성 이름과 일치해야 합니다.

허용된 값: 문자열

필수? 예

type

플레이어 속성 값의 데이터 형식입니다.

허용되는 값: “string”, “number”, “string_list”, “string_number_map”

필수? 예

default

플레이어에게 매치메이킹 요청이 제공하지 않을 때 사용할 기본값입니다.

허용 값: 플레이어 속성에 허용되는 모든 값.

필수? 아니요

algorithm

매치메이킹 프로세스를 사용자 지정하기 위한 선택적 구성 설정.

필수? 아니요

strategy

매치를 구축할 때 사용할 메서드입니다. 이 속성을 설정하지 않으면 기본 동작인 “exhaustiveSearch”가 됩니다.

허용된 값:

- “exhaustiveSearch” - 표준 매칭 메서드. FlexMatch는 일련의 사용자 지정 매치 규칙을 기반으로 풀에 있는 다른 티켓을 평가하여 가장 오래된 티켓을 중심으로 매치를 일괄적으로 구성합니다. 이 전략은 플레이어 40명 이하의 매치에 사용됩니다. 이 전략을 사용할 때는 `batchingPreference`를 “random” or “sorted”로 설정해야 합니다.
- “balanced” - 라지 매치를 빠르게 형성하도록 최적화된 메서드입니다. 이 전략은 플레이어가 41~200명 정도인 매치에만 사용됩니다. 티켓 풀을 미리 정렬하고, 잠재적 매치를 구성하여, 플레이어를 팀에 배정한 다음, 지정된 플레이어 속성을 사용하여 각 팀의 밸런스를 맞추는 방식으로 매치를 구성합니다. 예를 들어, 이 전략을 사용하여 한 매치에 참가한 모든 팀의 평균 스킬 레벨을 동일하게 맞출 수 있습니다. 이 전략을 사용할 때는 `balancedAttribute`를 설정해야 하며 `batchingPreference`는 “largestPopulation” 또는 “fastestRegion”으로 설정해야 합니다. 대부분의 사용자 지정 규칙 유형은 이 전략에서 인식되지 않습니다.

필수? 예

batchingPreference

매치 구성을 위해 티켓을 그룹화하기 전에 사용할 사전 정렬 메서드입니다. 티켓 풀을 미리 정렬하면 특정 특성에 따라 티켓이 일괄 처리되므로 최종 매치에서 플레이어 간의 균일성이 높아지는 경향이 있습니다.

허용된 값:

- “random” - `strategy` = “exhaustiveSearch”인 경우에만 유효. 사전 정렬은 수행되지 않으며, 풀의 티켓은 무작위로 일괄 처리됩니다. 이는 전체 검색 전략의 기본 동작입니다.
- “sorted” - `strategy` = “exhaustiveSearch”인 경우에만 유효. 티켓 풀은 `sortbyAttributes`에 나열된 플레이어 속성에 따라 사전 정렬됩니다.

- “largestPopulation” - `strategy` = “balanced”인 경우에만 유효. 티켓 풀은 플레이어가 가장 짧은 지연 시간 레벨을 보고한 리전을 기준으로 사전 정렬됩니다. 이는 밸런스 전략의 기본 동작입니다.
- “fastestRegion” - `strategy` = “balanced”인 경우에만 유효. 티켓 풀은 플레이어가 가장 짧은 지연 시간 레벨을 보고한 리전을 기준으로 사전 정렬됩니다. 결과 매치를 완료하는 데 시간이 더 오래 걸리지만 모든 플레이어의 지연 시간은 낮은 편입니다.

필수? 예

`balancedAttribute`

밸런스 전략으로 라지 매치를 구성할 때 사용할 플레이어 속성의 이름입니다.

허용되는 값: `type` = “number”로 `playerAttributes`에 선언된 모든 속성.

필수? `strategy` = “balanced”인 경우, 예.

`sortByAttributes`

일괄 처리 전에 티켓 풀을 사전 정렬할 때 사용할 플레이어 속성 목록입니다. 이 속성은 전체 검색 전략을 사용하여 사전 정렬할 때만 사용됩니다. 속성 목록의 순서에 따라 정렬 순서가 결정됩니다. FlexMatch는 영숫자 값에 대해 표준 정렬 규칙을 사용합니다.

허용되는 값: `playerAttributes`에 선언된 모든 속성.

필수? `batchingPreference` = “sorted”인 경우, 예.

