Amazon GameLift

FlexMatch Guia do desenvolvedor do

Versão

Amazon GameLift: FlexMatch Guia do desenvolvedor do

Table of Contents

O que é o Amazon GameLift FlexMatch?

GameLift FlexMatch O é um serviço de criação de jogos personalizável para jogos multijogador. Com o FlexMatch, você pode criar um conjunto personalizado de regras que define a aparência de uma partida multijogador para o seu jogo e determina como avaliar e selecionar jogadores compatíveis para cada partida. Você também pode personalizar os principais aspectos do processo de matchmaking para se adequar ao seu jogo, incluindo o ajuste fino do algoritmo de correspondência.

O FlexMatch está disponível tanto como um GameLift solução de hospedagem de jogos (incluindo Realtime Servers) e como um serviço autônomo de matchmaking. Você pode implementar FlexMatch como um recurso autônomo com jogos que usam peer-to-peer arquitetura ou hospede servidores de jogos no local ou em outras soluções de computação em nuvem (incluindo GameLift FleetIQ). Este guia fornece informações detalhadas sobre como criar um sistema de matchmaking para qualquer um desses cenários.

O FlexMatch oferece a flexibilidade para definir prioridades de matchmaking dependendo dos requisitos do jogo. Por exemplo, você pode fazer o seguinte:

- Encontre um equilíbrio entre a velocidade e a qualidade da partida. Defina regras de jogo para encontrar rapidamente partidas que sejam boas o suficiente ou peça aos jogadores um pouco mais para encontrar a melhor partida possível para uma ótima experiência do jogador.
- Faça partidas com base em jogadores bem combinados ou equipes bem combinadas. Crie uma partida em que todos os jogadores tenham características semelhantes, como habilidade ou experiência. Alternativamente, formar partidas em que as características combinadas de cada equipe são semelhantes, mesmo que as características dos jogadores individuais sejam mais variadas.
- Priorize como a latência do jogador entra em uma partida. Defina um limite rígido de latência para todos os jogadores em uma partida, certifique-se de que todos em uma partida tenham latência semelhante ou façam as duas coisas.

Pronto para começar a trabalhar com o FlexMatch?

para o step-by-step Orientações sobre como colocar o jogo em funcionamento com o FlexMatch, veja os tópicos a seguir:

- Integração FlexMatch com hospedagem GameLift (p. 8)
- GameLift FlexMatch integração para matchmaking autônomo (p. 7)

Key (Chave) FlexMatch características

Os seguintes recursos estão disponíveis com todos FlexMatch cenários, se você usa o FlexMatch como um serviço autônomo ou com GameLift hospedagem de jogos.

- Marcação de jogador personalizável. Projete e construa casamenteiros para se adequar a todos os modos de jogo que você oferece aos seus jogadores. Crie um conjunto de regras personalizadas para avaliar os principais atributos de jogador (como o nível de habilidade ou a função) e dados de latência geográfica para formar ótimas correspondências de jogador para o seu jogo.

- Marcação baseada em latência. Forneça dados de latência do jogador e crie regras de partida que exigem que os jogadores em uma partida tenham tempos de resposta semelhantes. Esse recurso é útil quando seus pools de matchmaking de jogadores abrangem várias regiões geográficas.
- Support para correspondências de até 200 jogadores. Crie partidas de até 40 jogadores usando regras de partida personalizadas para o seu jogo. Crie partidas de até 200 jogadores usando um processo de correspondência que usa um processo de correspondência personalizado simplificado para manter os tempos de espera do jogador gerenciáveis.
- Aceitação de jogador. Exija que os jogadores optem por participar de uma partida proposta antes de finalizar a partida e iniciar uma sessão de jogo. Use esse recurso para iniciar seu fluxo de trabalho de aceitação personalizado e denunciar respostas do jogador para FlexMatch Antes de colocar uma nova sessão de jogo para a correspondência. Se nem todos os jogadores aceitarem uma partida, a partida proposta falhará e os jogadores que aceitaram retornam automaticamente ao pool de partidas.

  Suporte para grupos de jogadores. Gere correspondências para grupos de jogadores que desejam jogar juntos no mesmo time. Usar o FlexMatch Para encontrar jogadores adicionais para preencher a correspondência conforme o necessário.
- Regras de correspondência expansíveis. Relaxe gradualmente os requisitos de correspondência depois de um determinado período sem encontrar uma correspondência bem-sucedida. A expansão de regras permite que você decida onde e quando relaxar as regras de partida inicial, para que os jogadores possam entrar em jogos jogáveis mais rapidamente.
- Combine a alocação. Preencha os slots de jogador vazios em uma sessão de jogo existente com novos jogadores bem combinados. Personalize quando e como solicitar novos jogadores e use as mesmas regras de correspondência personalizadas para encontrar jogadores adicionais.

FlexMatch com GameLift hospedagem

Para jogos hospedados com o GameLift, FlexMatch A oferece os seguintes recursos adicionais. Estes estão disponíveis ao usar GameLift para hospedar servidores de jogos personalizados ou ao usar Realtime Servers. Jogos hospedados em recursos do Amazon Elastic Compute Cloud (Amazon EC2) com GameLift O FleetIQ deve implementar o FlexMatch como um recurso independente.

- Posicionamento de sessão de jogo Quando uma partida é feita com sucesso, FlexMatch Solicita automaticamente um novo posicionamento de sessão de jogo do GameLift. Os dados gerados durante a criação de partidas, incluindo IDs de jogadores e atribuições de equipe, são fornecidos ao servidor do jogo para que ele possa usar essas informações para iniciar a sessão do jogo para a partida. FlexMatch em seguida, passa de volta as informações de conexão da sessão do jogo para que os clientes do jogo possam participar do jogo. Para minimizar a latência experimentada pelos jogadores em uma partida, a colocação da sessão de jogo com GameLift também pode usar dados de latência de jogadores regionais, se fornecidos.
- Recarga de correspondência automática. Com esse recurso ativado, FlexMatch envia automaticamente uma solicitação de preenchimento de correspondência quando uma nova sessão de jogo começa com slots de jogadores não preenchidos. Seu sistema de matchmaking inicia o processo de posicionamento da sessão do jogo com um número mínimo de jogadores e, em seguida, preenche rapidamente os slots restantes. Você não pode usar o preenchimento automático para substituir jogadores que saem de uma sessão de jogo correspondente.

Definição de preço para GameLift FlexMatch do

O GameLift cobra por instâncias por duração do uso e pela largura de banda por quantidade de dados transferidos. Se você hospedar seus jogos em GameLift servidores, FlexMatch o uso está incluído nas taxas do GameLift. Se você hospedar seus jogos em outra solução de servidor, FlexMatch O uso é cobrado separadamente. Para obter uma lista completa de cobranças e preços do GameLift, consulteAmazônia GameLift Definição de preços.

Para obter informações sobre como calcular o custo de hospedagem de seus jogos ou matchmaking com o GameLift, consulteGerando estimativas de preços do GameLift, que descreve como usar oAWS Pricing Calculator.

Como o Amazon GameLift FlexMatch funciona

Este tópico fornece uma visão geral do serviço GameLift FlexMatch, incluindo os componentes principais de um sistema FlexMatch e como eles interagem.

Você pode usar o FlexMatch com jogos que usam hospedagem gerenciada GameLift ou com jogos que usam outra solução de hospedagem. Jogos hospedados em servidores GameLift, incluindo Realtime Servers, usam o serviço GameLift integrado para localizar automaticamente os servidores de jogos disponíveis e iniciar sessões de jogo para as partidas. Os jogos que usam o FlexMatch como um serviço autônomo, incluindo o GameLift FleetIQ, devem se coordenar com o sistema de hospedagem existente para atribuir recursos de hospedagem e iniciar sessões de jogo para as partidas.

Para obter orientações detalhadas sobre como configurar o FlexMatch para seus jogos, consulteConceitos básicos do FlexMatch (p. 7).

Componentes de marcação de jogos

Um sistema de marcação de jogos do FlexMatch inclui alguns ou todos os seguintes componentes.

Componentes do GameLift

Esses são recursos do GameLift que controlam como o serviço FlexMatch executa o pareamento para o seu jogo. Eles são criados e mantidos usando as ferramentas GameLift, incluindo o console e oAWSCLI ou, alternativamente, usando programaticamente oAWSSDK para GameLift.

- Configuração de matchmaking FlexMatch (também chamada de matchmaker)— Um matchmaker é um conjunto de valores de configuração que personaliza o processo de matchmaking para o seu jogo. Um jogo pode ter vários marcadores de jogos, cada um configurado para diferentes modos de jogo ou experiências conforme necessário. Quando o jogo envia uma solicitação de matchmaking para o FlexMatch, ele especifica qual matchmaker usar.
- Conjunto de regras de criação de jogos do FlexMatch— Um conjunto de regras contém todas as informações necessárias para avaliar jogadores para possíveis partidas e aprovar ou rejeitar. O conjunto de regras define a estrutura de equipe de uma partida, declara os atributos do jogador que são usados para avaliação e fornece regras que descrevem os critérios para uma partida aceitável. As regras podem ser aplicadas a jogadores individuais, a equipes ou a correspondência inteira. Por exemplo, uma regra pode exigir que todos os jogadores da correspondência escolham o mesmo mapa de jogo ou pode exigir que todas as equipes tenham uma média de habilidade de jogador semelhante.
- Fila de sessão de jogos GameLift (somente para FlexMatch com hospedagem gerenciada GameLift)— Uma fila de sessão de jogo localiza os recursos de hospedagem disponíveis e inicia uma nova sessão de jogo para a correspondência. A configuração da fila determina onde o GameLift procura os recursos de hospedagem disponíveis e como selecionar o melhor host disponível para uma correspondência.

Componentes personalizados

Os componentes a seguir abrangem a funcionalidade necessária para um sistema FlexMatch completo que você deve implementar com base na arquitetura do seu jogo.

- Interface do jogador para matchmaking— Esta interface permite que os jogadores participem de uma partida. No mínimo, ele inicia uma solicitação de pareamento por meio do componente de serviço de matchmaking do cliente e fornece dados específicos do jogador, como dados de nível de habilidade e latência, conforme necessário para o processo de matchmaking.

Note

Como prática recomendada, a comunicação com o serviço FlexMatch deve ser feita por um serviço de back-end, não de um cliente de jogo.

- Serviço de marcação de jogos do cliente— Este serviço coloca as solicitações de ingresso do jogador a partir da interface do jogador, gera solicitações de pareamento e as envia para o serviço FlexMatch. Para solicitações em andamento, ele monitora eventos de matchmaking, rastreia o status do pareamento e toma medidas conforme necessário. Dependendo de como você gerencia a hospedagem de sessões de jogo em seu jogo, esse serviço pode retornar informações de conexão de sessão de jogo aos jogadores. Este componente usa oAWSSDK com a API GameLift para se comunicar com o serviço FlexMatch.
- Serviço de posicionamento de correspondência (somente para FlexMatch como um serviço autônomo)— Este componente funciona com seu sistema de hospedagem de jogos existente para localizar recursos de hospedagem disponíveis e iniciar novas sessões de jogos para partidas. O componente deve obter os resultados do pareamento e extrair as informações necessárias para iniciar uma nova sessão de jogo, incluindo IDs de jogador, atributos e atribuições de equipe para todos os jogadores da partida.

Processo de marcar jogos do FlexMatch

Este tópico descreve um cenário básico de pareamento e interações entre os vários componentes do jogo e o serviço FlexMatch.

Solicitar marcação de jogos para jogadores

Um jogador que usa o cliente do jogo clica em um botão “Entrar no jogo”. Essa ação faz com que o serviço de pareamento do cliente envie uma solicitação de pareamento para o FlexMatch. A solicitação identifica o matchmaker FlexMatch a ser usado ao atender a solicitação. A solicitação também inclui informações do jogador que seu matchmaker personalizado exige, como nível de habilidade, preferências de jogo ou dados de latência geográfica. Você pode fazer pedidos de matchmaking para um jogador ou vários jogadores.

Adicionar solicitações ao pool de matchmaking

Quando o FlexMatch recebe a solicitação de pareamento, ele gera um ticket de matchmaking e o adiciona ao pool de tickets do matchmaker. O tíquete permanece no grupo até que ele seja correspondido ou seja atingido um limite máximo de tempo. Seu serviço de matchmaking do cliente é notificado periodicamente sobre eventos de matchmaking, incluindo alterações no status do ticket.

Construa uma partida

Seu matchmaker FlexMatch executa continuamente o seguinte processo em todos os tickets em seu pool:

1. O matchmaker classifica a piscina por idade do ingresso e começa a construir uma partida potencial começando com o ingresso mais antigo.
2. O matchmaker adiciona um segundo ticket à possível correspondência e avalia o resultado em relação às suas regras de matchmaking personalizadas. Se a partida potencial passar na avaliação, os jogadores do ingresso serão atribuídos a uma equipe.
3. O matchmaker adiciona o próximo ticket em sequência e repete o processo de avaliação. Quando todos os slots de jogadores tiverem sido preenchidos, a partida estará pronta.

O pareamento para partidas grandes (41 a 200 jogadores) usa uma versão modificada do processo descrito acima para que ele possa construir partidas em um período de tempo razoável. Em vez de avaliar cada tíquete individualmente, o matchmaker divide um pool de ingressos pré-classificado em partidas potenciais e, em seguida, equilibra cada partida com base em uma característica de jogador que você especificou. Por exemplo, um matchmaker pode pré-classificar os tickets com base em locais de baixa latência semelhantes e, em seguida, usar o balanceamento pós-partida para garantir que as equipes sejam uniformemente correspondidas pela habilidade do jogador.

Relate os resultados do marcação de jogos

Quando uma correspondência aceitável é encontrada, todos os ingressos correspondentes são atualizados e um evento de matchmaking bem-sucedido é gerado para cada ingresso correspondente.

- O FlexMatch como um serviço independente: Seu jogo recebe resultados da partida em um evento de matchmaking bem-sucedido. Os dados do resultado incluem uma lista de todos os jogadores correspondentes e as atribuições de equipe. Se suas solicitações de partida contiverem informações de latência do jogador, os resultados também sugerem uma localização geográfica ideal para a partida.
- FlexMatch com uma solução de hospedagem GameLift: Os resultados da correspondência são passados automaticamente para uma fila do GameLift para posicionamento da sessão do jogo. O matchmaker determina qual fila é usada para posicionamento da sessão do jogo.

Inicie uma sessão de jogo para a partida

Depois que uma partida proposta é formada com sucesso, uma nova sessão de jogo é iniciada. Seus servidores de jogo devem ser capazes de usar os dados do resultado do pareamento, incluindo IDs de jogadores e atribuições de equipe, ao configurar uma sessão de jogo para a partida.

- O FlexMatch como um serviço independente: Seu serviço de posicionamento de correspondência personalizado obtém dados de resultados de correspondência de eventos bem-sucedidos de matchmaking e se conecta ao seu sistema de posicionamento de sessão de jogo existente para localizar um recurso de hospedagem disponível para a partida. Depois que um recurso de hospedagem é encontrado, o serviço de posicionamento de correspondência coordena com seu sistema de hospedagem existente para iniciar uma nova sessão de jogo e adquirir informações de conexão.
- FlexMatch com uma solução de hospedagem GameLift: A fila de sessão de jogo localiza o melhor servidor de jogo disponível para a correspondência. Dependendo de como a fila está configurada, ela tenta colocar a sessão do jogo com os recursos de menor custo e onde os jogadores terão baixa latência (se os dados de latência do jogador forem fornecidos). Uma vez que a sessão do jogo é colocada com sucesso, o serviço GameLift solicita que o servidor do jogo inicie uma nova sessão de jogo, transmitindo os resultados da criação de partidas e outros dados opcionais do jogo.

Connect jogadores à partida

Depois que uma sessão de jogo é iniciada, os jogadores se conectam à sessão, reivindicam a atribuição da equipe e iniciam a jogabilidade.

- O FlexMatch como um serviço independente: Seu jogo usa o sistema de gerenciamento de sessão de jogo existente para fornecer informações de conexão de volta aos jogadores.
- FlexMatch com uma solução de hospedagem GameLift: Em um posicionamento bem-sucedido da sessão de jogo, o FlexMatch atualiza todos os tickets correspondentes com informações de conexão de sessão de jogo e um ID de sessão de jogador.

Como configurar o FlexMatch

O GameLift FlexMatch é umAWSserviço, e você deve ter umAWSconta para usar este serviço. Criar uma conta da AWS é grátis. Para obter mais informações sobre o que é possível fazer com umAWSconta, consulteConceitos básicos doAWS.

Se você estiver usando o FlexMatch com outras soluções do GameLift, consulte os tópicos a seguir:

- Configurando o acesso para hospedagem GameLift e Servidores em Tempo Real
- Como configurar o acesso para hospedagem no Amazon EC2 com o GameLift FleetIQ

Para configurar sua conta do GameLift

1. Obtenha uma conta. AbertoAmazon Web Servicese escolhaFazer login no console. Siga as solicitações para criar uma nova conta ou fazer login em uma existente.
2. Configure um grupo de usuários administrativos. Abrir oAWS Identity and Access Management(IAM) console de serviço e siga as etapas para criar ou atualizar usuários ou grupos de usuários. O IAM gerencia o acesso ao seuAWSServiços e recursos. Todos que acessarem seus recursos FlexMatch, usando o console GameLift ou chamando APIs do GameLift, devem ter acesso explícito. Para obter instruções detalhadas sobre como usar o console (ou oAWSCLI ou outras ferramentas) para configurar grupos de usuários, consulteCriação de usuários do IAM.
3. Anexe uma política de permissões ao seu usuário ou grupo de usuários.Acesso aoAWSserviços e recursos são gerenciados anexando umPolítica do IAMPara um usuário ou grupo de usuários. As políticas de permissões especificam um conjunto deAWSserviços e ações aos quais um usuário precisa ter acesso.

   Para o GameLift, é necessário criar uma política de permissões personalizada e anexe-a a cada usuário ou grupo de usuários. Uma política é um documento JSON. Use o exemplo abaixo para criar sua política.

O exemplo a seguir ilustra uma política de permissões embutidas com permissões administrativas para todos os recursos e ações do GameLift. Você pode optar por limitar o acesso especificando somente itens específicos do FlexMatch.

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

Conceitos básicos do FlexMatch

Use os recursos nesta seção para ajudar você a começar a criar um sistema de criação de jogos de jogos do FlexMatch.

Tópicos

GameLift FlexMatch integração para matchmaking autônomo

Este tópico descreve o processo completo de integração para implementação FlexMatch como um serviço de Marcação de jogos independente. Use esse processo se seu jogo multijogador estiver hospedado usando peer-to-peer, hardware local com configuração personalizada ou outras primitivas de computação em nuvem. Esse processo também é para uso com GameLift O FleetIQ, que é uma solução de otimização de hospedagem para jogos hospedados no Amazon EC2. Se você estiver hospedando seu jogo usando GameLift hospedagem gerenciada (incluindo servidores em tempo real), consulteIntegração FlexMatch com hospedagem GameLift (p. 8).

Antes de iniciar a integração, você deve ter umaAWSconta e configure permissões de acesso para o GameLift serviçoServiço da Para obter detalhes, consulte Como configurar o FlexMatch (p. 6). Todas as tarefas essenciais relacionadas à criação e gerenciamento GameLift FlexMatch matchmakers e conjuntos de regras podem ser feitos usando a Amazon GameLift console, mas você também pode querer

1. Criar um FlexMatch conjunto de regras da correspondência. Seu conjunto de regras personalizado fornece instruções completas sobre como criar uma correspondência. Nele, você define a estrutura e o tamanho de cada equipe. Você também fornece um conjunto de requisitos que uma partida deve atender para ser válida, que FlexMatch usa para incluir ou excluir jogadores em uma partida. Esses requisitos podem se aplicar a jogadores individuais. Você também pode personalizar a FlexMatch algoritmo no conjunto de regras, como construir grandes partidas com até 200 jogadores. Consulte os seguintes tópicos:

   - Criar um conjunto de regras do FlexMatch (p. 16)
   - Exemplos de conjunto de regras FlexMatch (p. 26)
2. Configuração de notificações para eventos da Marcação de jogos. Use notificações para rastrear FlexMatch atividade de matchmaking, incluindo o status de solicitações de partidas pendentes. Esse é o mecanismo usado para fornecer os resultados de uma combinação proposta. Como as solicitações de marcação de jogos são assíncronas, você precisa de uma maneira de acompanhar o status das solicitações. Usar notificações é a opção preferida para isso. Consulte os seguintes tópicos:

   - Configurar FlexMatch Notificações de eventos do (p. 42)
   - Eventos de marcação do FlexMatch (p. 75)
3. Configurar um FlexMatch configuração da correspondência de jogos. Também chamado de matchmaker, esse componente recebe solicitações de matchmaking e as processa. Você configura um matchmaker especificando um conjunto de regras, alvo de notificação e tempo máximo de espera. Você também pode habilitar os recursos opcionais. Consulte os seguintes tópicos:

   - Projete um FlexMatch casamenteiro (p. 10)

- Criar uma configuração marcação de jogos (p. 12)

4. Crie um serviço de marcador de jogos para clientes Crie ou expanda um serviço de cliente de jogos com funcionalidade para criar e enviar solicitações de matchmaking para FlexMatch. Para criar solicitações de matchmaking, esse componente deve ter mecanismos para obter os dados do jogador exigidos pelo conjunto de regras de matchmaking e, opcionalmente, informações de latência regional. Ele também deve ter um método para criar e atribuir IDs de ticket exclusivos para cada solicitação. Você também pode optar por criar um fluxo de trabalho de aceitação de jogadores que exija que os jogadores optem por uma partida proposta. Este serviço também deve monitorar eventos de matchmaking para obter os resultados das partidas e iniciar a colocação da sessão de jogo para partidas bem-sucedidas. Consulte este tópico:

   - Adicionar FlexMatch para um cliente de jogo (p. 46)

5. Crie um serviço de colocação de fósforos. Crie um mecanismo que funcione com seu sistema de hospedagem de jogos existente para localizar os recursos de hospedagem disponíveis e iniciar novas sessões de jogo para partidas bem-sucedidas. Esse componente deve ser capaz de usar informações dos resultados da correspondência para obter um servidor de jogos disponível e iniciar uma nova sessão de jogo para a correspondência. Você também pode querer implementar um fluxo de trabalho para fazer solicitações de preenchimento de partidas, que usa matchmaking para preencher vagas abertas em sessões de jogos correspondentes que já estão em execução.

Integração FlexMatch com hospedagem GameLift

A FlexMatch está disponível com a hospedagem gerenciada do GameLift para servidores de jogos personalizados e servidores em tempo real. Para adicionar a criação de jogos do FlexMatch ao seu jogo, conclua as tarefas a seguir.

- Configurar um marcador de jogos. Um marcador de jogos recebe as solicitações de marcação de jogos dos jogadores e as processa. Ele agrupa os jogadores com base em um conjunto de regras definidas e, para cada correspondência bem-sucedida, cria uma nova sessão de jogo e de jogador. Para configurar um marcador de jogos, siga estas etapas:
  - Criar um conjunto de regras. Um conjunto de regras informa ao marcador de jogos como construir uma correspondência válida. Ele especifica a estrutura da equipe e como avaliar os jogadores para inclusão em uma correspondência. Consulte os seguintes tópicos:
    - Criar um conjunto de regras do FlexMatch (p. 16)
    - Exemplos de conjunto de regras FlexMatch (p. 26)
  - Criar uma fila de sessões de jogos. Uma fila localiza a melhor região para cada correspondência e cria uma nova sessão do jogo nessa região. Use uma fila existente ou criar uma nova para marcação de jogos. Consulte este tópico:
    - Criar uma fila
  - Configurar notificações (opcional). Como as solicitações de marcação de jogos são assíncronas, você precisa de uma maneira de acompanhar o status das solicitações. Notificações é a opção preferida. Consulte este tópico:
    - Configurar FlexMatch Notificações de eventos do (p. 42)
  - Configurar um marcador de jogos. Uma vez que você tem o conjunto de regras, a fila e o destino das notificações, crie a configuração para o seu marcador de jogos. Consulte os seguintes tópicos:
    - Projete um FlexMatch casamenteiro (p. 10)
    - Criar uma configuração marcação de jogos (p. 12)
- Integrar o FlexMatch ao seu serviço de cliente de jogos. Adicione funcionalidades ao seu serviço de cliente de jogos para iniciar novas sessões com a marcação de jogos. As solicitações de marcação de jogos especifica qual marcador deve ser usado e fornece os dados necessários do jogador para a correspondência. Consulte este tópico:
  - Adicionar FlexMatch para um cliente de jogo (p. 46)

- Integrar o FlexMatch ao seu servidor de jogos. Adicione funcionalidades ao servidor de jogos para iniciar sessões criadas por meio da marcação de jogos. As solicitações para esse tipo de sessão de jogo incluem informações específicas de correspondência, incluindo os jogadores e as atribuições de equipe. O servidor de jogos precisa acessar e usar essas informações ao calcular uma sessão de jogo para a correspondência. Consulte este tópico:
  - Adicionar o FlexMatch a um servidor de jogos hospedado pelo Gamelift (p. 51)
- Configurar a alocação do FlexMatch (opcional). Solicite novas correspondências de jogadores para preencher os slots de jogadores em jogos existentes. Você pode ativar a lotação automática para que o GameLift gerencie as solicitações de preenchimento. Ou você pode gerenciar a alocação manualmente adicionando funcionalidades ao serviço de cliente de jogos ou servidor de jogos para iniciar as solicitações de alocação de correspondência. Consulte este tópico:
  - Enchimento de jogos existentes com FlexMatch (p. 53)

    Note

    A disponibilização do FlexMatch não está disponível no momento para jogos que usam servidores em tempo real.

Construindo um GameLift FlexMatch casamenteiro

UMA FlexMatch processa o jogo do faz o trabalho de criar uma partida de jogo. Ele gerencia o grupo de solicitações de marcação de jogos recebidas, forma as equipes de uma correspondência, processa e seleciona jogadores para encontrar os melhores grupos de jogadores possível e inicia o processo de criar e iniciar uma sessão de jogos para a correspondência. Este tópico descreve os principais aspectos de um marcador de jogos e como configurar um marcador personalizado para o seu jogo.

Para obter uma descrição detalhada de como um FlexMatch processa as solicitações que recebe, consulteProcesso de marcar jogos do FlexMatch (p. 4).

Tópicos

Projete um FlexMatch casamenteiro

Este tópico fornece orientações sobre como projetar um marcador de jogos adequado ao seu jogo.

Configurar um marcador de jogos básico

No mínimo, um marcador de jogos precisa dos seguintes elementos:

- O conjunto de regras determina o tamanho e o escopo das equipes de uma correspondência e define um conjunto de regras para usar ao avaliar jogadores para um correspondência. Cada marcador de jogos é configurado para usar um conjunto de regras. Consulte Criar um conjunto de regras do FlexMatch (p. 16) e Exemplos de conjunto de regras FlexMatch (p. 26).
- Odestino de notificaçãoRecebe todas as notificações de eventos de marcações. Você precisa configurar um tópico do Amazon Simple Notification Service (SNS) e adicionar o ID do tópico ao marcador de jogos. Veja mais informações sobre a configuração de notificações em Configurar FlexMatch Notificações de eventos do (p. 42).
- O tempo de espera da solicitação determina quanto tempo uma solicitação de marcação de jogo pode permanecer no grupo de solicitações e ser avaliada para jogos potenciais. Quando uma solicitação expira, a correspondência do jogo falha e ela é removida do grupo.
- Ao usar FlexMatch com GameLift hospedagem gerenciada, ofila de sessões de jogosEncontra os melhores recursos disponíveis para hospedar uma sessão de jogo para a correspondência e inicia uma nova sessão de jogo. Cada fila é configurada com uma lista deAWSRegiões e tipos de recursos (incluindo instâncias spot ou sob demanda) que determinam onde as sessões do jogo podem ser colocadas. Para obter mais informações sobre filas, consulteUsar filas de várias regiões.

Selecione umAWSRegião para o marcador de jogos

Decida onde você deseja que a atividade de pareamento ocorra e crie seu matchmaker (configuração de matchmaking e conjunto de regras) nessa Região. Todas as solicitações para o matchmaker são enviadas

para um pool de tickets lá, onde são classificados e avaliados para correspondências viáveis. Depois que as partidas forem feitas, os jogadores podem ser direcionados para sessões de jogo em qualquer local que seja compatível com sua solução de hospedagem.

Ao escolherAWSRegiões para seus casamenteiros, considere como a localização pode afetar o desempenho deles e como otimizar a experiência de partida para os jogadores pretendidos. Recomendamos as seguintes melhores práticas:

- Coloque um casamenteiro em umAWSregião próxima aos seus jogadores — e do serviço ao cliente que envia FlexMatch Solicitar marcações de jogos. Essa abordagem diminui o efeito de latência em seu fluxo de trabalho de solicitação de pareamento e o torna mais eficiente.
- Se seu jogo atingir um público global, considere criar matchmakers em várias regiões e encaminhar solicitações de correspondência para o matchmaker mais próximo do jogador. Além de aumentar a eficiência, isso faz com que os grupos de ingressos se formem com jogadores geograficamente próximos uns dos outros, o que melhora a capacidade do matchmaker de combinar jogadores com base nos requisitos de latência.
- Ao usar FlexMatch com GameLift Hospedagem gerenciada, coloque seu marcador de jogos e a fila de sessões de jogo usada na mesmaAWSRegião : Isso ajuda a minimizar a latência de comunicação entre o marcador de jogos e a fila.

O FlexMatch recursosMatchMakingConfigurationeMatchmakingRuleSetpode ser colocado nos seguintes suportados pelo GameliftAWSRegiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico Nordeste (Sydney), Ásia-Pacífico Nordeste (Seul e Tóquio), Ásia-Pacífico Nordeste (Seul e Tóquio) e China (Regiões de Pequim e Ningxia).

Adicionar elementos opcionais

Além dessas exigências mínimas, você pode configurar seu marcador de jogos com as seguintes opções adicionais. Se você estiver usando FlexMatch com um GameLift solução de hospedagem, muitos recursos são incorporados. Se você estiver usando FlexMatch como um serviço autônomo de matchmaking, talvez você queira criar esses recursos em seu sistema.

Aceitação de jogador

Você pode configurar um marcador de jogos para exigir que todos os jogadores selecionados para uma correspondência devam aceitar a participação. Se o seu sistema exigir aceitação, todos os jogadores devem ter a opção de aceitar ou rejeitar uma correspondência proposta. Uma correspondência deve receber aceitações de todos os jogadores na correspondência proposta antes que ela possa ser concluída. Se qualquer jogador rejeitar ou não conseguir aceitar uma correspondência, ela será descartada e os tíquetes serão tratados da seguinte forma. Os tíquetes em que todos os jogadores do tíquete aceitaram a correspondência são retornados ao grupo de marcações para continuar o processamento. Os tíquetes em que pelo menos um jogador rejeitou a correspondência ou não respondeu são colocados em status de falha e não são mais processados. A aceitação do jogador exige um limite de tempo; todos os jogadores devem aceitar uma correspondência proposta dentro do limite de tempo para a correspondência continuar.

Modo de alocação

Usar o FlexMatch Para manter suas sessões de jogo preenchidas com novos jogadores de boa correspondência durante toda a vida útil da sessão do jogo. Ao lidar com solicitações de preenchimento, FlexMatch Usar o mesmo marcador de jogos usado para combinar com os jogadores originais. Você pode personalizar como os ingressos de aterro são priorizados com ingressos para novas partidas, colocando os ingressos de aterro na frente ou no final da linha. Isso significa que, à medida que novos jogadores entram no pool de jogos, eles são mais ou menos propensos a serem colocados em um jogo existente do que em um jogo recém-formado.

O preenchimento manual está disponível se o jogo usa FlexMatch gerenciado com o GameLift hospedagem ou com outras soluções de hospedagem. A alocação manual oferece a flexibilidade de

decidir quando acionar uma solicitação. Por exemplo, você pode adicionar novos jogadores somente durante determinadas fases do seu jogo ou somente quando certas condições existirem.

O preenchimento automático está disponível apenas para jogos que usam gerenciado GameLift hospedagem. Com esse recurso ativado, se uma sessão de jogo começar com slots de jogador abertos, GameLift começa a gerar automaticamente solicitações de preenchimento para ele. Esse recurso permite que você configure o matchmaking para que novos jogos sejam iniciados com um número mínimo de jogadores e, em seguida, rapidamente preenchidos à medida que novos jogadores entram no pool de matchmaking. Você pode desativar o preenchimento automático a qualquer momento durante a vida útil da sessão do jogo.

Propriedades de jogo

Para jogos que usam FlexMatch com GameLift hospedagem gerenciada, você pode fornecer informações adicionais a serem passadas para um servidor de jogo sempre que uma nova sessão de jogo for solicitada. Essa pode ser uma maneira útil de passar nas configurações do modo de jogo necessárias para iniciar uma sessão de jogo para o tipo de correspondências que estão sendo criadas. Todas as sessões de jogo para jogos criadas por um marcador de jogos recebem o mesmo conjunto de propriedades do jogo. Você pode variar as informações da propriedade do jogo criando diferentes configurações de matchmaking.

Slots de jogador reservados

É possível designar que determinados slots de jogador em cada correspondência sejam reservados e preenchidos posteriormente. Isso é feito através da configuração da propriedade de "contagem de jogador adicional" de uma configuração de marcação de jogos.

Dados de eventos personalizados

Use esta propriedade para incluir um conjunto de informações personalizadas em todos os eventos relacionados à marcação de jogos do marcador de jogos. Esse recurso pode ser útil para rastrear determinadas atividades exclusivas ao seu jogo, incluindo o desempenho do rastreamento de seus marcadores de jogos.

Criar uma configuração marcação de jogos

Para configurar um GameLift FlexMatch matchmaker para processar casamento solicitações, criar uma configuração de combinação. Use o GameLift Console do ou oAWS Command Line Interface(AWS CLI). Para obter mais informações sobre como criar um marcador de jogos, consulteProjete um FlexMatch casamenteiro (p. 10).

Criar um marcador de jogos para o GameLift hospedagem

Antes de criar uma configuração de combinação,criar um conjunto de regras (p. 25)e um GameLift fila de sessões de jogospara usar com o casamenteiro.

Console

1. Abrir o GameLift Console do nohttps://console.aws.amazon.com/gamelift/home.
2. Alterne para oAWSRegião onde você deseja criar seu marcador de jogos. Para obter uma lista de regiões compatíveis com o FlexMatch configurações de combinação, consulteSelecione umAWSRegião para o marcador de jogos (p. 10).
3. No painel de navegação, selecioneFlexMatch,Configurações de marcação de jogos.
4. NoConfigurações de marcação de jogospágina, selecioneCriar configuração configuração.

5. NoDefinir detalhes da configuração configuraçãopágina, abaixoDetalhes da configuração de marcação, faça o seguinte:

   a. para oName (Nome), insira um nome de marcador de jogos que possa ajudá-lo a identificá-lo em uma lista e em métricas. O nome do marcador de jogos deve ser exclusivo na região. As solicitações de marcação de jogos identificam qual marcador de jogos deve ser usado por seu nome e região.

   b. (Opcional) ParaDescrição, adicione uma descrição para ajudar a identificar o marcador de jogos.

   c. para oConjunto de regras, escolha um conjunto de regras da lista para usar com o matchmaker. A lista contém todos os conjuntos de regras que você criou na região da atual.

   d. para oFlexMatch modo, escolhagerenciadopelo GameLift hospedagem gerenciada. Este modo solicita FlexMatch para passar partidas bem-sucedidas para a fila de sessões de jogo especificada.

   e. para oAWSRegião, escolha a região onde você configurou a fila de sessões de jogos que deseja usar com o marcador de jogos.

   f. para oFila do, escolha a fila de sessões de jogos que deseja usar com o marcador de jogos.

6. Escolha Next (Próximo).

7. NoDefinição de configuraçõespágina, abaixoConfigurações marcação de jogos, faça o seguinte:

   a. para oTempo limite da solicitação, defina o tempo máximo, em segundos, para que o marcador de jogos conclua uma correspondência para cada solicitação. As solicitações de marcação de jogos que excederem esse tempo serão rejeitadas

   b. para oModo de preenchimento, selecione um modo para lidar com as alocações de correspondência. Para ativar o recurso de preenchimento automático, selecione o recurso de preenchimento automático, selecioneAutomatic. Ou, se você estiver gerenciando solicitações de atribuição em seu servidor de jogos ou cliente de jogo, ou se você optar por não preencher os jogos, escolhaManual.

   c. (Opcional) ParaNúmero adicional de jogadores, defina o número de slots de jogador para manter em aberto em uma correspondência. FlexMatch pode preencher esses slots com jogadores no future.

   d. (Opcional) EmOpções de aceitação de jogos, paraAceitação necessária, se você quiser exigir que cada jogador em uma correspondência proposta aceite ativamente a participação na correspondência, selecioneObrigatório. Se você selecionar essa opção, selecione essa opção, o paraTempo limite de aceitação do, defina quanto tempo, em segundos, você deseja que o marcador de jogos espere por aceitações do jogador antes de cancelar a correspondência.