`backfillPriority`

채우기 티켓을 매칭하기 위한 우선 순위 지정 메서드. 이 속성은 FlexMatch가 채우기 티켓을 일괄적으로 처리하는 시기를 결정합니다. 이는 전체 검색 전략을 사용하여 사전 정렬할 때만 사용됩니다. 이 속성을 설정하지 않으면 기본 동작인 “normal”이 됩니다.

허용된 값:

- “normal” - 매치를 구성할 때 티켓의 요청 유형(채우기 또는 새 매치)은 고려되지 않습니다.
- “high” - 티켓 배치가 요청 유형별로 정렬되고 (그런 다음, 연령별로 정렬되며) FlexMatch는 먼저 채우기 티켓을 매칭하려고 시도합니다.
- “low” - 티켓 배치가 요청 유형별로 정렬되고 (그런 다음, 연령별로 정렬되며) FlexMatch는 먼저 비채우기 티켓을 매칭하려고 시도합니다.

필수? 아니요

expansionAgeSelection

매치 규칙 확장을 위한 대기 시간을 계산하는 메서드입니다. 일정 시간이 지난 후에도 매치가 완료되지 않은 경우, 확장은 매치 요건을 완화하는 데 사용됩니다. 대기 시간은 부분적으로 채워진 매치에 이미 있는 티켓의 연령을 기준으로 계산됩니다. 이 속성을 설정하지 않으면 기본 동작인 “newest”가 됩니다.

허용된 값:

- “newest” - 확장 대기 시간은 부분적으로 완료된 매치 중 가장 최근에 생성된 타임스탬프가 있는 티켓을 기준으로 계산됩니다. 신규 티켓 하나가 대기 시간 클록을 다시 시작할 수 있기 때문에 확장은 더 느리게 실행되는 경향이 있습니다.
- “oldest” - 확장 대기 시간은 매치에서 가장 오래 전에 생성된 타임스탬프가 있는 티켓을 기준으로 계산됩니다. 확장은 더 빨리 실행되는 경향이 있습니다.

필수? 아니요

teams

매치 내 팀 구성. 각 팀의 팀 이름과 규모 범위를 제공합니다. 규칙 세트는 적어도 한 팀을 정의해야 합니다.

name

팀의 고유 이름입니다. 규칙 및 확장에서 팀 이름을 참조할 수 있습니다. 매치가 성공하면 매치메이킹 데이터에 있는 팀 이름을 기준으로 플레이어가 배정됩니다.

허용된 값: 문자열

필수? 예

maxPlayers

팀에 할당할 수 있는 최대 플레이어 수입니다.

허용된 값: 숫자

필수? 예

minPlayers

매치가 시작되기 전에 팀에 배정되어야 하는 최소 플레이어 수입니다.

허용된 값: 숫자

필수? 예

`quantity`

매치에서 생성할 수 있는 이 유형의 팀 수입니다. 팀원이 1보다 큰 팀은 번호가 추가되어 지정됩니다(“Red_1”, “Red_2” 등). 이 속성을 설정하지 않을 경우 기본값은 “1”입니다.

허용된 값: 숫자

필수? 아니요

`rules`

매치에 대한 플레이어를 평가하는 방법을 정의하는 규칙 문 모음을 생성합니다.

필수? 아니요

`name`

역할의 고유한 이름입니다. 규칙 세트의 모든 규칙은 고유한 이름을 가져야 합니다. 규칙 이름은 규칙과 관련된 활동을 추적하는 이벤트 로그 및 측정치에서 참조됩니다.

허용된 값: 문자열

필수? 예

`description`

규칙에 대한 텍스트 설명입니다. 이 정보는 규칙의 용도를 식별하는 데 사용할 수 있습니다. 매치메이킹 프로세스에는 사용되지 않습니다.

허용된 값: 문자열

필수? 아니요

`type`

규칙 문 유형입니다. 각 규칙 유형에는 설정해야 하는 추가 속성이 있습니다. 각 규칙 유형의 구조 및 사용에 대한 자세한 내용은 FlexMatch 규칙 유형 섹션을 참조하세요.

허용된 값:

- “absoluteSort” - 지정된 플레이어 속성이 일괄적으로 가장 오래된 티켓과 비교되는지 여부를 기준으로 티켓을 일괄 정렬하는 명시적인 정렬 메서드를 사용하여 정렬합니다.

- “collection” - 컬렉션의 값을 평가합니다(예: 컬렉션인 플레이어 속성 또는 여러 플레이어의 값 집합).
- “comparison” - 두 값을 비교합니다.
- “compound” - 규칙 세트에 있는 다른 규칙의 논리적 조합을 사용하여 복합 매치메이킹 규칙을 정의합니다. 플레이어가 40명 이하인 매치에만 지원됩니다.
- “distance” - 숫자 값 간의 거리를 측정합니다.
- “batchDistance” - 속성 값 간의 차이를 측정하고 이를 사용하여 매치 요청을 그룹화합니다.
- “distanceSort” - 숫자값이 포함된 지정된 플레이어 속성을 가장 오래된 티켓과 일괄 처리로 비교하는 방법을 기준으로 티켓을 일괄 정렬하는 명시적인 정렬 메서드를 사용하여 정렬합니다.
- “latency” - 매치메이킹 요청에 대해 보고된 리전별 지연 시간 데이터를 평가합니다.

필수? 예

`expansions`

매치를 완료할 수 없는 경우 시간이 지남에 따라 매치 요건을 완화하는 규칙. 매치를 더 쉽게 찾을 수 있도록 점진적으로 적용되는 일련의 단계로 확장을 설정합니다. 기본적으로 FlexMatch는 매치에 추가된 최신 티켓의 연령별로 대기 시간을 계산합니다. `expansionAgeSelection` 알고리즘 속성을 사용하여 확장 대기 시간을 계산하는 방식을 변경할 수 있습니다.

확장 대기 시간은 절대값이므로 각 단계의 대기 시간이 이전 단계보다 더 길어야 합니다. 예를 들어, 점진적인 확장을 예약하려면 30초, 40초, 50초의 대기 시간을 사용할 수 있습니다. 대기 시간은 매치메이킹 구성에 설정된 매치 요청에 허용된 최대 시간을 초과할 수 없습니다.

필수? 아니요

`target`

규칙 세트 요소는 완화되어야 합니다. 팀 규모 속성이나 모든 규칙 문 속성을 완화할 수 있습니다. 구문은 “<component name>[<rule/team name>].<property name>”입니다. 예를 들어, 팀 최소 규모를 변경하려면: `teams[Red, Yellow].minPlayers`. “minSkill”이라는 비교 규칙 문에서 최소 스킬 요구 사항을 변경하려면: `rules[minSkill].referenceValue`.

필수? 예

`steps`

`waitTimeSeconds`

대상 규칙 세트 요소에 새 값을 적용하기 전에 대기하는 시간(초)입니다.

필수? 예

value

대상 규칙 세트 요소의 새 값입니다.

FlexMatch 규칙 유형

배치 거리 규칙

```
batchDistance
```

배치 거리 규칙은 두 속성값 간의 차이를 측정합니다. 라지 및 스몰 매치 항목 모두에 배치 거리 규칙 유형을 사용할 수 있습니다. 다음과 같은 두 가지 유형의 배치 거리 규칙이 있습니다.

- 수치 속성 값을 비교합니다. 예를 들어, 이 유형의 배치 거리 규칙을 통해 매치의 모든 플레이어들이 서로 두 스킬 레벨 내에 있어야 한다고 정할 수 있습니다. 이 유형의 경우 모든 티켓 `batchAttribute` 간의 최대 거리를 정의합니다.
- 문자열 속성 값을 비교합니다. 예를 들어, 이 유형의 배치 거리 규칙을 적용하면 매치에 참여하는 모든 플레이어가 동일한 게임 모드를 요청해야 할 수 있습니다. 이 유형의 경우 FlexMatch가 배치를 구성하는 데 사용하는 `batchAttribute` 값을 정의합니다.

배치 거리 규칙 속성

- **`batchAttribute`** - 배치를 구성하는 데 사용되는 플레이어 속성 값입니다.
- **`maxDistance`** - 성공적인 매치에 대한 최대 거리 값입니다. 수치 속성을 비교하는 데 사용됩니다.
- **`partyAggregation`** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 티켓 플레이어의 최소(`min`), 최대(`max`), 평균(`avg`) 값이 포함됩니다. 기본값은 `avg`입니다.

Example

예시

```
{
  "name":"SimilarSkillRatings",
  "description":"All players must have similar skill ratings",
  "type":"batchDistance",
```

```
  "batchAttribute":"SkillRating",
  "maxDistance":"500"
}
```

```
{
  "name":"SameGameMode",
  "description":"All players must have the same game mode",
  "type":"batchDistance",
  "batchAttribute":"GameMode"
}
```

비교 규칙

```
comparison
```

비교 규칙은 또 다른 값과 플레이어 속성 값을 비교합니다. 다음과 같은 두 가지 유형의 비교 규칙이 있습니다.