8. (Opcional) EmConfigurações de notificação de eventos, faça o seguinte:

   a. (Opcional) ParaTópico do SNS, escolha um tópico do Amazon Simple Notification Service (Amazon SNS) para receber notificações de evento de marcação de jogos. Caso ainda não tenha configurado um tópico do SNS, você pode escolher isso mais tarde, editando a configuração de marcação de jogos. Para obter mais informações, consulte Configurar FlexMatch Notificações de eventos do (p. 42).

   b. (Opcional) ParaDados de eventos personalizados, insira quaisquer dados personalizados que deseje associar com este marcador de jogos no sistema de mensagens do evento. FlexMatch inclui esses dados em cada evento associado ao marcador de jogos.

9. (Opcional) ExpandaOutros dados de jogoe, em seguida, faça o seguinte:

   a. (Opcional) ParaDados da sessão de jogo, insira qualquer informação adicional relacionada ao jogo que você deseja FlexMatch para entregar novas sessões de jogo iniciadas com partidas feitas usando este casamento configuração.

   b. (Opcional) ParaPropriedades de jogo, adicione propriedades de par de chave/valor que contenham informações sobre uma nova sessão de jogo.

10. (Opcional) EmTags, adicione tags para ajudá-lo a gerenciar e rastrearAWSrecursos da AWS.
11. Escolha Next (Próximo).
12. NoRevisar e criarrevise suas escolhas e, em seguida, selecioneCriar. Após a criação bem-sucedida, o marcador de jogos ficará pronto para aceitar solicitações de marcação de jogos.

AWS CLI

Para criar uma configuração de marcação de jogos com a AWS CLI, abra uma janela de linha de comando e use o comando create-matchmaking-configuration para definir um novo marcador de jogos.

Este comando de exemplo cria uma nova configuração de marcação de jogos que exige a aceitação do jogador e permite o preenchimento automático. Ele também reserva dois slots de jogador para o FlexMatch para adicionar jogadores mais tarde e fornece alguns dados da sessão de jogo.

```
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"
```

Se a solicitação de criação da configuração para marcação de jogos for bem-sucedida, GameLiftO retorna umMatchmakingConfigurationobjeto com as configurações que você solicitou para o marcador de jogos. O novo marcador de jogos ficará pronto para aceitar solicitações de marcação de jogos.

Crie um matchmaker para autônomo FlexMatch

Antes de criar uma configuração de combinação,criar um conjunto de regras (p. 25)para usar com o casamenteiro.

Console

1. Abrir o GameLift Console do nohttps://console.aws.amazon.com/gamelift/home.
2. Alterne para oAWSRegião onde você deseja criar seu marcador de jogos. Para obter uma lista de regiões compatíveis com o FlexMatch configurações de combinação, consulteSelecione umAWSRegião para o marcador de jogos (p. 10).
3. No painel de navegação, selecioneFlexMatch,Configurações de marcação de jogos.
4. NoConfigurações de marcação de jogospágina, selecioneCriar configuração configuração.
5. NoDefinir detalhes da configuração configuraçãopágina, abaixoDetalhes da configuração de marcação, faça o seguinte:

    a. para oName (Nome), insira um nome de marcador de jogos que possa ajudá-lo a identificá-lo em uma lista e em métricas. O nome do marcador de jogos deve ser exclusivo na região. As solicitações de marcação de jogos identificam qual marcador de jogos deve ser usado por seu nome e região.

b. (Opcional) ParaDescrição, adicione uma descrição para ajudar a identificar o marcador de jogos.
c. para oConjunto de regras, escolha um conjunto de regras da lista para usar com o matchmaker. A lista contém todos os conjuntos de regras que você criou na região da atual.
d. para oFlexMatch modo, escolhaIndependente. Isso indica que você tem um mecanismo personalizado para iniciar novas sessões de jogo em uma solução de hospedagem fora do GameLift.

6. Escolha Next (Próximo).
7. NoDefinição de configuraçõespágina, abaixoConfigurações marcação de jogos, faça o seguinte:

a. para oTempo limite da solicitação, defina o tempo máximo, em segundos, para que o marcador de jogos conclua uma correspondência para cada solicitação. As solicitações de marcação de jogos que excederem esse tempo serão rejeitadas
b. (Opcional) EmOpções de aceitação de jogos, paraAceitação necessária, se você quiser exigir que cada jogador em uma correspondência proposta aceite ativamente a participação na correspondência, selecioneObrigatório. Se você selecionar essa opção, selecione essa opção, o paraTempo limite de aceitação do, defina quanto tempo, em segundos, você deseja que o marcador de jogos espere por aceitações do jogador antes de cancelar a correspondência.

8. (Opcional) EmConfigurações de notificação de eventos, faça o seguinte:

a. (Opcional) ParaTópico do SNS, escolha um tópico do Amazon SNS para receber notificações de evento de marcação de jogos. Caso ainda não tenha configurado um tópico do SNS, você pode escolher isso mais tarde, editando a configuração de marcação de jogos. Para obter mais informações, consulte Configurar FlexMatch Notificações de eventos do (p. 42).
b. (Opcional) ParaDados de eventos personalizados, insira quaisquer dados personalizados que deseje associar com este marcador de jogos no sistema de mensagens do evento. FlexMatch inclui esses dados em cada evento associado ao marcador de jogos.

9. (Opcional) EmTags, adicione tags para ajudá-lo a gerenciar e rastrearAWSrecursos da AWS.
10. Escolha Next (Próximo).
11. NoRevisar e criarrevise suas escolhas e, em seguida, selecioneCriar. Após a criação bem-sucedida, o marcador de jogos ficará pronto para aceitar solicitações de marcação de jogos.

AWS CLI

Para criar uma configuração de marcação de jogos com a AWS CLI, abra uma janela de linha de comando e use o comando create-matchmaking-configuration para definir um novo marcador de jogos.

Este comando de exemplo cria uma nova configuração de combinação para um matchmaker independente que requer a aceitação do jogador.

```
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"
```

Se a solicitação de criação da configuração para marcação de jogos for bem-sucedida, GameLiftO retorna umMatchmakingConfigurationobjeto com as configurações que você solicitou para o marcador de jogos. O novo marcador de jogos ficará pronto para aceitar solicitações de marcação de jogos.

Criar um conjunto de regras do FlexMatch

Cada marcador de jogos do FlexMatch deve ter um conjunto de regras. O conjunto de regras determina os dois elementos principais de uma correspondência, a estrutura e o tamanho da equipe, e como agrupar jogadores para a melhor correspondência de jogo possível

Por exemplo, um conjunto de regras pode descrever uma correspondência como esta: Criar uma correspondência com duas equipes de cinco jogadores cada, uma equipe é dos defensores e a outra equipe é dos invasores. Uma equipe pode ter jogadores iniciantes e especialistas, mas a habilidade média das duas equipes deve estar entre 10 pontos uma da outra. Se nenhuma correspondência for encontrada após 30 segundos, diminua gradualmente as exigências de habilidade.

Os tópicos nesta seção descrevem como projetar e criar um conjunto de regras de marcação de jogos. Ao criar um conjunto de regras, use o console do Amazon GameLift ou oAWSCLI.

Tópicos

Projetar um FlexMatch conjunto de regras

Este tópico aborda a estrutura básica de um conjunto de regras e como criar um conjunto de regras para pequenas correspondências (até 40 jogadores). Um conjunto de regras de marcação de jogos deve fazer duas coisas: definir um tamanho e uma estrutura para a equipe da correspondência e dizer ao marcador de jogos como escolher os jogadores e formar a melhor correspondência possível.

Mas seu conjunto de regras de matchmaking pode fazer mais. Por exemplo, é possível:

- Otimize o algoritmo de marcação de jogos do para o seu jogo.
- Defina os requisitos de latência mínima do jogador para proteger a qualidade do jogo.
- Relaxe gradualmente os requisitos da equipe e as regras de jogo ao longo do tempo, para que todos os jogadores ativos possam encontrar uma partida aceitável quando quiserem.
- Defina o tratamento de solicitações de matchmaking em grupo (agregação de grupos).
- Processe partidas grandes (mais de 40 jogadores). Saiba mais sobre a criação de correspondências grandes em Projetar um FlexMatch conjunto de regras de grande correspondência (p. 22).

Ao criar um conjunto de regras de marcação de jogos, você pode realizar algumas ou todas as seguintes tarefas:

- Descreva o conjunto de regras (p. 17)(obrigatório)
- Personalize o algoritmo de correspondência (p. 17) (Opcional)
- Declarar os atributos do jogador (p. 20)
- Defina as equipes da partida (p. 20)
- Defina regras para combinar jogadores (p. 21)
- Deixe que os requisitos relaxem com o tempo (p. 21)

Descreva o conjunto de regras

Forneça os detalhes do conjunto de regras.

- nome(opcional) — Um rótulo descritivo para seu próprio uso. Esse valor não está associado ao nome do conjunto de regras que você especifica ao criar o conjunto de regras com GameLift.
- ruleLanguageVersion(obrigatório) — A versão do idioma de expressão de propriedade usada para criar FlexMatch regras. O valor deve ser`1.0`.

Personalize o algoritmo de correspondência

Você pode alterar o algoritmo de correspondência padrão. O algoritmo padrão é otimizado para a maioria dos jogos para colocar os jogadores em partidas aceitáveis com o mínimo de tempo de espera. Você pode personalizar o algoritmo e ajustar o matchmaking para o seu jogo.

O valor FlexMatch o processo de matchmaking é o seguinte:

1. FlexMatch coloca todos os ingressos abertos de matchmaking e os ingressos preenchidos em uma bilheteria.
2. Os ingressos na piscina são agrupados aleatoriamente em um ou mais lotes para serem combinados. À medida que a bilheteria fica maior, FlexMatch forma lotes adicionais para manter o tamanho ideal do lote.
3. Dentro de cada lote, FlexMatch classifica os ingressos por idade.
4. FlexMatch cria uma combinação com base no tíquete mais antigo de cada lote.

Para personalizar o algoritmo de correspondência, adicione um`algorithm`componente do seu esquema de conjunto de regras. ConsulteEsquema conjunto de regras FlexMatch (p. 60)para obter as informações de referência completas do.

Use as seguintes personalizações opcionais para impactar diferentes estágios do seu processo de matchmaking.

- Adicionar classificação pré-lote (p. 17)
- Forme lotes com base nos atributos batchDistance
- Priorizar tíquetes de preenchimento (p. 18)
- Prefira ingressos mais antigos com expansões (p. 19)

Adicionar classificação pré-lote

Você pode configurar FlexMatch para classificar a bilheteria antes de formar lotes. Esse tipo de personalização é mais eficaz em jogos com grandes pools de ingressos. A classificação pré-lote pode ajudar a acelerar o processo de combinação e aumentar a uniformidade do jogador nas características definidas.

Defina métodos de classificação pré-lote usando a propriedade do algoritmo`batchingPreference`. A configuração padrão é a `random`.

As opções para personalizar a classificação pré-lote incluem:

- Classifique por atributos do jogador. Forneça uma lista de atributos do jogador para pré-ordenar o pool de ingressos. FlexMatch em seguida, cria lotes com mais uniformidade nos atributos classificados. Por exemplo, se você pré-classificar o pool de ingressos por habilidade do jogador, FlexMatch reúne tíquetes com níveis de habilidade semelhantes. Se o seu conjunto de regras também contém regras de jogo com base na habilidade do jogador, a classificação pré-lote pode melhorar a eficiência do matchmaking.

Para classificar por atributos do jogador, defina`batchingPreference`para`sorted`e defina`sortByAttributes`para a lista de atributos do jogador. Para usar um atributo, declare o atributo na`playerAttributes`componente do conjunto de regras.

No exemplo a seguir, FlexMatch classifica o pool de ingressos com base no mapa de jogo preferido dos jogadores e, em seguida, pela habilidade do jogador. É mais provável que os lotes resultantes contenham jogadores com habilidades semelhantes que desejam usar o mesmo mapa.

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

- Classifique por latência. Priorize com base em correspondências com a menor latência disponível ou criando rapidamente correspondências com latência aceitável. Essa personalização é útil para conjuntos de regras que formam partidas grandes (mais de 40 jogadores). A propriedade do algoritmo`strategy`deverá ser definido como`balanced`, o que limita os tipos disponíveis de instruções de regra. Consulte Projetar um FlexMatch conjunto de regras de grande correspondência (p. 22).

  FlexMatch pré-classifica os tickets com base nos dados de latência relatados de uma das seguintes formas:

  - Coloque os jogadores nas regiões de menor latência. O pool de ingressos é pré-classificado pelas regiões onde os jogadores relatam seus menores valores de latência. FlexMatch em seguida, agrupa tíquetes com baixa latência nas mesmas regiões, criando uma melhor experiência geral de jogo. Também reduz o número de ingressos em cada lote, então a combinação pode levar mais tempo. Para usar essa personalização, defina`batchingPreference`para`fastestRegion`, conforme mostrado no exemplo a seguir.

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

  - Coloque os jogadores em partidas de latência aceitáveis rapidamente. O pool de ingressos é pré-classificado por regiões, onde os jogadores relatam qualquer valor de latência aceitável. Isso forma menos lotes contendo mais tickets com latência aceitável nas mesmas regiões. Com mais ingressos em cada lote, encontrar combinações aceitáveis suficientes é mais fácil e rápido. Para usar essa personalização, defina a propriedade`batchingPreference`para `largestPopulation`, conforme mostrado no exemplo a seguir.

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

    Note

    `largetPopulation`é a configuração padrão para conjuntos de regras que usam a estratégia balanceada.

Priorizar tíquetes de preenchimento

Se o seu jogo implementar preenchimento automático ou preenchimento manual, você pode personalizar como FlexMatch processa tickets de matchmaking com base no tipo de solicitação (nova partida ou solicitação de preenchimento). Por padrão, FlexMatch trata os dois tipos de solicitações da mesma

forma. Você pode determinar se FlexMatch tenta preencher os tíquetes de preenchimento primeiro ou se FlexMatch preenche o tíquete de preenchimento quando novas partidas não podem ser feitas.

A priorização do preenchimento afeta como FlexMatch lida com os tíquetes depois de serem lotados. Use apenas a priorização de preenchimento com conjuntos de regras que usam a estratégia de pesquisa exaustiva. FlexMatch não corresponde a vários tíquetes preenchidos.

Para alterar a priorização dos tíquetes de preenchimento, defina a propriedade`backfillPriority`como segue:

- Combine primeiro os ingressos preenchidos Esta opção prompts FlexMatch para tentar preencher os tíquetes antes de criar novas partidas. Isso significa que os jogadores entrantes têm uma chance maior de entrar em um jogo existente. Defina `backfillPriority` para `high`.

  Se o seu jogo estiver usando preenchimento automático, use isso como uma prática recomendada. O preenchimento automático é mais frequentemente usado em jogos com períodos curtos de sessões de jogo e alto tempo de resposta do jogador. O preenchimento automático ajuda esses jogos a formar rapidamente o mínimo de partidas viáveis e iniciá-las, mesmo que FlexMatchprocura mais jogadores para preencher as vagas abertas.

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

- Os ingressos preenchidos pela última vez. Esta opção prompts FlexMatch ignorar os tickets preenchidos até que ele avalie todos os outros tickets. Isto significa que FlexMatch preenche os novos jogadores em jogos existentes quando não consegue combiná-los com novos jogos. Defina `backfillPriority` para `low`.

  Essa opção é útil quando você deseja usar o preenchimento como uma opção de última chance para colocar jogadores em um jogo, como quando há poucos jogadores para formar uma nova partida. Não despriorize o preenchimento automático de tíquetes com o preenchimento automático.

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

Prefira ingressos mais antigos com expansões

É possível personalizar como FlexMatch aplica regras de expansão, que relaxam os critérios de correspondência quando as partidas são difíceis de concluir. GameLift aplica regras de expansão quando os tíquetes de uma partida parcialmente concluída atingem uma certa idade. Os carimbos de data/hora de criação dos tickets determinam quando GameLift aplica as regras; por padrão, FlexMatch rastreia a data e hora do tíquete correspondido mais recentemente.

Para mudar quando FlexMatch aplica regras de expansão, define a propriedade`expansionAgeSelection`como segue:

- Expanda com base nos ingressos mais recentes. Essa opção aplica regras de expansão com base no tíquete mais recente adicionado à possível correspondência. Cada vez FlexMatch corresponde a um novo tíquete, o relógio é zerado. Com essa opção, as partidas resultantes tendem a ser de maior qualidade, mas demoram mais para corresponder; as solicitações de partidas podem expirar antes de serem concluídas se demorarem muito para serem combinadas. Defina`expansionAgeSelection`para`newest`.`newest`é padrão.
- Expanda com base nos ingressos mais antigos. Essa opção aplica regras de expansão com base no tíquete mais antigo da possível correspondência. Com essa opção, FlexMatch aplica expansões mais

rapidamente, o que melhora os tempos de espera dos primeiros jogadores combinados, mas diminui a qualidade da partida para todos os jogadores. Defina `expansionAgeSelection` para `oldest`.

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

Declarar os atributos do jogador

Nesta seção, liste os atributos individuais dos jogadores para incluir nas solicitações de matchmaking. Há duas razões pelas quais você pode declarar os atributos do jogador em um conjunto de regras:

- Quando o conjunto de regras contém regras que dependem dos atributos do jogador.
- Quando você quiser passar um atributo de jogador para a sessão do jogo por meio da solicitação de partida. Por exemplo, você pode querer passar as escolhas de personagens do jogador para a sessão do jogo antes que cada jogador se conecte.

Ao declarar um atributo de jogador, inclua as seguintes informações:

- nome(obrigatório) — este valor deve ser exclusivo para o conjunto de regras.
- tipo(obrigatório) — o tipo de dados do valor do atributo. Os tipos de dados válidos são número, string ou mapa de string.
- padrão(opcional) - insira um valor padrão para usar, se uma solicitação de marcação de jogos não fornecer um valor de atributo. Se nenhum padrão for declarado e uma solicitação não incluir um valor, FlexMatchnão consigo atender à solicitação.

Defina as equipes da partida

Descreva a estrutura e o tamanho das equipes de uma correspondência. Cada correspondência deve ter pelo menos uma equipe, e você pode definir quantas equipes quiser. Suas equipes podem ter o mesmo número de jogadores ou ser assimétricas. Por exemplo, você pode definir um time de monstros de jogadores únicos e uma equipe de caçadores com 10 jogadores.

FlexMatch O processa as solicitações de correspondência pequenas e grandes com base na definição do conjunto de regras sobre o tamanhos das equipes. As correspondências em potencial de até 40 jogadores são pequenas, as com mais de 40 são grandes. Para determinar um conjunto de regras da correspondência em potencial, adicione as configurações maxPlayer para todas as equipes definidas no conjunto de regras.

- nome(obrigatório) — atribua um nome exclusivo à cada equipe. Você usa esse nome em regras e expansões e FlexMatchreferências para os dados de matchmaking em uma sessão de jogo.
- MaxPlayers(obrigatório) - Especifique o número máximo de jogadores atribuídos ao grupo.
- minPlayers(obrigatório) - Especifique o número mínimo de jogadores a serem atribuídos ao grupo.
- quantidade(opcional) — Especifique o número de equipes a serem formadas com essa definição. Quando FlexMatch cria uma correspondência, dá a essas equipes o nome fornecido com um número anexado. Por exemplo`Red-Team_1`,`Red-Team_2`, e`Red-Team_3`.

FlexMatch tenta preencher as equipes com o máximo de jogadores, mas cria equipes com um número menor. Se você deseja que todas as equipes da correspondência sejam do mesmo tamanho, crie uma regra para isso. Consulte oExemplos de conjunto de regras FlexMatch (p. 26)tópico para um exemplo de um`EqualTeamSizes`regra.

Defina regras para combinar jogadores

Crie um conjunto de instruções de regras que avaliem os jogadores para aceitação em uma correspondência. As regras podem definir os requisitos que se aplicam a jogadores individuais, a equipes ou a uma correspondência inteira. Quando GameLift O processa uma solicitação de correspondência, ele começa com o jogador mais antigo no grupo de jogadores disponíveis e cria uma correspondência para cada jogador. Para obter ajuda detalhada sobre a criação FlexMatch regras, vejaFlexMatch Tipos de regras (p. 67).

- nome(obrigatório) — um nome significativo que identifica exclusivamente a regra dentro de um conjunto de regras. Os nomes de regra também são referenciados em logs de eventos e métricas que controlam as atividades relacionadas a essa regra.
- descrição(opcional) - use este elemento para anexar uma descrição de texto livre.
- tipo(obrigatório) — o elemento tipo identifica uma operação ser usada ao processar a regra. Cada tipo de regra exige um conjunto de propriedades adicionais. Veja uma lista dos tipos de regra válidos e as propriedades em Linguagem do FlexMatch (p. 60).
- Propriedade do tipo de regra (pode ser necessária) — Dependendo do tipo de regra definida, pode ser preciso definir determinadas propriedades. Saiba mais sobre as propriedades e como usar o FlexMatch linguagem de expressão de propriedade emLinguagem do FlexMatch (p. 60).

Deixe que os requisitos relaxem com o tempo

As expansões permitem que você diminua os critérios das regras ao longo do tempo quando FlexMatch não consigo encontrar uma combinação. Esse recurso garante que FlexMatch disponibiliza o melhor quando não consegue fazer uma combinação perfeita. Ao diminuir as regras com uma expansão, você expande gradualmente o grupo de jogadores que são aceitáveis.

As expansões começam quando a idade do tíquete mais novo na partida incompleta coincide com o tempo de espera de expansão. Quando FlexMatch adiciona um novo tíquete à partida, o relógio do tempo de espera da expansão pode ser redefinido. Você pode personalizar a forma como as expansões começam na`algorithm`seção do conjunto de regras.

Aqui está um exemplo de uma expansão que aumenta gradualmente o nível mínimo de habilidade necessário para a partida. O conjunto de regras usa uma declaração de regra de distância, chamadaSkillDeltaexigir que todos os jogadores em uma correspondência estejam dentro de 5 níveis de habilidades em relação ao outro. Se nenhuma nova partida for feita por quinze segundos, essa expansão procurará uma diferença de nível de habilidade de 10 e, dez segundos depois, procurará uma diferença de 20.

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

Com marcadores de jogos que têm a alocação automática ativada, não diminua os requisitos de contagem de jogadores muito rapidamente. Leva alguns segundos para a nova sessão de jogo iniciar e começar a alocação automática. Uma abordagem melhor é iniciar a expansão depois que as tendências de alocação iniciarem para seus jogos. O tempo de expansão varia de acordo com a composição da equipe, então faça testes para encontrar a melhor estratégia de expansão para o seu jogo.

Projetar um FlexMatch conjunto de regras de grande correspondência

Se seu conjunto de regras criar partidas que permitam de 41 a 200 jogadores, você precisará fazer alguns ajustes na configuração do conjunto de regras. Esses ajustes otimizam o algoritmo de partida para que ele possa criar grandes partidas viáveis e, ao mesmo tempo, manter os tempos de espera dos jogadores curtos. Como resultado, grandes conjuntos de regras de correspondência substituem regras personalizadas demoradas por soluções padrão que são otimizadas para prioridades comuns de matchmaking.

Veja como determinar se você precisa otimizar seu conjunto de regras para partidas grandes:

1. Para cada equipe definida em seu conjunto de regras, obtenha o valor demaxPlayer,
2. Some todos osmaxPlayervalores. Se o total exceder 40, você tem um conjunto de regras de correspondência grande.

Para otimizar seu conjunto de regras para partidas grandes, faça os ajustes descritos a seguir. Consulte o esquema de um conjunto de regras de correspondências grande emEsquema de conjunto de regras para correspondências grandes (p. 62)e exemplos de conjuntos de regras emExemplo 7: Crie uma correspondência grande (p. 36).

Personalize o algoritmo de correspondência para partidas grandes

Adicione um componente de algoritmo para o conjunto de regras, se ainda não existir. Defina as seguintes propriedades.

- `strategy`(obrigatório) — Defina `o`strategypropriedade para “equilibrada”. Essa configuração é acionada FlexMatch para fazer verificações adicionais após a partida para encontrar o equilíbrio ideal da equipe com base em um atributo de jogador especificado, que é definido no`balancedAttribute`propriedade. A estratégia equilibrada substitui a necessidade de regras personalizadas para criar equipes iguais.
- `balancedAttribute`(obrigatório) — Identifique um atributo de jogador para usar ao equilibrar as equipes em uma correspondência. Esse atributo deve ter um tipo de dado numérico (duplo ou inteiro). Por exemplo, se você optar por equilibrar a habilidade do jogador, FlexMatch tenta designar jogadores para que todas as equipes tenham níveis de habilidade agregados que sejam tão iguais quanto possível. O atributo de balanceamento deve ser declarado nos atributos do jogador do conjunto de regras.
- `batchingPreference`(opcional) — Escolha quanta ênfase você deseja colocar na formação das partidas de menor latência possível para seus jogadores. Essa configuração afeta a forma como os tíquetes das partidas são classificados antes da construção das partidas. Entre as opções estão:
  - População maior. FlexMatch permite correspondências usando todos os tickets do grupo que têm valores de latência aceitáveis em pelo menos uma região em comum. Como resultado, o potencial pool de ingressos tende a ser grande, o que facilita o preenchimento de fósforos mais rapidamente. Os jogadores podem ser colocados em jogos com latência aceitável, mas nem sempre ótima. Se o`batchingPreference`A propriedade não está definida, este é o comportamento padrão quando`strategy`está definido como “equilibrado”.
  - Região mais rápida. FlexMatch pré-classifica todos os tickets no pool com base em onde eles relatam os menores valores de latência. Como resultado, as correspondências tendem a ser formadas com jogadores que relatam baixa latência nas mesmas regiões. Ao mesmo tempo, o potencial pool de ingressos para cada partida é menor, o que pode aumentar o tempo necessário para preencher uma partida. Além disso, como uma prioridade maior é atribuída à latência, os jogadores nas correspondências podem variar mais amplamente em relação ao atributo de balanceamento.

O exemplo a seguir configura o algoritmo de correspondência para se comportar da seguinte maneira: (1) Pré-classifique o pool de tíquetes para agrupar os tíquetes por região, onde eles tenham valores de

latência aceitáveis; (2) Forme lotes de tíquetes classificados para combinar; (3) Crie partidas com tíquetes em um lote e equilibre as equipes para equilibrar a habilidade média do jogador.

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

Declarar os atributos do jogador

Certifique-se de declarar o atributo de jogador usado como atributo de balanceamento de algoritmo do conjunto de regras. Esse atributo deve ser incluído para cada jogador em uma solicitação de matchmaking. Você pode fornecer um valor padrão para o atributo do jogador, mas o balanceamento de atributos funciona melhor quando valores específicos do jogador são fornecidos.

Definir equipes

O processo de definição do tamanho da equipe e da estrutura é o mesmo que o de pequenas correspondências, mas a maneira como FlexMatch preenche as equipes é diferente. Isso afeta como as correspondências serão quando estiverem somente parcialmente preenchidas. Você pode ajustar o tamanho mínimo da equipe em resposta.

FlexMatch O usa as seguintes regras ao atribuir um jogador a uma equipe. Primeiro: procura equipes que ainda não alcançaram o requisito mínimo de jogadores. Depois: dessas equipes, procura a que tem mais slots abertos.

Para várias correspondências que definem o mesmo tamanho, os jogadores são adicionados sequencialmente para cada equipe até encher. Como resultado, as equipes em uma correspondência sempre têm um número quase igual de jogadores, mesmo quando a correspondência não está cheia. Atualmente, não há como forçar o mesmo tamanho em correspondências grandes. Para correspondências com equipes de tamanho assimétrico, o processo é um pouco mais complexo. Nesse cenário, os jogadores são atribuídos inicialmente às equipes maiores que têm a maioria dos slots abertos. Conforme o número de slots abertos se tornam mais uniformemente distribuído em todas as equipes, os jogadores são alocados nas equipes menores.

Por exemplo, digamos que você tenha um conjunto de regras com três equipes. As equipes vermelha e azul estão definidas com`maxPlayers=10`,`minPlayers=5`. A equipe verde está definida para`maxPlayers=3`,`minPlayers=2`. Aqui está a sequência de preenchimento:

1. Nenhuma equipe atingiu`minPlayers`. As equipes vermelha e azul têm 10 slots abertos, enquanto a verde tem 3. Os primeiros 10 jogadores são atribuídos (5 cada) para as equipes vermelha e azul. Ambas as as as as equipes atingiram`minPlayers`.
2. A equipe verde ainda não atingiu`minPlayers`. Os próximos dois jogadores são atribuídos à equipe verde. A equipe verde já chegou`minPlayers`.
3. Com todas as equipes em`minPlayers`, os jogadores adicionais agora são atribuídos com base no número de slots abertos. Cada uma das equipes vermelha e azul tem 5 vagas abertas, enquanto a equipe verde tem 1. Os próximos 8 jogadores são atribuídos (4 cada) para as equipes vermelha e azul. Todas as equipes agora têm 1 vaga aberta.
4. As três slots de jogador restantes são atribuídas (1 para cada) para as equipes sem uma ordem específica.

Estabeleça regras para partidas grandes

O matchmaking para partidas grandes depende principalmente da estratégia de balanceamento e das otimizações de latência em lotes. A maioria das regras personalizadas não estão disponíveis. No entanto, você pode incorporar os seguintes tipos de regras:

- Regra que define um limite rígido na latência do jogador. Usar `latency`tipo de regra com a propriedade`maxLatency`. ConsulteRegra de latência (p. 71)referência. Aqui está um exemplo que define a latência de jogador máximo de 200 milissegundos:

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

- Regra para agrupar jogadores com base na proximidade em um atributo de jogador especificado. Isso é diferente de definir um atributo de balanceamento como parte do algoritmo de grande correspondência, que se concentra na criação de equipes iguais. Esta regra agrupa os tíquetes de matchmaking com base na semelhança nos valores de atributos especificados, como habilidade de iniciante ou especialista, o que tende a levar a partidas de jogadores que estão estreitamente alinhados com o atributo especificado. Usar a`batchDistance`tipo de regra, identifique um atributo com base numérica e especifique a faixa mais ampla a ser permitida. ConsulteRegra de distância da (p. 67)referência. Aqui está um exemplo que exige que os jogadores de uma partida estejam dentro de um nível de habilidade um do outro:

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

Relaxe os requisitos de partida

Assim como ocorre com as pequenas correspondências, você pode usar as expansões para diminuir os requisitos de correspondência ao longo do tempo, quando não há correspondências válidas possíveis. Nas correspondências grandes, você tem a opção de diminuir as regras de latência do jogador ou as contagens.

Se você estiver usando alocação de correspondência automática para correspondências grandes, evite diminuir a contagem muito rapidamente. FlexMatch O começa a gerar as solicitações de alocação somente depois que uma correspondência é criada. Durante esse tempo, FlexMatch O pode criar várias novas sessões de jogos preenchidas, principalmente quando as regras de contagem são diminuídas. Como resultado, você obtém mais sessões de jogos do que precisa e jogadores em todas elas. A prática recomendada é permitir mais tempo para a primeira contagem de jogadores, suficiente para a sessão ser iniciada. Uma vez que as solicitações de alocação de prioridade mais alta são fornecidas com grandes correspondências, os jogadores recebidos serão inscritos em jogos existentes antes de novo jogo iniciar. Talvez você precise experimentar para localizar o tempo de espera ideal para seu jogo.

Aqui está um exemplo que gradualmente reduz a contagem de jogadores da equipe amarela, com um tempo de espera inicial mais longo. Lembre-se de que os tempos de nas expansões do conjunto de regras são absolutos, e não compostos. Portanto, a primeira expansão ocorre em cinco segundos, e a segunda ocorre cinco segundos mais tarde, em dez segundos.

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

Criar conjuntos de regras de marcação de

Antes de criar um conjunto de regras de marcação de jogos para o GameLift FlexMatch casamenteiro, recomendamos verificar osintaxe do conjunto de regras (p. 60). Depois de criar um conjunto de regras usando o GameLift Console doAWS Command Line Interface(AWS CLI), você não pode alterá-lo.

Observe que há umcota de serviçopara o número máximo de conjuntos de regras que você pode ter em umAWSPor isso, é uma boa ideia excluir os conjuntos de regras não utilizados.

Tópicos relacionados

- Projetar um FlexMatch conjunto de regras (p. 16)
- Exemplos de conjunto de regras FlexMatch (p. 26)
- Linguagem do FlexMatch (p. 60)

Console

Criar um conjunto de regras

1. Abrir o GameLift Console dohttps://console.aws.amazon.com/gamelift/.
2. Alterne para oAWSRegião onde você deseja criar seu conjunto de regras. Defina os conjuntos de regras na mesma região da configuração de marcação de jogos que os utiliza.
3. No painel de navegação, selecioneFlexMatch,Conjuntos de regras da marcação.
4. NoConjuntos de regras da marcaçãopágina, selecioneCriar conjunto de regras.
5. NoCriar um conjunto de regras de marcação, faça o seguinte:
   a. UnderConfigurações de conjunto de regras, paraName (Nome), insira um nome descritivo exclusivo que você pode usar para identificá-lo em uma lista ou em tabelas de eventos e métricas.
   b. para oConjunto de regras, insira seu conjunto de regras em JSON. Para obter informações sobre como criar um conjunto de regras, consulteProjetar um FlexMatch conjunto de regras (p. 16). Você também pode usar um dos conjuntos de regras de exemplo doExemplos de conjunto de regras FlexMatch (p. 26).
   c. SelecioneValidarpara verificar se a sintaxe do conjunto de regras está correta. Não é possível editar os conjuntos de regras após a criação, por isso é uma boa ideia validá-los primeiro.
   d. (Opcional) EmTags, adicione tags para ajudá-lo a gerenciar e rastrearAWSrecursos da AWS.
6. Escolha Create (Criar). Se a criação for bem-sucedida, você poderá usar o conjunto de regras com um marcador de jogos.

AWS CLI

Criar um conjunto de regras

Abra uma janela de linha de comando e use o comandocreate-matchmaking-rule-set.

Este comando de exemplo cria um conjunto de regras de marcação de jogos simples que define uma única equipe. Certifique-se de criar o conjunto de regras no mesmoAWSRegião como as configurações de combinação que a usa.

```
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}]}'
```

Se a solicitação de criação for bem-sucedida, o GameLift O retorna umMatchmakingRuleSetobjeto que inclui as configurações especificadas. Um matchmaker agora pode usar o novo conjunto de regras.

Console

Excluir um conjunto de regras

1. Abrir o GameLift Console dohttps://console.aws.amazon.com/gamelift/.
2. Alterne para a Região na qual você criou o conjunto de regras.
3. No painel de navegação, selecioneFlexMatch,Conjuntos de regras da marcação.
4. NoConjuntos de regras da marcação, selecione o conjunto de regras que você deseja excluir e, em seguida, selecioneExcluir.
5. NoExcluir conjunto de regras, selecioneExcluirpara confirmar a exclusão.

   Note

   Se uma configuração de combinação estiver usando o conjunto de regras, GameLiftexibe uma mensagem de erro (Não é possível excluir o conjunto de regras). Se isso ocorrer, altere a configuração de marcação de jogos para usar um conjunto de regras diferente, então tente novamente. Para descobrir quais configurações de marcação de jogos estão usando um conjunto de regras, selecione o nome de um conjunto de regras para visualizar a página de detalhes.

AWS CLI

Excluir um conjunto de regras

Abra uma janela de linha de comando e use o comandodelete-matchmaking-rule-setpara excluir um conjunto de regras de marcação de jogos.

Se uma configuração de combinação estiver usando o conjunto de regras, GameLift O retorna uma mensagem de erro. Se isso ocorrer, altere a configuração de marcação de jogos para usar um conjunto de regras diferente, então tente novamente. Para obter uma lista das configurações de marcação de jogos que estão usando um conjunto de regras, use o comando dodescribe-matchmaking-configurationse especifique o nome do conjunto de regras.

Este comando de exemplo verifica o uso do conjunto de regras de marcação de jogos e, depois, exclui o conjunto de regras.

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

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

Exemplos de conjunto de regras FlexMatch

Os conjuntos de regras do FlexMatch cobrem uma variedade de cenários de marcação de jogos. Os exemplos a seguir estão em conformidade com a estrutura de configuração do FlexMatch e o idioma de expressão de propriedade. Copie esses conjuntos de regras em sua totalidade ou escolha componentes, se necessário.