- 참조 값과 비교합니다. 예를 들어, 이 유형의 비교 규칙을 적용하려면 매칭된 플레이어의 스킬 레벨이 특정 수준 이상이어야 할 수 있습니다. 이 유형의 경우 플레이어 속성, 참조 값, 비교 작업을 지정합니다.
- 플레이어 간을 비교합니다. 예를 들어, 이 유형의 비교 규칙을 적용하려면 매치에 참여하는 모든 플레이어가 서로 다른 캐릭터를 사용해야 할 수 있습니다. 이 유형의 경우 플레이어 속성을 지정하고 equal(=) 또는 not-equal(!=) 비교 연산을 지정합니다. 참조 값을 지정하지 마세요.

> **Note**
>
> 배치 거리 규칙은 플레이어 속성을 비교하는 데 더 효율적입니다. 매치메이킹 지연 시간을 줄이려면 가능하면 배치 거리 규칙을 사용합니다.

비교 규칙 속성

- **`measurements`** - 비교할 플레이어 속성 값입니다.
- **`referenceValue`** - 가능한 매치의 측정값과 비교할 값입니다.
- **`operation`** - 측정값을 참조 값과 비교하는 방법을 결정하는 값입니다. 유효한 연산에는 다음이 포함됩니다. <, <=, =, !=, >, >=.

- **partyAggregation** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 티켓 플레이어의 최소(min), 최대(max), 평균(avg) 값이 포함됩니다. 기본값은 avg입니다.

거리 규칙

```
distance
```

거리 규칙은 플레이어 스킬 레벨 간의 거리와 같이 두 숫자 값 간의 차이를 측정합니다. 예를 들어 거리 규칙에 따라 모든 플레이어가 최소 30시간 이상 게임을 플레이해야 할 수 있습니다.

> **Note**
>
> 배치 거리 규칙은 플레이어 속성을 비교하는 데 더 효율적입니다. 매치메이킹 지연 시간을 줄이려면 가능하면 배치 거리 규칙을 사용합니다.

거리 규칙 속성

- **measurements** - 거리를 측정할 플레이어 속성값입니다. 이 속성은 숫자 값이 있는 속성이어야 합니다.
- **referenceValue** - 가능한 매치에 대한 거리를 측정하는 숫자 값입니다.
- **minDistance/maxDistance** - 성공적인 매치에 대한 최소 또는 최대 거리 값입니다.
- **partyAggregation** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 티켓 플레이어의 최소(min), 최대(max), 평균(avg) 값이 포함됩니다. 기본값은 avg입니다.

수집 규칙

```
collection
```

수집 규칙은 플레이어 속성 값 그룹을 배치에 있는 다른 플레이어의 속성 값 또는 참조 값과 비교합니다. 속성 모음에는 여러 플레이어에 대한 속성 값, 한 플레이어에 대한 속성 값(문자열 목록), 또는 둘 모두가 포함될 수 있습니다. 예를 들어, 수집 규칙은 팀 내 플레이어가 선택한 캐릭터를 살펴볼 수 있습니다. 그러면 규칙에 따라 팀에 특정 캐릭터 중 한 명 이상이 있어야 할 수도 있습니다.

수집 규칙 속성

- **measurements** - 비교할 플레이어 속성 값의 모음입니다. 속성 값은 반드시 문자열 목록이어야 합니다.
- **referenceValue** - 가능한 매치에 대한 측정치를 비교하는 데 사용하는 값 (또는 값 모음)입니다.
- **operation** - 측정값 모음을 비교하는 방법을 결정하는 값입니다. 유효한 작업에는 다음이 포함됩니다.
  - `intersection` - 이 작업을 통해 모든 플레이어의 모음에서 동일한 값의 수를 측정합니다. 교차 작업을 사용하는 규칙의 예는 예제 4: 최선의 매치 검색을 위한 명시적 정렬 사용 섹션을 참조하세요.
  - `contains` - 이 작업을 통해 특정 참조 값을 포함하는 플레이어 속성 모음의 수를 측정합니다. 작업을 사용하는 규칙의 예는 예제 3: 팀 레벨 요구 사항 및 지연 시간 제한 설정 섹션을 참조하세요.
  - `reference_intersection_count` - 이 작업을 통해 참조 값 모음에 항목을 매칭하는 플레이어 속성 모음의 항목 수를 측정합니다. 이 작업을 사용하여 다른 여러 플레이어 속성을 비교할 수 있습니다. 여러 플레이어 속성 모음을 비교하는 규칙의 예는 예제 5: 복수 플레이어 속성 간의 교집합 찾기 섹션을 참조하세요.
- **minCount/maxCount** - 성공적인 매치에 대한 최소 또는 최대 개수 값입니다.
- **partyAggregation** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 이 값의 경우 `union`을 사용하여 그룹에 있는 모든 플레이어의 플레이어 속성을 조합할 수 있습니다. 또는 `intersection`을 사용하여 그룹이 공통으로 가지고 있는 플레이어 속성을 사용할 수도 있습니다. 기본값은 `union`입니다.

복합 규칙

```
compound
```

복합 규칙은 논리적 문을 사용하여 40명 이하의 플레이어로 구성된 매치를 구성합니다. 단일 규칙 세트에 여러 복합 규칙을 사용할 수 있습니다. 여러 복합 규칙을 사용하는 경우 모든 복합 규칙이 true여야 매치를 구성할 수 있습니다.

확장 규칙을 사용하여 복합 규칙을 확장할 수는 없지만 기본 규칙 또는 지원 규칙을 확장할 수 있습니다.

복합 규칙 속성

- **statement** - 개별 규칙을 결합하여 복합 규칙을 구성하는 데 사용되는 로직입니다. 이 속성에서 지정한 규칙은 규칙 세트에서 이전에 정의되어 있어야 합니다. 복합 규칙에는 `batchDistance` 규칙을 사용할 수 없습니다.

이 속성은 다음과 같은 논리 연산자를 지원합니다.

- and - 제공된 두 인수가 true인 경우 표현식은 true입니다.
- or - 제공된 두 인수 중 하나가 true인 경우 표현식은 true입니다.
- not - 표현식에서 인수 결과를 반대로 바꿉니다.
- xor - 인수 중 하나만 true인 경우 표현식은 true입니다.

Example 예

다음 예제에서는 선택한 게임 모드에 따라 다양한 스킬 레벨의 플레이어를 매칭합니다.

```
{
    "name": "CompoundRuleExample",
    "type": "compound",
    "statement": "or(and(SeriousPlayers, VeryCloseSkill), and(CasualPlayers,
 SomewhatCloseSkill))"
}
```

지연 규칙

```
latency
```

지연 규칙은 위치별 플레이어 지연 시간을 측정합니다. 지연 규칙은 지연 시간이 최대값보다 더 높은 모든 위치를 무시합니다. 지연 규칙이 허용되려면 플레이어의 최소 한 위치에서 지연 시간 값이 최대값보다 낮아야 합니다. maxLatency 속성을 지정하여 라지 매치에 이 규칙 유형을 사용할 수 있습니다.

지연 시간 규칙 속성

- **maxLatency** - 위치에 허용 가능한 최대 지연 시간 값입니다. 티켓에 지연 시간이 최대값 미만인 위치가 없는 경우 티켓은 지연 시간 규칙과 일치하지 않습니다.
- **maxDistance** - 각 티켓의 지연 시간과 거리 참조 값 사이의 최대값입니다.
- **distanceReference** - 티켓 지연 시간을 비교할 지연 시간 값입니다. 거리 참조 값의 최대 거리 내에 있는 티켓은 매칭에 성공합니다. 유효한 옵션은 최소(min) 및 평균(avg) 플레이어 지연 시간 값을 포함합니다.
- **partyAggregation** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 티켓 플레이어의 최소(min), 최대(max), 평균(avg) 값이 포함됩니다. 기본값은 avg입니다.

> **Note**
>
> 대기열을 사용하면 지연 시간 규칙과 일치하지 않는 리전에 게임 세션을 배치할 수 있습니다. 대기열의 지연 시간 정책에 대한 자세한 내용은 플레이어 지연 시간 정책 생성을 참조하세요.

절대 정렬 규칙

```
absoluteSort
```

절대 정렬 규칙은 배치에 추가된 첫 번째 티켓과 비교하여 지정된 플레이어 속성을 기준으로 매치메이킹 티켓 배치를 정렬합니다.

절대 정렬 규칙 속성

- **`sortDirection`** - 매치메이킹 티켓을 정렬하는 순서입니다. 유효한 옵션은 다음과 같습니다. `ascending`, `descending`.
- **`sortAttribute`** - 티켓 정렬 기준으로 사용할 플레이어 속성입니다.
- **`mapKey`** - 맵인 경우 플레이어 속성을 정렬하는 옵션입니다. 유효한 옵션은 다음과 같습니다.
  - `minValue` - 값이 가장 낮은 키가 첫 번째입니다.
  - `maxValue` - 값이 가장 높은 키가 첫 번째입니다.
- **`partyAggregation`** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 최소(`min`) 플레이어 속성, 최대(`max`) 플레이어 속성, 그룹 내 플레이어에 대한 모든 플레이어 속성의 평균(`avg`)을 포함합니다. 기본값은 `avg`입니다.