Para obter mais informações sobre o uso e conjuntos de regras do FlexMatch, consulte os tópicos a seguir:

- Criar um conjunto de regras do FlexMatch (p. 16)
- Projetar um FlexMatch conjunto de regras (p. 16)
- Esquema conjunto de regras FlexMatch (p. 60)
- Linguagem do FlexMatch (p. 60)

> **Note**
>
> Ao avaliar um tíquete de marcação de jogos que inclui vários jogadores, todos na solicitação precisam atender às exigências da correspondência.

Exemplo 1: Crie duas equipes com jogadores uniformemente combinados

Este exemplo ilustra como configurar duas equipes com correspondência igual de jogadores com as instruções a seguir.

- Crie duas equipes de jogadores.
  - Inclua entre quatro e oito jogadores em cada equipe.
  - As equipes finais devem ter o mesmo número de jogadores.
- Inclua um nível de habilidade do jogador (se não for fornecido, o padrão é 10).
- Escolha jogadores baseado em se o nível de habilidade deles é semelhante a outros jogadores. Verifique se ambas as equipes têm um nível de habilidade médio por jogador de 10 pontos entre si.
- Se a correspondência não for preenchida rapidamente, atenue a exigência de habilidade do jogador para concluir uma correspondência em tempo razoável.
  - Depois de 5 segundos, expanda a pesquisa para permitir equipes com habilidades de jogador médias de 50 pontos.
  - Depois de 15 segundos, expanda a pesquisa para permitir equipes com habilidades de jogador médias de 100 pontos.

Observações sobre como usar o conjunto de regras:

- Este exemplo permite equipes de qualquer tamanho entre quatro e oito jogadores (embora elas precisem ser do mesmo tamanho). Para equipes com uma faixa de tamanhos válida, o marcador de jogos faz uma tentativa de melhor esforço para corresponder ao número máximo de jogadores permitidos.
- A regra `FairTeamSkill` garante que as equipes sejam uniformemente correlacionadas com base na habilidade do jogador. Para avaliar essa regra para cada novo jogador em potencial, o FlexMatch adiciona o jogador a uma equipe provisoriamente e calcula as médias. Se a regra falhar, o jogador em potencial não será adicionado à correspondência.
- Como ambas as equipes têm estruturas idênticas, você pode optar por criar apenas uma definição de equipe e definir a quantidade da equipe como "2". Nesse cenário, se você nomeou a equipe como "aliens", suas equipes recebem os nomes "aliens_1" e "aliens_2".

```
{
    "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
        }]
    }]
}
```

Exemplo 2: Crie equipes irregulares (caçadores x monstro)

Este exemplo descreve um modo de jogo em que um grupo de jogadores caça um único monstro. As pessoas escolhem a função de um caçador ou de um monstro. Os caçadores especificam o nível de habilidade mínimo para o monstro que eles querem enfrentar. O tamanho mínimo da equipe do caçador pode ser atenuado ao longo do tempo para concluir a correspondência. Este cenário define as instruções a seguir:

- Crie uma equipe de exatamente cinco caçadores.
- Crie uma equipe separada de exatamente um monstro.
- Inclua os atributos de jogador a seguir:
  - O nível de habilidade de um jogador (se não for fornecido, o padrão é 10).
  - O nível de habilidade preferido do monstro de um jogador (se não for fornecido, o padrão é 10).
  - Se o jogador deseja ser o monstro (se não for fornecido, o padrão é 0 ou falso).
- Escolha um jogador para ser o monstro com base nos seguintes critérios:
  - O jogador deve solicitar a função do monstro.
  - O jogador deve atender ou exceder o nível mais alto de habilidade preferido pelos jogadores que já foram adicionados à equipe do caçador.
- Escolha jogadores para a equipe do caçador com base nos seguintes critérios:

  - Os jogadores que solicitam a função do monstro não podem entrar na equipe do caçador.
  - Se a função do monstro já tiver sido preenchida, o jogador poderá querer um nível de habilidade de monstro que seja inferior à habilidade do monstro proposto.
- Se uma correspondência não for preenchida rapidamente, atenue o tamanho mínimo da equipe do caçador da seguinte forma:
  - Após 30 segundos, permita que um jogo inicie com apenas quatro jogadores na equipe do caçador.
  - Após 60 segundos, permita que um jogo inicie com apenas três pessoas na equipe do caçador.

Observações sobre como usar o conjunto de regras:

- Usando duas equipes separadas para caçadores e monstro, você pode avaliar a associação com base em conjuntos diferentes de critérios.

```
{
    "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
        }]
    }]
}
```

Exemplo 3: Definir requisitos de nível de equipe e limites de latência

Este exemplo ilustra como configurar as equipes de jogadores e aplicar um conjunto de regras a cada equipe em vez de cada jogador individual. Ele usa uma única definição para criar três equipes correspondentes iguais. Ele também estabelece uma latência máxima para todos os jogadores. Os máximos de latência podem ser atenuados ao longo do tempo para concluir a correspondência. Este cenário define as instruções a seguir:

- Crie três equipes de jogadores.
  - Inclua entre três e cinco jogadores em cada equipe.
  - As equipes finais devem conter aproximadamente ou o mesmo número de jogadores (em uma equipe).
- Inclua os atributos de jogador a seguir:
  - O nível de habilidade de um jogador (se não for fornecido, o padrão é 10).
  - A função do personagem de um jogador (se não for fornecida, o padrão é "camponês").
- Escolha jogadores com base em se o nível de habilidade deles é semelhante a outros jogadores na correspondência.
  - Verifique se cada equipe tem uma habilidade de jogador média de 10 pontos entre si.
- Limite as equipes ao número seguinte de personagens “médicos”:
  - Uma correspondência inteira pode ter um máximo de cinco médicos.
- Somente faz a correspondência de jogadores que mostram latência de até 50 milissegundos.
- Se uma correspondência não for preenchida rapidamente, atenue a exigência de latência de jogador, da seguinte forma:
  - Após 10 segundos, permita valores de latência de jogador de até 100 ms.
  - Após 20 segundos, permita valores de latência de jogador de até 150 ms.

Observações sobre como usar o conjunto de regras:

- O conjunto de regras garante que as equipes sejam uniformemente correspondidas com base na habilidade do jogador. Como avaliar o`FairTeamSkill`Regra, o FlexMatch adiciona o jogador em potencial a uma equipe e calcula a habilidade média dos jogadores na equipe. Depois, ele compara com a habilidade média nas duas equipes. Se a regra falhar, o jogador em potencial não será adicionado à correspondência.
- As exigências de nível de correspondência e da equipe (número total de médicos) são atendidas por meio de uma regra de coleção. Esse tipo de regra usa uma lista de atributos de personagem para todos os jogadores e verifica em relação às contagens máximas. Use `flatten` para criar uma lista de todos os jogadores em todas as equipes.
- Ao avaliar com base na latência, observe o seguinte:

- Os dados de latência são fornecidos na solicitação de marcação de jogos como parte do objeto Player. Não se trata de um atributo de jogador; portanto, não é preciso ser listado como tal.
- O marcador de jogos avalia a latência de acordo com a região. Qualquer região com uma latência mais alta do que o máximo é ignorada. Para ser aceito para uma correspondência, um jogador deve ter pelo menos uma região com uma latência abaixo do máximo.
- Se uma solicitação de marcação de jogos omitir dados de latência de um ou mais jogadores, ela será rejeitada em todas as partidas.

```
{
    "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
        }]
    }]
}
```

Exemplo 4: Use a classificação explícita para encontrar as melhores correspondências

Este exemplo configura uma correspondência simples com duas equipes de três jogadores. Ele ilustra como usar as regras de classificação explícitas para ajudar a encontrar as melhores correspondências o mais rápido possível. Essas regras classificam todos os tíquetes de matchmaking ativos para criar as melhores correspondências com base em determinados requisitos principais. Este cenário é implementado com as seguintes instruções:

- Crie duas equipes de jogadores.
- Inclua exatamente três jogadores em cada equipe.
- Inclua os atributos de jogador a seguir:
  - Nível de experiência (se não for fornecido, o padrão é 50).
  - Modos de jogo preferidos (pode listar vários valores) (se não forem fornecidos, os padrões são “cooperação” e “combate mortal”).
  - Mapas de jogo preferidos, incluindo o nome do mapa e ponderação preferida (se não for fornecido, o padrão é `"defaultMap"` com um peso 100).
- Configurar pré-classificação:
  - Classifique os jogadores com base na preferência pelo mesmo mapa de jogo que o jogador âncora. Os jogadores podem ter vários mapas de jogo favoritos, portanto, este exemplo usa um valor preferencial.
  - Classifique os jogadores com base na proximidade do nível de experiência deles com a do jogador âncora. Com essa classificação, todos os jogadores em todas as equipes terão níveis de experiência o mais próximos possíveis.
- Todos os jogadores em todas as equipes precisam ter selecionado pelo menos um modo de jogo em comum.
- Todos os jogadores em todas as equipes precisam ter selecionado pelo menos um mapa de jogo em comum.

Observações sobre como usar o conjunto de regras:

- A classificação do mapa de jogo usa uma classificação absoluta que compara o valor do atributo mapPreference. Como é o primeiro no conjunto de regras, esse tipo é realizado primeiro.
- A classificação da experiência usa uma classificação de distância para comparar um nível de habilidade do jogador em potencial com a habilidade do jogador âncora.
- As classificações são feitas na ordem que são listadas em um conjunto de regras. Neste cenário, os jogadores são classificados pela preferência de mapa de jogo e depois pelo nível de experiência.

```
{
    "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
    }]
}
```

Exemplo 5: Encontre interseções em vários atributos de jogador

Este exemplo ilustra como usar uma regra de coleta para encontrar interseções em dois ou mais atributos do jogador. Ao trabalhar com coleções, você pode usar a operação `intersection` para um único atributo e a operação `reference_intersection_count` para vários atributos.

Para ilustrar essa abordagem, o exemplo avalia os jogadores em uma partida com base nas preferências dos seus personagens. O jogo de exemplo é um estilo "gratuito para todos", em que todos os jogadores em uma partida são oponentes. Cada jogador deve (1) escolher um personagem para si e (2) escolher os personagens adversários. Precisamos de uma regra que garanta que todos os jogadores em uma partida estejam usando um personagem na lista de oponentes preferidos de todos os outros jogadores.

O conjunto de regras de exemplo descreve uma correspondência com as seguintes características:

- Estrutura da equipe: Uma equipe de cinco jogadores
- Atributos do jogador:
  - MyCharacter: O personagem escolhido pelo jogador.
  - preferredOpponents: Lista de personagens contra os quais o jogador quer jogar.
- Regras de correspondência: Uma correspondência potencial é aceitável se cada personagem em uso estiver na lista de oponentes preferidos de todos os jogadores.

Para implementar a regra de correspondência, este exemplo usa uma regra de coleta com os seguintes valores de propriedade:

- Operação — Usos`reference_intersection_count`Operação para avaliar como cada lista de strings no valor de medição intercepta a lista de strings no valor de referência.
- Medição — Usa `o`flattenExpressão de propriedade para criar uma lista de listas de strings, cada uma delas contendo um jogadorMyCharacterValor de atributo.
- Valor de referência — Usa `o`set_intersectionexpressão de propriedade para criar uma lista de cadeias de caracteres de todospreferredOpponentsValores de atributo comuns a todos os jogadores na correspondência.
- Restrições —`minCount`É definido como 1 para garantir que o personagem escolhido de cada jogador (uma lista strings na medição) corresponda a pelo menos um dos oponentes preferidos comuns a todos os jogadores. (uma string no valor de referência).
- Expansão — se uma correspondência não for feita em 15 segundos, ajuste o requisito mínimo de interseção.

O fluxo do processo para esta regra é o seguinte:

1. Um jogador é adicionado ao jogo em potencial. O valor de referência (uma lista de strings) é recalculado para incluir interseções com a lista de oponentes preferidos do novo jogador. O valor de medição (uma lista das listas de strings) é recalculado para adicionar o caractere escolhido do novo jogador como uma nova lista de strings.
2. O Amazon GameLift verifica se cada lista de strings no valor de medição (caracteres escolhidos pelos jogadores) faz intersecção com pelo menos uma string no valor de referência (oponentes preferidos dos jogadores). Neste exemplo, como cada lista de strings na medição contém apenas um valor, a interseção é 0 ou 1.
3. Se alguma lista de strings na medição não fizer interseção com a lista de strings dos valores de referência, a regra falhará e o novo jogador será removido da correspondência em potencial.

4. Se uma partida não for preenchida em 15 segundos, elimine o requisito de correspondência do oponente para preencher os slots restantes do jogador na partida.

```
{
    "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
        }]
    }]
}
```

Exemplo 6: Compare atributos em todos os jogadores

Este exemplo ilustra como comparar os atributos do jogador em um grupo de jogadores.

O conjunto de regras de exemplo descreve uma correspondência com as seguintes características:

- Estrutura da equipe: Duas equipes de jogador único
- Atributos do jogador:
  - gameMode: Tipo de jogo escolhido pelo jogador (se não for fornecido, o valor padrão será “turn-based”).
  - gameMap: Mundo do jogo escolhido pelo jogador (se não for fornecido, o valor padrão será 1).
  - personagem: Personagem escolhido pelo jogador (se não houver nenhum valor padrão significa que os jogadores precisarão especificar um personagem).
- Regras de correspondência: Os jogadores correspondentes precisam atender aos seguintes requisitos:
  - Os jogadores devem escolher o mesmo modo de jogo.
  - Os jogadores devem escolher o mesmo mapa de jogo.
  - Os jogadores precisam escolher personagens diferentes.

Observações sobre como usar o conjunto de regras:

- Para implementar a regra de correspondência, este exemplo usa regras de comparação de modo a verificar os valores de atributos de todos os jogadores. Para modo de jogo e mapa, a regra verifica se os valores são os mesmos. Para caractere, a regra verifica se os valores são diferentes.
- Este exemplo usa uma definição de jogador com uma propriedade de quantidade para criar as duas equipes de jogadores. Eles recebem os nomes: "jogador_1" e "jogador_2".

```
{
    "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])"]
    }]
}
```

Exemplo 7: Crie uma correspondência grande

Este exemplo ilustra como configurar um conjunto de regras para correspondências que podem exceder 40 jogadores. Quando um conjunto de regras descreve as equipes com uma contagem total de maxPlayer maior que 40, ele é processado como uma correspondência grande. Saiba mais em Projetar um FlexMatch conjunto de regras de grande correspondência (p. 22).

O exemplo cria um conjunto de regras de correspondências usando as seguintes instruções:

- Crie uma equipe com até 200 jogadores, com um requisito mínimo de 175 jogadores.
- Os critérios de balanceamento: Selecione jogadores com base no nível de habilidade semelhantes. Todos os jogadores devem informar o nível de habilidade deles para a correspondência ser feita.
- Preferência de lotes: Agrupe jogadores por critérios de balanceamento semelhantes ao criar correspondências.
- Regras de latência: Defina a latência máxima aceitável do jogador como 150 milissegundos.
- Se a correspondência não for preenchida rapidamente, atenue a exigência para concluir uma correspondência em tempo razoável.
  - Após 10 segundos, aceite uma equipe com 150 jogadores.
  - Após 12 segundos, eleve a latência aceitável para 200 milissegundos.
  - Depois de 15 segundos, aceite uma equipe com 100 jogadores.

Observações sobre como usar o conjunto de regras:

- Como o algoritmo usa a preferência de processamento em grupos "maior população" primeiro, os jogadores são classificados com base nos critérios de balanceamento. Como resultado, as correspondências tendem a ser mais completas e conter jogadores mais semelhantes em relação à habilidade. Todos os jogadores atendem aos requisitos de latência aceitáveis, mas podem não ter a melhor latência possível para sua localização.
- A estratégia de algoritmo usada neste conjunto de regras, "maior população", é a configuração padrão. Para usar o padrão, você pode optar por omitir a configuração.
- Se você tiver habilitado a alocação de correspondência, não diminua os requisitos de contagem de jogadores muito rapidamente, ou você pode acabar com muitas sessões de jogos parcialmente preenchidas. Saiba mais em Relaxe os requisitos de partida (p. 24).

```
{
    "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
        }]
    }]
}
```

Exemplo 8: Crie uma partida grande com várias equipes

Este exemplo ilustra como configurar um conjunto de regras para correspondências de várias equipes que podem exceder 40 jogadores. Este exemplo ilustra como criar várias equipes idênticas com uma definição e como as equipes de tamanho assimétrico são preenchidas durante a criação de correspondência.

O exemplo cria um conjunto de regras de correspondências usando as seguintes instruções:

- Crie dez idênticas equipes idênticas de "caçadores" com até 15 jogadores, e uma equipe de "monstros" com exatamente 5 jogadores.
- Os critérios de balanceamento: Selecione jogadores com base no número de mortes de monstros. Se os jogadores não relatarem a contagem de mortes, use um valor padrão de 5.
- Preferência de lotes: Jogadores de grupo com base nas regiões em que relatam a latência mais rápida.
- Regra de latência: Define uma latência máxima aceitável do jogador como 200 milissegundos.
- Se a correspondência não for preenchida rapidamente, atenue a exigência para concluir uma correspondência em tempo razoável.
  - Depois de 15 segundos, aceite equipes com 10 jogadores.
  - Após 20 segundos, aceite equipes com 8 jogadores.

Observações sobre como usar o conjunto de regras:

- Este conjunto de regras define equipes que podem ter até 155 jogadores, o que faz a correspondência ser grande (10 x 15 caçadores + 5 monstros = 155)
- Como o algoritmo usa a preferência de agrupamento por "região mais rápida", os jogadores tendem a ser colocados em regiões onde mostram latência mais rápida e não em regiões de latência mas alta (mas é aceitável). Ao mesmo tempo, as correspondências podem ter um número menor de jogadores, e os critérios de balanceamento de monstros (número de habilidades) pode variar mais.
- Quando uma expansão é definida para uma definição de várias equipes (quantidade > 1), ela se aplica a todas as equipes criadas com essa definição. Portanto, diminuir o tamanho mínimo da equipe de caçadores faz com que todas as dez equipes de caçadores sejam impactadas igualmente.
- Como esse conjunto de regras é otimizado para minimizar a latência do jogador, a latência atua como uma regra genérica para excluir os jogadores que não têm opções de conexão aceitáveis. Não precisamos diminuir essa exigência.
- Veja como o FlexMatch preenche correspondências para esse conjunto de regras antes de qualquer expansão entrar em vigor:
  - Nenhuma equipe atingiu a contagem minPlayers até agora. As equipes de caçadores têm 15 slots abertos, enquanto a equipe de monstros tem 5 slots abertos.
    - Os primeiros 100 jogadores são atribuídos (10 cada) às 10 equipes de caçadores.
    - Os próximos 22 jogadores são atribuídos sequencialmente (2 para cada) às equipes de caçadores e à equipe de monstros.
  - As equipes de caçadores atingiram a contagem minPlayers de 12 jogadores cada. A equipe de monstros tem dois jogadores e não atingiu a contagem minPlayers.
    - Os próximos três jogadores são atribuídos à equipe de monstros.
  - Todas as equipes atingiram a contagem minPlayers. As equipes de caçadores têm três slots abertos cada uma. A equipe de monstros está cheia.