Example

예

다음 예제 규칙은 플레이어를 스킬 레벨별로 정렬하고 그룹의 스킬 레벨의 평균을 계산합니다.

```
{
    "name":"AbsoluteSortExample",
    "type":"absoluteSort",
    "sortDirection":"ascending",
    "sortAttribute":"skill",
    "partyAggregation":"avg"
```

```
}
```

거리 정렬 규칙

```
distanceSort
```

거리 정렬 규칙은 배치에 추가된 첫 번째 티켓에서 지정된 플레이어 속성 거리를 기준으로 매치메이킹 티켓 배치를 정렬합니다.

거리 정렬 규칙 속성

- **`sortDirection`** - 매치메이킹 티켓을 정렬하는 방향입니다. 유효한 옵션은 다음과 같습니다. `ascending`, `descending`.
- **`sortAttribute`** - 티켓 정렬 기준으로 사용할 플레이어 속성입니다.
- **`mapKey`** - 맵인 경우 플레이어 속성을 정렬하는 옵션입니다. 유효한 옵션은 다음과 같습니다.
  - `minValue` - 배치에 추가된 첫 번째 티켓의 경우 값이 가장 낮은 키를 찾습니다.
  - `maxValue` - 배치에 추가된 첫 번째 티켓의 경우 값이 가장 높은 키를 찾습니다.
- **`partyAggregation`** - FlexMatch가 여러 플레이어(그룹)가 있는 티켓을 처리하는 방식을 결정하는 값입니다. 유효한 옵션에는 티켓 플레이어의 최소(`min`), 최대(`max`), 평균(`avg`) 값이 포함됩니다. 기본값은 `avg`입니다.

FlexMatch 속성 표현식

속성 표현식은 매치메이킹과 관련된 특정 속성을 참조하는 데 사용할 수 있습니다. 이를 통해 속성 값을 정의할 때 계산과 로직을 사용할 수 있습니다. 속성 표현식의 결과는 일반적으로 다음 두 가지 형식 중 하나를 취합니다.

- 개별 플레이어 데이터.
- 개별 플레이어 데이터의 계산된 컬렉션.

일반적인 매치메이킹 속성 표현식.

속성 표현식은 플레이어, 팀 또는 매치의 특정 값을 식별합니다. 다음의 부분 표현식이 팀 및 플레이어 식별 방식을 설명합니다.

목표	입력	의미	출력
매치에서 특정 팀을 식별하는 방법:	`teams[red]`	레드 팀	팀
매치에서 특정 팀 집합을 식별하는 방법:	`teams[red,blue]`	Red 팀 및 Blue 팀	List<Team>
매치에서 모든 팀을 식별하는 방법:	`teams[*]`	모든 팀	List<Team>
특정 팀에서 플레이어를 식별하는 방법:	`team[red] .players`	Red 팀에 있는 플레이어	List<Player>
매치의 특정 팀 집합에서 플레이어를 식별하는 방법:	`team[red, blue].players`	매치에 속한 플레이어의 팀별 그룹화	List<List<Player>>
매치에서 플레이어를 식별하는 방법:	`team[*].players`	매치에 속한 플레이어의 팀별 그룹화	List<List<Player>>

속성 표현식 예제

다음 표는 이전 예제를 바탕으로 구축되는 일부 속성 표현식을 나타낸 것입니다.

표현식	의미	결과 유형
`teams[red].players [playerId]`	레드 팀에 속한 모든 플레이어의 플레이어 ID	List<string>
`teams[red].players .attributes[skill]`	레드 팀에 속한 모든 플레이어의 "스킬" 속성	List<number>
`teams[red,blue].players.attributes[skill]`	Red 팀과 Blue 팀에 속한 모든 플레이어의 “skill” 속성(팀별로 그룹화)	List<List<number>>

표현식	의미	결과 유형
`teams[*].players.attributes[skill]`	매치에 속한 모든 플레이어(팀별 그룹화)의 "스킬" 속성	List<List<number>>

속성 표현식 집계

속성 표현식은 다음 함수 또는 함수 조합을 사용하여 팀 데이터를 집계하는 데 사용할 수 있습니다.

집계	입력	의미	출력
`min`	List<number>	목록에 있는 모든 수의 최소값을 구합니다.	숫자
`max`	List<number>	목록에 있는 모든 수의 최대값을 구합니다.	숫자
`avg`	List<number>	목록에 있는 모든 수의 평균을 구합니다.	숫자
`median`	List<number>	목록에 있는 모든 수의 중간값을 구합니다.	숫자
`sum`	List<number>	목록에 있는 모든 수의 합계를 구합니다.	숫자
`count`	List<?>	목록에 있는 원소의 수를 구합니다.	숫자
`stddev`	List<number>	목록에 있는 모든 수의 표준편차를 구합니다.	숫자
`flatten`	List<List<?>>	중첩된 목록의 컬렉션을 모든 원소를 포함하는 단일 목록으로 변환합니다.	List<?>
`set_intersection`	List<List<string>>	모음에 있는 모든 문자열 목록에서 확인된 문	List<string>

집계	입력	의미	출력
		자열의 목록을 가져옵니다.	
모두 해당	List<List<?>>	중첩된 목록 상의 모든 연산이 각 하위 목록에서 개별적으로 이루어져 결과 목록을 산출합니다.	List<?>

다음 표는 집계 함수를 사용하는 일부 유효한 속성 표현식을 나타낸 것입니다.

표현식	의미	결과 유형
flatten(teams[*].players.attributes[skill])	매치에 속한 모든 플레이어(그룹화되지 않음)의 "스킬" 속성	List<number>
avg(teams[red].players.attributes[skill])	레드 팀 플레이어의 평균 스킬	숫자
avg(teams[*].players.attributes[skill])	매치에 속한 각 팀의 평균 스킬	List<number>
avg(flatten(teams[*].players.attributes[skill]))	매치에 속한 모든 플레이어의 평균 스킬 레벨입니다. 이 표현식은 평면화된 플레이어 스킬 목록을 구한 다음 평균합니다.	숫자
count(teams[red].players)	레드 팀에 있는 플레이어의 수	숫자
count (teams[*].players)	매치에 속한 각 팀의 플레이어 수	List<number>

표현식	의미	결과 유형
max(avg(teams[*].players.attributes[skill]))	매치에서 가장 높은 팀 스킬 레벨	숫자

FlexMatch 매치메이킹 이벤트

Amazon GameLift FlexMatch는 매치메이킹 티켓이 처리되는 동안 각 매치메이킹 티켓에 대해 이벤트를 내보냅니다. FlexMatch 이벤트 알림 설정에 설명된 대로 Amazon SNS 주제에 이러한 이벤트를 게시할 수 있습니다. 또한 이러한 이벤트는 Amazon CloudWatch Events에 거의 실시간으로 전송됩니다.

이 주제에서는 FlexMatch 이벤트의 구조를 설명하고 각 이벤트 유형의 예를 제공합니다. 매치메이킹 티켓 상태에 대한 자세한 내용은 Amazon GameLift API 참조의 MatchmakingTicket을 참조하세요.

MatchmakingSearching

티켓이 매치메이킹에 입력되었습니다. 여기에는 신규 요청과 실패한 제안된 매치의 일부였던 요청이 포함됩니다.

리소스: ConfigurationArn

세부 정보: type, tickets, estimatedWaitMillis, gameSessionInfo

예

```
{
  "version": "0",
  "id": "cc3d3ebe-1d90-48f8-b268-c96655b8f013",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:15:36.421Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
```

```
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ],
    "estimatedWaitMillis": "NOT_AVAILABLE",
    "type": "MatchmakingSearching",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

PotentialMatchCreated

잠재적 매치가 생성되었습니다. 이는 수락을 요구하는지 여부에 상관없이 모든 새로운 잠재적 매치에 대해 방출됩니다.

리소스: ConfigurationArn

세부 정보: type, tickets, acceptanceTimeout, acceptanceRequired, ruleEvaluationMetrics, gameSessionInfo, matchId

예

```
{
  "version": "0",
  "id": "fce8633f-aea3-45bc-aeba-99d639cad2d4",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:17:41.178Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
```

```
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T21:17:40.657Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
        ]
      }
    ],
    "acceptanceTimeout": 600,
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
```

```
        "failedCount": 0
      }
    ],
    "acceptanceRequired": true,
    "type": "PotentialMatchCreated",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "3faf26ac-f06e-43e5-8d86-08feff26f692"
  }
}
```

AcceptMatch

플레이어가 잠재적 매치를 수락했습니다. 이 이벤트에는 매치에 속한 각 플레이어의 수락 상태가 표시됩니다. 데이터 누락은 해당 플레이어에 대해 AcceptMatch가 호출되지 않았음을 의미합니다.

리소스: ConfigurationArn

세부 정보: type, tickets, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "b3f76d66-c8e5-416a-aa4c-aa1278153edc",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:04:42.660Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
```

```
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T20:04:16.637Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue",
            "accepted": false
          }
        ]
      }
    ],
    "type": "AcceptMatch",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue",
          "accepted": false
        }
      ]
    },
    "matchId": "848b5f1f-0460-488e-8631-2960934d13e5"
  }
}
```

AcceptMatchCompleted

플레이어 수락, 플레이어 거부 또는 수락 타임아웃으로 매치 수락이 완료됩니다.