    - Os últimos 30 jogadores são atribuídos sequencialmente às equipes de caçadores, garantindo que todas as equipes têm praticamente o mesmo tamanho (um jogador a mais ou a menos).
- Se você tiver habilitado a alocação para correspondências criadas dentro desse conjunto de regras, não diminua os requisitos de contagem de jogadores muito rapidamente, ou você pode acabar com muitas sessões de jogos parcialmente preenchidas. Saiba mais em Relaxe os requisitos de partida (p. 24).

```
{
    "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
        }]
    }]
}
```

Exemplo 9: Crie uma partida grande com jogadores com atributos semelhantes

Este exemplo ilustra como configurar um conjunto de regras para correspondências de duas equipes usando`batchDistance`. No exemplo:

- O`SimilarLeague`Regra garante que todos os jogadores em uma correspondência tenham um`league`dentro de 2 de outros jogadores.
- O`SimilarSkill`Regra garante que todos os jogadores em uma correspondência tenham um`skill`dentro de 10 de outros jogadores. Se um jogador estiver esperando 10 segundos, a distância será expandida para 20. Se um jogador estiver esperando 20 segundos, a distância será expandida para 40.

- OSameMapRegra garante que todos os jogadores em uma correspondência tenham solicitado o mesmomap.
- OSameModeRegra garante que todos os jogadores em uma correspondência tenham solicitado o mesmomode.

```
{
    "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
        }]
    }]
```

```
}
```

Exemplo 10: Use uma regra composta para criar uma partida com jogadores com atributos semelhantes ou seleções semelhantes

Este exemplo ilustra como configurar um conjunto de regras para correspondências de duas equipes usando`compound`. No exemplo:

- O`SimilarLeagueDistance`Regra garante que todos os jogadores em uma correspondência tenham um`league`dentro de 2 de outros jogadores.
- O`SimilarSkillDistance`Regra garante que todos os jogadores em uma correspondência tenham um`skill`dentro de 10 de outros jogadores. Se um jogador estiver esperando 10 segundos, a distância será expandida para 20. Se um jogador estiver esperando 20 segundos, a distância será expandida para 40.
- O`SameMapComparison`Regra garante que todos os jogadores em uma correspondência tenham solicitado o mesmo`map`.
- O`SameModeComparison`Regra garante que todos os jogadores em uma correspondência tenham solicitado o mesmo`mode`.
- O`CompoundRuleMatchmaker`Uma regra garante uma correspondência se pelo menos uma das seguintes condições for true:
  - Jogadores em uma partida solicitaram o mesmo`map`E o mesmo`mode`.
  - Jogadores em uma partida têm comparável`skill`e`league`atributos.

```
{
    "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
        }]
    }]
}
```

Configurar FlexMatch Notificações de eventos do

Se você estiver usando GameLift FlexMatch Marcações de jogos do no seu jogo, você precisa de uma maneira de monitorar o status de solicitações de marcação de jogos individuais. Implementar notificações de eventos é um método rápido e eficiente para rastrear eventos de marcação de jogos. Todos os jogos em produção, ou em pré-produção com atividade de marcação de jogos de alto volume, devem usar notificações de eventos.

Há duas opções para a configuração de notificações de eventos. Você pode usar a Amazon CloudWatch Eventos, que tem um pacote de ferramentas disponíveis para gerenciar eventos e executar ações neles. Ou, é possível configurar seus próprios tópicos do Amazon Simple Notification Service (Amazon SNS) e configurar seu marcador de jogos para enviar notificações de eventos diretamente aos tópicos.

Para uma lista do FlexMatch eventos que GameLift emite, consulteEventos de marcação do FlexMatch (p. 75).

Configurar CloudWatch Eventos

GameLift O publica automaticamente todos os eventos de marcação de jogos CloudWatch Events (Eventos). com CloudWatch Eventos, você pode configurar regras para rotear os eventos de marcação de jogos para uma variedade de destinos, incluindo os tópicos do SNS e outrosAWSserviços para processamento. Por exemplo, é possível definir uma regra para rotear o evento”PotentialMatchCreated“para umAWS Lambdafunção que lida com as aceitações do jogador.

Para obter mais informações sobre como usar CloudWatch Eventos, incluindo uma coleção de tutoriais, consulteConceitos básicos da Amazon CloudWatch EventosnoAmazônia CloudWatch Guia do usuário do Eventos.

Se você planeja usar CloudWatch Eventos, ao configurar seus marcadores de jogos, será possível manter o campo de destino da notificação vazio, ou fazer referência a um tópico do SNS, se quiser usar ambas as opções.

Você pode acessar GameLift eventos de marcação de jogos CloudWatch Eventos do noCloudWatch console. Para obter mais informações, consulteFaça login na Amazon CloudWatch Console do. CloudWatch Eventos identifica cada evento de matchmaking pelo serviço (GameLift), o nome e o tíquete de marcação de jogos.

Configurar um tópico do Amazon SNS

Você pode ter GameLift publicar todos os eventos que um FlexMatch O marcador de jogos é gerado em um tópico do Amazon SNS.

Para criar um tópico do SNS para GameLift Notificações de eventos do

1. Abra o console do Amazon SNS.
2. No painel de navegação, escolha Topics (Tópicos).
3. Na página Topics (Tópicos), escolha Create topic (Criar tópico).
4. Crie um tópico no console do . Para obter mais informações, consulteComo criar um tópico usando oAWS Management ConsolenoGuia do desenvolvedor do Amazon Simple Notification Service.
5. NoDetalhespágina do tópico, escolhaEdite.
6. NoEditepágina do seu tópico, expandaPolítica de acesso -opcionaise, em seguida, adicione a sintaxe em negrito do seguinteAWS Identity and Access Management(IAM) até o final da política existente. (A política inteira é mostrada aqui para oferecer clareza.) Certifique-se de usar os detalhes do Nome de recurso da Amazon (ARN) no tópico do SNS e GameLift marcação de jogos do.

```
{
  "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_matchmaking_configuration_name"
        }
      }
    }
  ]
}
```

7. Escolha Save changes (Salvar alterações).

Configurar uma inscrição do tópico para uma função do Lambda

Você pode chamar uma função do Lambda usando notificações de eventos no tópico do Amazon SNS. Ao configurar o marcador de jogos, defina o destino da notificação no ARN do tópico do SNS.

Os seguintes exemplos deAWS CloudFormationconfigura uma assinatura para um tópico do SNS chamado`MyFlexMatchEventTopic`para chamar uma função do Lambda chamada`FlexMatchEventHandlerLambdaFunction`. O modelo cria uma política de permissões do IAM que permite GameLift para escrever no tópico do SNS. Por fim, ele adiciona permissões ao tópico do SNS para chamar a função do Lambda.

```
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
```

Preparando jogos para FlexMatch

Use o GameLift FlexMatch para adicionar funcionalidade de partida de jogadores aos seus jogos. O FlexMatch está disponível com as soluções gerenciadas do GameLift para servidores de jogos personalizados e servidores em tempo real.

O FlexMatch compara o serviço de marcação de jogos com o mecanismo de regras personalizável. Isso permite que você projete como combinar jogadores com base em seus atributos e modos de jogo de acordo com o seu jogo. Conte também com o FlexMatch para gerenciar os detalhes da formação de grupos de jogadores e colocá-los em jogos. Veja mais detalhes sobre a correspondência personalizada em Exemplos de conjunto de regras FlexMatch (p. 26).

O FlexMatch conta com o recurso Filas. Assim que uma correspondência é criada, o FlexMatch enviará os detalhes da combinação para uma fila da sua escolha. Em sua fila procura os recursos de hospedagem disponíveis em suas frotas do Amazon GameLift e inicia uma nova sessão de jogo para a correspondência.

Os tópicos nesta seção discutem como adicionar o suporte da marcação de jogos aos seus servidores e clientes de jogos. Para criar um marcador ao seu jogo, consulte Construindo um GameLift FlexMatch casamenteiro (p. 10). Para obter mais informações sobre o funcionamento do FlexMatch, consulteComo o Amazon GameLift FlexMatch funciona (p. 3).

Adicionar FlexMatch para um cliente de jogo

Este tópico descreve como adicionar o FlexMatch suporte de matchmaking para seus serviços de jogos do lado do cliente. O processo é essencialmente o mesmo se você estiver usando o FlexMatch com GameLift hospedagem gerenciada ou com outra solução de hospedagem. Para saber mais a respeito FlexMatch e como configurar um marcador de jogos personalizado para seus jogos, consulte os seguintes tópicos:

- Integração FlexMatch com hospedagem GameLift (p. 8)
- Como o Amazon GameLift FlexMatch funciona (p. 3)
- Construindo um GameLift FlexMatch casamenteiro (p. 10)
- Exemplos de conjunto de regras FlexMatch (p. 26)

Para habilitar o FlexMatch Marcação de jogos no seu jogo, adicione as seguintes funcionalidades:

- Prepare-se para solicitar marcação de jogos para um ou mais jogadores (obrigatório).
- Acompanhe o status das solicitações de marcação de jogos (obrigatório).
- Solicitar aceitação do jogador para uma correspondência proposta (opcional).
- Depois que uma sessão de jogo for criada para a nova correspondência, obtenha as informações sobre a conexão do jogador e ingresse no jogo.

Prepare-se para solicitar matchmaking para jogadores

É altamente recomendável que o cliente do jogo faça solicitações de marcação por meio de um serviço de jogo no lado do cliente. Ao usar uma fonte confiável, você pode proteger mais facilmente contra tentativas de invasão e dados de jogadores falsos. Se o seu jogo tem um serviço de diretório de sessão, esta é uma boa opção para lidar com as solicitações de marcação de jogos.

Para preparar seu serviço de cliente, execute as seguintes tarefas:

- Adicionar o GameLift API. Seu serviço ao cliente usa a funcionalidade no GameLift API, que faz parte doAWSSDK. ConsulteGameLift SDKs para serviços ao clientepara saber mais sobre oAWSSDK e download da versão mais recente. Adicione esse SDK ao seu projeto de serviço de cliente do jogo.
- Configure um sistema de tíquete de marcação de jogos. Todas as solicitações de marcação de jogos ter um ID de tíquete exclusivo. É necessário um mecanismo para gerar IDs exclusivos e atribui-los às novas solicitações de correspondência. Um ID de tíquete pode usar qualquer formato de string com até 128 caracteres.
- Obter informações sobre o marcador de jogos. Obtenha o nome da configuração de marcação de jogos que você planeja usar. Também é necessária a lista dos atributos de jogador exigidos do marcador de jogos, que estão definidos no conjunto de regras do marcador de jogos.
- Obter dados do jogador. Configure uma maneira de obter dados relevantes para cada jogador. Isso inclui o ID do jogador, os valores de atributo do jogador e os dados de latência atualizados para cada região em que o jogador provavelmente será colocado em um jogo.
- (opcional) Habilite a alocação de marcação. Decida como você quer preencher seus jogos correspondentes. Se seus marcadores de jogos estiverem com o modo de alocação definido como "manual", é recomendável adicionar o suporte de alocação ao seu jogo. Se seus marcadores de jogos estiverem com o modo de alocação definido como "automático", é recomendável desativar essa função nas sessões de jogos individuais. Saiba mais sobre o gerenciamento de correspondência de alocação em Enchimento de jogos existentes com FlexMatch (p. 53).

Solicitar marcação de jogos para jogadores

Adicione o código ao seu serviço de cliente para criar e gerenciar as solicitações de marcação de jogos para um marcador de jogos do FlexMatch. O processo de solicitação FlexMatch matchmaking é idêntico para jogos que usam FlexMatch com GameLiftmanaged hospedagem e para jogos que usam FlexMatch como uma solução autônoma.

Criar uma solicitação de marcação de jogos:

- Chame o GameLift APIStartMatchmaking. Cada solicitação deve conter as seguintes informações.

  Marcador de jogos

  O nome da configuração de marcação de jogos a ser usada para a solicitação. FlexMatch O insere cada solicitação no grupo do marcador de jogos especificado, e a solicitação é processada de acordo com a forma como o marcador está configurado. Isso inclui aplicar um limite de tempo, para solicitar a aceitação de correspondências de jogadores, que a fila usará ao criar uma sessão de jogo resultante, etc. Saiba mais sobre os marcadores de jogos e os conjuntos de regras em Projete um FlexMatch casamenteiro (p. 10).

  ID do tíquete

  Um ID de tíquete exclusivo atribuído à solicitação. Tudo relacionado à solicitação, incluindo eventos e notificações, fará referência ao ID do tíquete.

  Dados do jogador

  Lista de jogadores para os quais você quer criar uma correspondência. Se algum dos jogadores na solicitação não atender aos requisitos de correspondência, com base nas regras de correspondência e nos mínimos de latência, a solicitação de marcação de jogos nunca resultará em uma correspondência bem-sucedida. Você pode incluir até dez jogadores em uma solicitação de correspondência. Quando há vários jogadores em uma solicitação, FlexMatch O tenta criar uma única correspondência e atribuir todos os jogadores à mesma equipe (selecionada aleatoriamente). Se uma solicitação contiver muitos jogadores para caber em uma das equipes de correspondência, a solicitação não será correspondida. Por exemplo, se você tiver configurado o marcador de jogos para criar correspondências 2v2 (duas equipes de dois jogadores), você não poderá enviar uma solicitação de marcação de jogos contendo mais de dois jogadores.

Note

Um jogador (identificado pelo ID) só pode ser incluído em uma solicitação de marcação de jogos por vez. Ao criar uma nova solicitação para um jogador, todos os tíquetes de marcação de jogos ativos com o mesmo ID de jogador são automaticamente cancelados.

Para cada jogador listado, inclua os seguintes dados:

- ID do jogador— Cada jogador deve ter um ID de jogador exclusivo, gerado por você. ConsulteGerar IDs de jogador.
- Os atributos do jogador— Se o marcador de jogos usado chamar atributos do jogador, a solicitação deverá fornecer esses atributos para cada jogador. Os atributos necessários são definidos no conjunto de regras do marcador de jogos, que também especifica o tipo de dados do atributo. Um atributo é opcional somente quando o conjunto de regras especifica um valor padrão para ele. Se a solicitação de correspondência não fornecer os atributos necessários para todos os jogadores, a solicitação de marcação não será bem-sucedida. Saiba mais sobre conjuntos de regras do marcador e os atributos de jogador em Criar um conjunto de regras do FlexMatch (p. 16) e Exemplos de conjunto de regras FlexMatch (p. 26).
- Player latencies— Se o marcador de jogos em uso tiver uma regra de latência de jogador, a solicitação deverá relatar a latência para cada jogador. Os dados de latência são uma lista de um ou mais valores para cada jogador. Eles representam a latência enfrentada pelo jogador nas regiões da fila do marcador de jogos. Se nenhum valor de latência for incluídos na solicitação, o jogador não poderá ser correspondido, e a solicitação falhará.

Recuperar os detalhes da solicitação de correspondência:

- Assim que uma solicitação de correspondência for enviada, você poderá visualizar os detalhes da solicitação chamandoDescribeMatchmakingcom o ID do tíquete da solicitação. Essa chamada retorna as informações da solicitação, incluindo o status atual. Quando uma solicitação tiver sido concluída com êxito, o tíquete também conterá as informações necessárias para que um cliente de jogos se conecte à correspondência.

Cancelar uma solicitação de correspondência:

- Você pode cancelar uma solicitação de marcação de jogos a qualquer momento chamandoStopMatchmakingcom o ID do tíquete da solicitação.

Acompanhe os eventos de marcação

Configure notificações para rastrear eventos que GameLift emite para processos de matchmaking. Você pode configurar as notificações diretamente, criando um tópico de SNS ou usando a Amazon EventBridge. Para obter mais informações sobre a configuração de notificações, consulte Configurar FlexMatch Notificações de eventos do (p. 42). Quando você tiver configurado as notificações, adicione um ouvinte em seu serviço de cliente para detectar os eventos e responder, conforme necessário.

Também é uma boa ideia fazer backup das notificações sondando periodicamente as atualizações de status quando passar um período significativo sem notificação. Para minimizar o impacto no desempenho de marcação de jogos, certifique-se de fazer uma sondagem somente depois de aguardar pelo menos 30 segundos após o envio do ticket de marcação de jogos ou após a última notificação recebida.

Recupere um tíquete de solicitação de marcação de jogos, incluindo o status atual, chamandoDescribeMatchmakingcom o ID do tíquete da solicitação. Recomendamos a sondagem não mais de uma vez a cada 10 segundos. Essa abordagem é para uso somente durante cenários de desenvolvimento de baixo volume.

Note

Você deve configurar seu jogo com notificações de eventos antes do uso da marcação de jogos de alto volume, como teste de carga de pré-produção. Todos os jogos em versão pública devem usar notificações independentemente do volume. A abordagem de sondagem contínua é apropriada apenas para jogos em desenvolvimento com baixo uso de marcação de jogos.

Solicitar aceitação do jogador

Se você estiver usando um marcador de jogos com a aceitação do jogador ativada, adicione o código ao seu serviço de cliente para gerenciar o processo de aceitação do jogador. O processo de gerenciamento de aceitações de jogadores é idêntico para jogos que usam FlexMatch com GameLift-hospedagem gerenciada e para jogos que usam FlexMatch como uma solução autônoma.