리소스: ConfigurationArn

세부 정보: type, tickets, acceptance, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "b1990d3d-f737-4d6c-b150-af5ace8c35d3",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T20:43:14.621Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T20:30:40.972Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T20:33:14.111Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
        ]
      }
```

```
    ],
    "acceptance": "TimedOut",
    "type": "AcceptMatchCompleted",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "a0d9bd24-4695-4f12-876f-ea6386dd6dce"
  }
}
```

MatchmakingSucceeded

매치메이킹이 성공적으로 완료되고 게임 세션이 생성되었습니다.

리소스: ConfigurationArn

세부 정보: type, tickets, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "5ccb6523-0566-412d-b63c-1569e00d023d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T19:59:09.159Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
```

```
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:58:59.277Z",
        "players": [
          {
            "playerId": "player-1",
            "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T19:59:08.663Z",
        "players": [
          {
            "playerId": "player-2",
            "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
            "team": "blue"
          }
        ]
      }
    ],
    "type": "MatchmakingSucceeded",
    "gameSessionInfo": {
      "gameSessionArn": "arn:aws:gamelift:us-west-2:123456789012:gamesession/836cf48d-
bcb0-4a2c-bec1-9c456541352a",
      "ipAddress": "192.168.1.1",
      "port": 10777,
      "players": [
        {
          "playerId": "player-1",
          "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
          "team": "blue"
        }
      ]
    },
    "matchId": "c0ec1a54-7fec-4b55-8583-76d67adb7754"
  }
```

```
}
```

MatchmakingTimedOut

매치메이킹 티켓이 시간 초과로 실패했습니다.

리소스: ConfigurationArn

세부 정보: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "fe528a7d-46ad-4bdc-96cb-b094b5f6bf56",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:11:35.598Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "reason": "TimedOut",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
```

```
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
        "failedCount": 0
      }
    ],
    "type": "MatchmakingTimedOut",
    "message": "Removed from matchmaking due to timing out.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    }
  }
}
```

MatchmakingCancelled

매치메이킹 티켓이 취소되었습니다.

리소스: ConfigurationArn

세부 정보: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "8d6f84da-5e15-4741-8d5c-5ac99091c27f",
```

```
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:00:07.843Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "reason": "Cancelled",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:59:26.118Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 0,
        "failedCount": 0
      }
    ],
```

```
    "type": "MatchmakingCancelled",
    "message": "Cancelled by request.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

MatchmakingFailed

매치메이킹 티켓에 오류가 발생했습니다. 이는 액세스가 불가한 게임 세션 대기열 혹은 내부 오류 때문일 수 있습니다.

리소스: ConfigurationArn

세부 정보: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

예

```
{
  "version": "0",
  "id": "025b55a4-41ac-4cf4-89d1-f2b3c6fd8f9d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-16T18:41:09.970Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-16T18:41:02.631Z",
        "players": [
          {
            "playerId": "player-1",
```

```
            "team": "red"
          }
        ]
      }
    ],
    "customEventData": "foo",
    "type": "MatchmakingFailed",
    "reason": "UNEXPECTED_ERROR",
    "message": "An unexpected error was encountered during match placing.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    },
    "matchId": "3ea83c13-218b-43a3-936e-135cc570cba7"
  }
}
```

FlexMatch를 사용한 보안

AWS에서 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. FlexMatch를 사용할 때 공동 책임 모델을 적용하는 방법에 대한 자세한 내용은 Amazon GameLift의 보안을 참조하세요.

Amazon GameLift FlexMatch 릴리스 노트 및 SDK 버전

Amazon GameLift 릴리스 정보는 서비스와 관련된 새로운 FlexMatch 기능, 업데이트, 수정 사항에 대한 세부 정보를 제공합니다. 이 페이지에는 Amazon GameLift SDK 버전 기록도 포함되어 있습니다.

Amazon GameLift 개발자 리소스

모든 Amazon GameLift 설명서 및 개발자 리소스를 보려면 Amazon GameLift 설명서 홈페이지를 참조하세요.

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.