Solicitar aceitação do jogador para uma correspondência proposta:

1. Detectar quando uma correspondência proposta precisa a aceitação do jogador. Monitore o tíquete de marcação para detectar quando o status mudar para `REQUIRES_ACCEPTANCE`. Uma alteração nesse status aciona o FlexMatch evento`MatchmakingRequiresAcceptance`.
2. Obtenha aceitações de todos os jogadores. Crie um mecanismo para apresentar os detalhes da correspondência proposta para cada jogador no tíquete da marcação de jogos. Os jogadores devem ser capazes de indicar que aceitam ou rejeitam a correspondência proposta. Você pode recuperar os detalhes da correspondência chamandoDescribeMatchmaking. Os jogadores têm um tempo limitado para responder antes que o marcador de jogos revogue a correspondência proposta e continue em frente.
3. Relate respostas do jogador para o FlexMatch. Relate respostas do jogador, chamandoAcceptMatchcom aceitação ou rejeição. Todos os jogadores em uma solicitação de marcação de jogos devem aceitar a correspondência para ela e continuar.
4. Gerencie os tíquetes com aceitações falhas. Uma solicitação falha quando qualquer jogador na correspondência proposta rejeita a correspondência ou não responde até o limite de tempo. Os ingressos para jogadores que aceitaram a partida são automaticamente devolvidos ao pool de ingressos. Os tíquetes para jogadores que não aceitaram a correspondência ganham um status de falha e não são mais processados. Para tíquetes com vários jogadores, se algum jogador no tíquete não aceitar a partida, o tíquete inteiro falhará.

Connect a uma correspondência

Adicione seu serviço de cliente para processar uma correspondência formada com êxito (status)`COMPLETED`ou evento`MatchmakingSucceeded`). Isso inclui notificar os jogadores correspondentes e enviar informações de conexão aos seus clientes de jogos.

Para jogos que usam GameLift hospedagem gerenciada, quando uma solicitação de marcação de jogos é atendida com êxito, as informações de conexão da sessão de jogo são adicionadas ao tíquete de marcação de jogos. Recupere um tíquete de marcação de jogos concluído chamandoDescribeMatchmaking. As informações de conexão incluem o endereço IP e a porta do jogo, bem como um ID de sessão do jogador para cada um. Saiba mais em GameSessionConnectionInfo. Seu cliente de jogo pode usar essas informações para se conectar diretamente à sessão do jogo da correspondência. A solicitação de conexão deve incluir um ID de sessão de jogador e um ID de jogador. Esses dados associam o jogador conectado à sessão do jogo, que inclui os dados de correspondência da atribuições de equipe (consulte oGameSession).

Para jogos que usam outras soluções de hospedagem, incluindo GameLift FleetIQ, você deve criar um mecanismo para permitir que os jogadores da partida se conectem à sessão de jogo apropriada.

Amostra de solicitações de marcação

Os trechos de código a seguir criam solicitações de marcação de jogos para vários marcadores de jogos. Conforme descrito, uma solicitação deve fornecer os atributos de jogador que são necessárias pelo marcador de jogos em uso, conforme definido no conjunto de regras do marcador. O atributo fornecido deve usar o mesmo tipo de dados, número (N) ou string (S) definido no conjunto de regras.

```
# 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)
```

Adicionar o FlexMatch a um servidor de jogos hospedado pelo Gamelift

Este tópico descreve como adicionar suporte à marcação de jogos FlexMatch a servidores de jogos personalizados que estão usando a hospedagem gerenciada do GameLift. Para saber mais sobre como adicionar o FlexMatch aos jogos, consulte os seguintes tópicos:

- Como o Amazon GameLift FlexMatch funciona (p. 3)
- Integração FlexMatch com hospedagem GameLift (p. 8)

As informações neste tópico pressupõem que você integrou com sucesso o GameLift Server SDK ao seu projeto de servidor de jogos, como descrito emAdicionar o GameLift ao servidor de jogos. Com essa tarefa concluída, você tem a maioria dos mecanismos de que precisa. As seções neste tópico abrangem o trabalho restante para lidar com jogos configurados com o FlexMatch.

Configure o servidor para marcações de jogos

Para configurar seu servidor para lidar com jogos correspondentes, conclua as tarefas a seguir.

1. Iniciar sessões de jogos criadas com a marcação de jogos. Para solicitar uma nova sessão de jogo, o GameLift envia umonStartGameSession()Solicitar o servidor de jogos com um objeto de sessão de jogo (consulteGameSession). Seu servidor de jogos usa as informações da sessão do jogo, incluindo dados personalizados do jogo, para iniciar a sessão solicitada. Para obter mais detalhes, consulte .Inicie uma sessão de jogo.

   Para jogos correspondentes, o objeto de sessão do jogo também contém um conjunto de dados do marcador de jogos. Os dados do marcador de jogos incluem informações de que o seu servidor de jogos precisa para iniciar uma nova sessão de jogo para a correspondência. Isso inclui a estrutura da equipe da correspondência, as atribuições da equipe e alguns atributos de jogador que podem ser relevantes para seu jogo. Por exemplo, seu jogo pode desbloquear determinados recursos ou níveis com base no nível de habilidade médio por jogador, ou escolher um mapa com base nas preferências dos jogadores. Saiba mais em Trabalhar com os dados do marcador de jogos (p. 52).
2. Gerenciar conexões de jogador. Ao se conectar a um jogo, um cliente de jogos consulta um ID de jogador e um ID de sessão do jogador (consulteValidar a um novo jogador). O servidor de jogos usa o ID do jogador para associar um jogador novo às informações do jogador nos dados do marcador. Os dados do marcador de jogos identificam a atribuição da equipe do jogador e podem fornecer outras informações para representar corretamente o jogador no jogo.
3. Informar quando os jogadores deixam um jogo. Verifique se seu servidor de jogos está chamando a Server APIRemovePlayerSession()para denunciar jogadores descartados (consulteDenunciar o término da sessão). Essa etapa é importante se você estiver usando o FlexMatch para preencher slots vazios em jogos existentes. Ele é essencial se o jogo iniciar solicitações de alocação por meio de um serviço de jogo no lado do cliente. Saiba mais sobre a implementação da transferência do FlexMatch noEnchimento de jogos existentes com FlexMatch (p. 53).
4. Solicitar novos jogadores para sessões de jogo existentes correspondentes (opcional). Decida como você quer preencher seus jogos correspondentes. Se seus marcadores de jogos estiverem com o modo de alocação definido como "manual", é recomendável adicionar o suporte de alocação ao seu jogo. Se seus marcadores de jogos estiverem com o modo de alocação definido como "automático", é recomendável desativar essa função nas sessões de jogos individuais. Por exemplo, talvez você queira interromper uma sessão de alocação de jogo uma vez após determinado ponto do jogo ser

atingido. Saiba mais sobre o gerenciamento de correspondência de alocação em Enchimento de jogos existentes com FlexMatch (p. 53).

Trabalhar com os dados do marcador de jogos

Seu servidor de jogos deve ser capaz de reconhecer e usar as informações do jogo em um objeto GameSession. O serviço GameLift transmite esses objetos para seu servidor de jogos sempre que uma sessão de jogo é iniciada ou atualizada. As principais informações da sessão de jogo incluem ID e nome da sessão, número máximo de jogadores, informações sobre a conexão e dados personalizados do jogo (se fornecidos).

Para sessões de jogos que são criadas usando o FlexMatch, o`GameSession`O objeto também contém um conjunto de dados do marcador de jogos. Além de um ID de correspondência exclusiva, ele identifica o marcador de jogos que criou a correspondência e descreve as equipes, as atribuições das equipes e os jogadores. Ele inclui os atributos de jogador da solicitação de marcação de jogos original (consulte o objeto Player). Ele não inclui a latência do jogador. Se você precisar dos dados de latência dos jogadores atuais, como, por exemplo, para alocação de correspondência, recomendamos obter dados atualizados.

Note

Os dados do marcador de jogos especificam o ARN completo da configuração de jogos, que identifica o nome da configuração,AWSconta e região. Ao solicitar a alocação de correspondência de um cliente serviço de jogos, somente o nome da configuração é necessário. É possível extrair o nome da configuração analisando a string subsequente a ":matchmakingconfiguration/". No exemplo mostrado, o nome da configuração da marcação de jogos é "MyMatchmakerConfig".

O JSON a seguir mostra um típico conjunto de dados do marcador de jogos. Este exemplo descreve um jogo de dois jogadores, no qual a correspondência dos jogadores é feita com base nas classificações de habilidades e nos maiores níveis alcançados. O marcador de jogos também considera o personagem e garante que os jogadores correspondentes têm pelo menos uma preferência de mapa em comum. Nesse cenário, o servidor de jogos deve ser capaz de determinar qual mapa é o preferido e usá-lo na sessão do jogo.

```
{
 "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}}
   }
  }]
 }]
}
```

Enchimento de jogos existentes com FlexMatch

A alocação de correspondência usa os mecanismos do FlexMatch para encontrar novos jogadores nas sessões de jogos existentes correspondentes. Embora você sempre possa adicionar jogadores a qualquer jogo (vejaJunte-se a um jogador a uma sessão de jogo), a alocação de correspondência garante que os novos jogadores atendem aos mesmos critérios de correspondência que os jogadores atuais. Além disso, a alocação de correspondência atribui os novos jogadores às equipes, gerencia a aceitação de jogadores e envia informações atualizadas sobre correspondência para o processo do servidor de jogos. Saiba mais sobre a alocação de correspondência em Processo de marcar jogos do FlexMatch (p. 4).

> Note
>
> O preenchimento do FlexMatch não está disponível no momento para jogos que usam os Servidores em tempo real.

Existem dois tipos de mecanismos de alocação:

- Para preencher sessões de jogos que começam com menos jogadores do que o máximo permitido, habilite a alocação automática.
- Para substituir os jogadores que desistirem de uma sessão de jogo em andamento, adicione a funcionalidade ao servidor do jogo para enviar solicitações de alocação.

Ativar a alocação automática

Com a alocação automática de correspondência, o GameLift acionará automaticamente uma solicitação de alocação sempre que uma sessão de jogo começa com um ou mais slots de jogadores não preenchidos. Este recurso permite que os jogos comecem assim que o número mínimo de jogadores combinados for encontrado e preenche os slots restantes mais tarde, à medida que jogadores adicionais forem combinados. É possível optar por interromper a alocação automática a qualquer momento.

Como exemplo, considere um jogo que pode conter de seis a dez jogadores. O FlexMatch localiza inicialmente seis jogadores, forma a partida e inicia uma nova sessão de jogo. Com a alocação automática, a nova sessão de jogo pode solicitar imediatamente mais quatro jogadores. Dependendo do estilo de jogo, podemos permitir que novos jogadores entrem a qualquer momento durante a sessão do jogo. Como alternativa, é possível interromper a alocação automática após a fase inicial de configuração e antes do início da partida.

Para adicionar uma alocação automática ao seu jogo, faça as seguintes atualizações no seu jogo.

1. Habilite a alocação automática. A alocação automática é gerenciada em uma configuração de marcação de jogos. Quando ativado, ele é usado com todas as sessões de jogo combinadas criadas com esse matchmaker. O GameLift começa a gerar solicitações de alocação para uma sessão de jogo não completa assim que a sessão inicia em um servidor de jogos.

   Para ativar a alocação automática, abra uma correspondência de configuração e defina o modo de alocação como "AUTOMÁTICO". Para obter mais detalhes, consulte Criar uma configuração marcação de jogos (p. 12).
2. Ativar a priorização da alocação. Personalize seu processo de matchmaking para priorizar o preenchimento de solicitações de preenchimento antes de criar novas correspondências. Em seu conjunto de regras de matchmaking, adicione um componente de algoritmo e defina a prioridade de preenchimento como “alta”. Para obter mais detalhes, consulte Personalize o algoritmo de correspondência (p. 17).
3. Atualize a sessão do jogo com novos dados da marcadora de jogos. O Amazon GameLift atualiza o servidor de jogos com informações de correspondência usando a função de retorno de chamada do SDK Server`onUpdateGameSession`(consulteInicialize o processo do servidor). Adicione o

código ao seu servidor de jogos para manipular os objetos de sessão de jogo atualizado como resultado da atividade de alocação. Saiba mais em Atualizar dados de correspondência no servidor do jogo (p. 58).

4. Desative a alocação automática para uma sessão de jogo. Você pode optar por interromper a alocação automática a qualquer momento durante uma sessão de jogo individual. Para interromper a alocação automática, adicione o código para o cliente de jogo ou o servidor de jogos para fazer a chamada de API do GameLiftStopMatchmaking. Essa chamada requer o ID do tíquete. Use o ID de tíquete de alocação da última solicitação. Você pode obter essas informações nos dados de marcação da sessão do jogo, que são atualizados conforme descrito na etapa anterior.

Enviar solicitações de preenchimento (de um servidor de jogos)

Você pode iniciar as solicitações de alocação de correspondência diretamente do processo do servidor de jogos que está hospedando a sessão do jogo. O processo do servidor tem as informações mais atualizadas sobre os jogadores atuais conectados ao jogo e o status dos slots de jogador vazios.

Este tópico pressupõe que você já criou os componentes necessários do FlexMatch e adicionou com êxito os processos de criação de jogos ao seu servidor de jogos e ao serviço de jogo do lado do cliente. Para obter mais detalhes sobre como configurar o FlexMatch, consulte Integração FlexMatch com hospedagem GameLift (p. 8).

Para habilitar a alocação de correspondência para seu jogo, adicione as seguintes funcionalidades:

- Enviar solicitações de alocação de marcação de jogos a um marcador de jogos e acompanhar o status das solicitações.
- Atualizar as informações de correspondência para a sessão do jogo. Consulte Atualizar dados de correspondência no servidor do jogo (p. 58).

Assim como ocorre com outras funcionalidades do servidor, um servidor de jogos usa o SDK Server do Amazon GameLift. Esse SDK está disponível em C++ e C #.

Para fazer solicitações de alocação de correspondência a partir do seu servidor de jogo, execute as seguintes tarefas.

1. Acione uma solicitação de alocação de correspondência. Geralmente, você inicia uma solicitação de alocação sempre que um jogo correspondente tem um ou mais slots de jogador vazios. Você pode vincular as solicitações de alocação a circunstâncias específicas, como preencher funções de personagens críticos ou equilibrar as equipes. Talvez você também queira limitar a atividade de alocação com base no tempo de uma sessão de jogo.
2. Crie uma solicitação de alocação. Adicione o código para criar e enviar solicitações de alocação de correspondência para um marcador de jogos do FlexMatch. As solicitações de alocação são processadas usando estas APIs de servidor:

   - StartMatchBackfill()
   - StopMatchBackfill()

   Para criar uma solicitação de alocação, chame `StartMatchBackfill` com as informações a seguir. Para cancelar uma solicitação de alocação, chame `StopMatchBackfill` com o ID do tíquete de solicitação de alocação.

   - ID do tíquete— Forneça um ID de tíquete de marcação de jogo (ou faça com que ele seja gerado automaticamente). Você pode usar o mesmo mecanismo para atribuir IDs de tíquete a solicitações

de alocação e de marcação de jogos. Os tíquetes para a marcação de jogos e alocação são processados da mesma forma.

- Marcador de jogos— Identifique o marcador de jogos a ser usado para a solicitação de alocação. Geralmente, você usa o mesmo marcador de jogo usado para criar o jogo original. Essa solicitação usa um ARN de configuração de marcação de jogos. Essas informações são armazenadas no objeto de sessão do jogo (GameSession), que foi fornecido para o processo do servidor pelo Amazon GameLift ao ativar a sessão do jogo. O ARN da configuração da marcação do jogo está incluído na propriedade `MatchmakerData`.
- ARN da sessão de jogo— Identifique a sessão do jogo que está sendo alocada. Você pode obter o ARN da sessão do jogo chamando a API do servidorGetGameSessionId(). Durante o processo de marcação de jogos, os tíquetes para as novas solicitações não têm um ID de sessão de jogo, mas os tíquetes de solicitações de alocação têm. A presença de um ID de sessão do jogo é uma forma de saber a diferença entre os tíquetes de jogos novos e os tíquetes de alocações.
- Dados do jogador— Incluir informações do jogador (Player) Para todos os jogadores atuais na sessão do jogo você está alocando. Essas informações permitem que o marcador de jogos localize as melhores correspondências possíveis entre jogadores para os jogadores presentes na sessão de jogo atual. Você deve incluir a associação da equipe para cada jogador. Não especifique uma equipe se você não estiver usando o preenchimento. Se o seu servidor de jogos estiver informando o status da conexão dos jogadores com precisão, você poderá obter esses dados da seguinte forma:

  1. O processo do servidor que hospeda a sessão de jogo deve ter as informações mais atuais sobre quais jogadores estão atualmente conectados à sessão do jogo.
  2. Para obter IDs de jogadores, atributos e atribuições de equipe, obtenha os dados dos jogadores do objeto de sessão do jogo (GameSession),`MatchmakerData`propriedade (consulteTrabalhar com os dados do marcador de jogos (p. 52)). Os dados do marcador de jogos incluem todos os jogadores que foram inseridos por correspondência na sessão do jogo. Portanto, você precisará obter os dados apenas dos jogadores conectados no momento.
  3. Para obter a latência do jogador, se o marcador de jogos fizer chamadas para os dados de latência, colete os novos valores de latência de todos os jogadores atuais e os inclua em cada objeto `Player`. Se a latência de dados for omitida e o marcador de jogos tiver uma regra de latência, a correspondência dessa solicitação não será bem-sucedida. As solicitações de alocação exigem dados de latência somente para a região onde a sessão de jogo se encontra no momento. Você pode obter a região onde a sessão do jogo está na propriedade `GameSessionId` do objeto `GameSession`; este valor é um ARN, o que inclui a região.

3. Acompanhar o status de uma solicitação de alocação. O Amazon GameLift atualiza o servidor de jogos com o status das solicitações de alocação usando a função de retorno de chamada do SDK Server`onUpdateGameSession`(consulteInicialize o processo do servidor). Adicione o código para processar as mensagens de status, assim como os objetos atualizados da sessão de jogo como resultado das solicitações de alocação bem-sucedidas, emAtualizar dados de correspondência no servidor do jogo (p. 58).

   Um marcador de jogos pode processar apenas uma solicitação de alocação de correspondência de uma sessão de jogo por vez. Se você precisar cancelar uma solicitação, chameStopMatchBackfill(). Se você precisar alterar uma solicitação, chame `StopMatchBackfill` e, em seguida, envie uma solicitação atualizada.

Enviar solicitações de preenchimento (de um serviço ao cliente)

Como alternativa ao envio de solicitações de alocação a partir de um servidor de jogos, você pode enviá-las a partir de um serviço de jogo do lado do cliente. Para usar essa opção, o serviço do lado do cliente deve ter acesso aos dados atuais sobre as atividades da sessão de jogo e as conexões dos jogadores; se o seu jogo usa um serviço de diretório de sessão, essa pode ser uma boa opção.

Este tópico pressupõe que você já criou os componentes necessários do FlexMatch e adicionou com êxito os processos de criação de jogos ao seu servidor de jogos e ao serviço de jogo do lado do cliente. Para obter mais detalhes sobre como configurar o FlexMatch, consulte Integração FlexMatch com hospedagem GameLift (p. 8).

Para habilitar a alocação de correspondência para seu jogo, adicione as seguintes funcionalidades:

- Enviar solicitações de alocação de marcação de jogos a um marcador de jogos e acompanhar o status das solicitações.
- Atualizar as informações de correspondência para a sessão do jogo. Consulte Atualizar dados de correspondência no servidor do jogo (p. 58)

Assim como ocorre com outras funcionalidades do cliente, o serviço de jogo do lado do cliente usa oAWSSDK com a API do Amazon GameLift. Este SDK está disponível em C++, C# e em diversas outras linguagens. Para obter uma descrição geral das APIs de cliente, consulte a Referência de API do serviço Amazon GameLift, que descreve a API de serviço de nível baixo para ações relacionadas ao Amazon GameLift e inclui links para guias de referência específicos da linguagem.

Para configurar um serviço de jogo do lado do cliente para alocar jogos correspondentes, execute as tarefas a seguir.

1. Acione uma solicitação de alocação. Geralmente, um jogo inicia uma solicitação de alocação sempre que um jogo correspondente tem um ou mais slots de jogador vazios. Você pode vincular as solicitações de alocação a circunstâncias específicas, como preencher funções de personagens críticos ou equilibrar as equipes. Talvez você também queira limitar a alocação com base no tempo de uma sessão de jogo. Independentemente da forma como acionará a solicitação, você precisará ao menos das informações a seguir. Você pode obter essas informações do objeto da sessão do jogo (GameSession) chamando DescribeGameSessions com um ID de sessão de jogo.

   - Número de slots de jogador vazios no momento. Esse valor pode ser calculado a partir do limite máximo de jogadores de uma sessão de jogo e da contagem atual de jogadores. A contagem atual de jogadores é atualizada sempre que o servidor de jogos entra em contato com o serviço Amazon GameLift para validar uma nova conexão de jogador ou para informar que um jogador saiu.

   - Política de criação. Essa configuração indica se a sessão do jogo está aceitando novos jogadores no momento.

   O objeto da sessão do jogo contém outras informações potencialmente úteis, incluindo o horário de início da sessão do jogo, as propriedades de jogos personalizados e os dados do marcador de jogos.
2. Crie uma solicitação de alocação. Adicione o código para criar e enviar solicitações de alocação de correspondência para um marcador de jogos do FlexMatch. As solicitações de alocação são processadas usando estas APIs de cliente:

   - StartMatchBackfill
   - StopMatchmaking

   Para criar uma solicitação de alocação, chame `StartMatchBackfill` com as informações a seguir. Uma solicitação de alocação é semelhante a uma solicitação de marcação de jogos (consulte Solicitar marcação de jogos para jogadores (p. 47)), mas também identifica a sessão do jogo existente. Para cancelar uma solicitação de alocação, chame `StopMatchmaking` com o ID do tíquete de solicitação de alocação.

   - ID do tíquete— Forneça um ID de tíquete de marcação de jogo (ou faça com que ele seja gerado automaticamente). Você pode usar o mesmo mecanismo para atribuir IDs de tíquete a solicitações

de alocação e de marcação de jogos. Os tíquetes para a marcação de jogos e alocação são processados da mesma forma.

- Marcador de jogos— Identifique o nome de uma configuração de marcação de jogos para usar. Geralmente, você usa o mesmo marcador de jogo para alocar que foi usado para criar a correspondência original. Essas informações estão em um objeto de sessão do jogo (GameSession), na propriedade `MatchmakerData`, sob o ARN da configuração de marcação de jogos. O valor de nome é a string que aparece logo após ""matchmakingconfiguration/". (Por exemplo, no valor do ARN "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4", o nome da configuração de marcação de jogos é "MM-4v4".)
- ARN da sessão de jogo— Especifique a sessão do jogo que está sendo alocada. Use a propriedade `GameSessionId` do objeto de sessão do jogo; esse ID usa o valor do ARN de que você precisa. Os tíquetes de marcação de jogos (MatchmakingTicket) para as solicitações de alocação têm o ID de sessão do jogo enquanto estão sendo processados; os tíquetes de novas solicitações de marcação de jogos não recebem um ID de sessão do jogo até que a correspondência seja inserida; a presença do ID de sessão do jogo é uma forma de saber a diferença entre os tíquetes de novas correspondências e os tíquetes de alocação.
- Dados do jogador— Incluir informações do jogador (Player) Para todos os jogadores atuais na sessão do jogo você está alocando. Essas informações permitem que o marcador de jogos localize as melhores correspondências de jogadores possíveis para os jogadores presentes na sessão de jogo atual. Você deve incluir a associação da equipe para cada jogador. Não especifique uma equipe se você não estiver usando o preenchimento. Se o seu servidor de jogos estiver informando o status da conexão dos jogadores com precisão, você poderá obter esses dados da seguinte forma:

  1. Chame DescribePlayerSessions() com o ID de sessão do jogo para descobrir todos os jogadores que estão conectados à sessão do jogo no momento. Cada sessão de jogador inclui um ID de jogador. Você pode adicionar um filtro de status para recuperar somente as sessões de jogador ativas.
  2. Obtenha os dados de jogador no objeto de sessão do jogo (GameSession), na propriedade `MatchmakerData` (consulte Trabalhar com os dados do marcador de jogos (p. 52). Use os IDs de jogador adquiridos na etapa anterior para obter apenas os dados dos jogadores conectados no momento. Como os dados do marcador de jogos não são atualizados quando os jogadores abandonam o jogo, você precisa extrair os dados apenas para os jogadores atuais.
  3. Para obter a latência do jogador, se o marcador de jogos fizer chamadas para os dados de latência, colete os novos valores de latência de todos os jogadores atuais e os inclua no objeto `Player`. Se a latência de dados for omitida e o marcador de jogos tiver uma regra de latência, a correspondência dessa solicitação não será bem-sucedida. As solicitações de alocação exigem dados de latência somente para a região onde a sessão de jogo se encontra no momento. Você pode obter a região onde a sessão do jogo está na propriedade `GameSessionId` do objeto `GameSession`; este valor é um ARN, o que inclui a região.

3. Acompanhe o status da solicitação de alocação. Adicione o código para seguir as atualizações de status do tíquete de marcação de jogos. Você pode usar a configuração do mecanismo para acompanhar os tíquetes das novas solicitações de marcação de jogos (consulte Acompanhe os eventos de marcação (p. 48)) usando a notificação de evento (de preferência) ou a sondagem. Embora você não precise acionar a atividade de aceitação de jogadores com solicitações de alocação e as informações dos jogadores sejam atualizadas no servidor de jogos, você ainda precisa monitorar o status do tíquete para tratar as falhas de solicitação e as solicitações que foram reenviadas.

   Um marcador de jogos pode processar apenas uma solicitação de alocação de correspondência de uma sessão de jogo por vez. Se você precisar cancelar uma solicitação, chame StopMatchmaking. Se você precisar alterar uma solicitação, chame `StopMatchmaking` e, em seguida, envie uma solicitação atualizada.

   Assim que uma solicitação de alocação é bem-sucedida, o servidor de jogos recebe um objeto `GameSession` atualizado e processa as tarefas necessárias para incluir os novos jogadores na sessão do jogo. Veja mais em Atualizar dados de correspondência no servidor do jogo (p. 58).

Atualizar dados de correspondência no servidor do jogo

Independentemente de como você inicia as solicitações de alocação de correspondência em seu jogo, o servidor de jogos deve ser capaz de processar as atualizações da sessão do jogo que o Amazon GameLift fornece como resultado das solicitações de alocação de correspondência.

Quando o Amazon GameLift conclui uma solicitação de alocação de correspondência, com ou não, ele chama o servidor de jogos usando a função de retorno de chamadaonUpdateGameSession. Essa chamada tem três parâmetros de entrada: um ID de tíquete de alocação de correspondência, uma mensagem de status, um objeto GameSession contendo os dados de marcação de jogos mais atualizados, incluindo as informações dos jogadores. Você precisa adicionar o código a seguir ao seu servidor de jogos como parte da integração do servidor de jogos:

1. Implemente a função onUpdateGameSession. Esta função deve ser capaz de processar as seguintes mensagens de status (updateReason):
   - MATCHMAKING_DATA_UPDATED — A correspondência dos novos jogadores com a sessão do jogo foi bem-sucedida. O objeto GameSession contém os dados atualizados do marcador de jogos, incluindo dados dos jogadores existentes e dos jogadores recém-incluídos por correspondência.
   - BACKFILL_FAILED — A tentativa de alocação de correspondência falhou devido a um erro interno. O objeto GameSession não é alterado.
   - BACKFILL_TIMED_OUT — O marcador de jogos não encontrou uma correspondência para alocar dentro do limite de tempo. O objeto GameSession não é alterado.
   - BACKFILL_CANCELLED — A solicitação de alocação de correspondência foi cancelada por uma chamada StopMatchmaking (cliente) ou StopMatchBackfill (servidor). O objeto GameSession não é alterado.
2. Para que as correspondências de alocação sejam bem-sucedidas, use os dados atualizados do marcador de jogos para processar os novos jogadores quando eles se conectarem à sessão do jogo. No mínimo, você precisará usar as atribuições da equipe para os novos jogadores, bem como outros atributos de jogadores que são necessários para que o jogador possa começar a jogar.
3. Na chamada do servidor de jogo para a ação do SDK ServerProcessReady(), adicione oonUpdateGameSessionnome do método callback como um parâmetro de processo.

Referência do GameLift FlexMatch

Esta seção contém documentação de referência sobre a correspondência com o GameLift FlexMatch.

Tópicos

GameLift FlexMatch Referência da API (AWSSDK)

Este tópico fornece uma listagem baseada em tarefas de operações de API para o GameLift FlexMatch. O GameLift FlexMatcha API de serviço do é empacotada noAWSSDK no`aws.gameliftnamespace`.

GameLift FlexMatch fornece serviços de matchmaking para uso com jogos hospedados com GameLiftsoluções de hospedagem (incluindo hospedagem gerenciada para servidores de jogos personalizados ou servidores em tempo real) e hospedagem no Amazon EC2 com GameLift FleetIQ), bem como com outros sistemas de hospedagem, como peer-to-peer, primitivas de computação local ou em nuvem. Consulte oGameLift Guia do desenvolvedorpara obter mais informações sobre outros GameLift opções de hospedagem.

Configurar regras e processos de marcação de jogos

Chame essas operações para criar um FlexMatch marcação de jogos, configure o processo de marcação de jogos do e defina um conjunto de regras personalizadas para criar partidas e equipes.

Configuração da marcação de jogos

- CreateMatchmakingConfiguration— crie uma configuração de marcação de jogos com instruções para avaliar grupos de jogadores e criar equipes de jogadores. Ao usar GameLift para hospedagem, especifique também como criar uma nova sessão de jogo para a correspondência.
- DescribeMatchmakingConfigurations— Recuperar configurações de marcação de jogos definidas em GameLift região.
- UpdateMatchmakingConfiguration— Alterar as configurações da configuração de marcação de jogos. fila.
- DeleteMatchmakingConfiguration— remova uma configuração de marcação de jogos da região.

Conjunto de regras da marcação de jogos

- CreateMatchmakingRuleSet— crie um conjunto de regras para usar na pesquisa de correspondências de jogadores.
- DescribeMatchmakingRuleSets— Recuperar conjuntos de regras de marcação de jogos definidos em um GameLift região.
- ValidateMatchmakingRuleSet— Verifique a sintaxe de um conjunto de regras de marcação de jogos.
- DeleteMatchmakingRuleSet— remova um conjunto de regras de marcação de jogos da região.

Solicite uma partida para um jogador ou jogadores

Chame essas operações do serviço de cliente de jogo para gerenciar solicitações de marcação de jogos de jogadores.

- StartMatchmaking— Solicitar a marcação de jogos para jogadores ou grupos que desejem jogar na mesma correspondência.
- DescribeMatchmaking— Obtenha detalhes sobre uma solicitação de marcação de jogos, incluindo o status.
- AcceptMatch— Para uma correspondência que exige a aceitação do jogador, notifique GameLift quando um jogador aceitar uma correspondência proposta.
- StopMatchmaking— cancele uma solicitação de marcação de jogos.
- StartMatchBackfill- Solicitar correspondências adicionais de jogadores para preencher vagas vazias em uma sessão de jogo existente.

Linguagens de programação disponíveis

Linguagem do FlexMatch

Os tópicos de referência nesta seção descrevem a sintaxe e a semântica que são usadas para criar regras de matchmaking para uso com o GameLift FlexMatch. Para obter ajuda detalhada com a escrita de regras de pareamento e conjuntos de regras, consulteCriar um conjunto de regras do FlexMatch (p. 16).

Tópicos

Esquema conjunto de regras FlexMatch

Os conjuntos de regras FlexMatch usam esquema padrão para regras de correspondências pequenas e grandes. Para obter descrições detalhadas de cada seção do, consulteDefinições de propriedade do conjunto de regras Flex (p. 63).

Esquema de conjunto de regras para pequenas correspondências

O esquema a seguir documenta todas as propriedades possíveis e valores permitidos para um conjunto de regras usado para criar partidas de até 40 jogadores.

```
{
    "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
        }]
    }]
}
```

Esquema de conjunto de regras para correspondências grandes

O esquema a seguir documenta todas as propriedades possíveis e valores permitidos para um conjunto de regras usado para correspondências com mais de 40 jogadores. Se o total de`maxPlayers`Os valores de todas as equipes do conjunto de regras excede 40 e os processos do FlexMatch correspondem às solicitações que usam esse conjunto de regras de acordo com as diretrizes para as correspondências grandes.

```
{
    "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
        }]
    }]
}
```

Definições de propriedade do conjunto de regras Flex

Esta seção define cada propriedade no esquema do conjunto de regras. Para obter ajuda adicional com a criação de um conjunto de regras, consulteCriar um conjunto de regras do FlexMatch (p. 16).

`name`

Um rótulo descritivo para o conjunto de regras. Esse valor não está associado ao nome atribuído ao GameLiftRecurso MatchMakingRuleSet. Esse valor está incluído nos dados de matchmaking que descrevem uma correspondência concluída, mas não é usado por nenhum processo do GameLift.

Valores permitidos: String

Obrigatório? Não

`ruleLanguageVersion`

A versão da linguagem de expressão de propriedade FlexMatch que está sendo usada.

Valores permitidos: “1.0"

Obrigatório? Sim

`playerAttributes`

Uma coleção de dados do jogador que está incluída nas solicitações de matchmaking e é usada no processo de matchmaking. Você também pode declarar atributos aqui para que os dados do jogador sejam incluídos nos dados de matchmaking que são passados para servidores de jogos, mesmo que os dados não sejam usados no processo de matchmaking.

Obrigatório? Não

`name`

Um nome exclusivo para o atributo do jogador a ser usado pelo matchmaker. Esse nome deve corresponder ao nome do atributo do jogador que é referenciado em solicitações de matchmaking.

Valores permitidos: String

Obrigatório? Sim

`type`

O tipo de dados do valor do atributo player.

Valores permitidos: “string”, “number”, “string_list”, “string_number_map”

Obrigatório? Sim

`default`

Um valor padrão a ser usado quando uma solicitação de criação de jogos não fornece um para um jogador.

Valores permitidos: Qualquer valor permitido para o atributo player.

Obrigatório? Não

`algorithm`

Configurações opcionais para personalizar o processo de matchmaking.

Obrigatório? Não

`strategy`

O método a ser usado ao criar correspondências. Se essa propriedade não for definida, o comportamento padrão será “ExhaustiveSearch”.

Valores permitidos:

- “ExhaustiveSearch” — Método de correspondência padrão. O FlexMatch forma uma correspondência em torno do ticket mais antigo em um lote, avaliando outros tickets no pool com base em um conjunto de regras de correspondência personalizadas. Essa estratégia é usada para partidas de 40 jogadores ou menos. Ao usar essa estratégia,`batchingPreference`deve ser definido como “aleatório” ou “classificado”.
- “balanceado” — Método otimizado para formar grandes correspondências rapidamente. Essa estratégia é usada apenas para partidas de 41 a 200 jogadores. Ele forma partidas pré-classificando o pool de ingressos, construindo possíveis partidas e atribuindo jogadores a equipes e, em seguida, equilibrando cada equipe em uma partida usando um atributo de jogador especificado. Por exemplo, essa estratégia pode ser usada para igualar os níveis médios de habilidade de todas as equipes em uma partida. Ao usar essa estratégia,`balancedAttribute`deve ser definido e`batchingPreference`deve ser definido como “maiorPopulação” ou “FastestRegion”. A maioria dos tipos de regras personalizadas não é reconhecida com essa estratégia.

Obrigatório? Sim

`batchingPreference`

O método de pré-classificação a ser usado antes de agrupar tíquetes para construção de partidas. A pré-classificação do pool de ingressos faz com que os ingressos sejam agrupados com base em uma característica específica, que tende a aumentar a uniformidade entre os jogadores nas partidas finais.

Valores permitidos:

- “random” — Válido somente com`strategy`= “ExaustiveSearch”. Nenhuma pré-classificação é feita; os tickets no pool são agrupados aleatoriamente. Esse é o comportamento padrão para uma estratégia de pesquisa exaustiva.
- “sorted” — Válido somente com`strategy`= “ExaustiveSearch”. O pool de ingressos é pré-classificado com base nos atributos do jogador listados em`sortbyAttributes`.
- “LargestPopulation” — Válido somente com`strategy`= “equilibrado”. O pool de ingressos é pré-classificado por regiões onde os jogadores estão relatando níveis de latência aceitáveis. Esse é o comportamento padrão para uma estratégia equilibrada.
- “FastestRegion” — Válido somente com`strategy`= “equilibrado”. O pool de ingressos é pré-classificado por regiões onde os jogadores estão relatando seus níveis de latência mais baixos. As partidas resultantes demoram mais para serem concluídas, mas a latência para todos os jogadores tende a ser baixa.

Obrigatório? Sim

`balancedAttribute`

O nome de um atributo de jogador a ser usado ao construir partidas grandes com a estratégia equilibrada.

Valores permitidos: Qualquer atributo declarado em`playerAttributes`com`type`= “número”.

Obrigatório? Sim, se`strategy`= “equilibrado”.

`sortByAttributes`

Uma lista de atributos de jogadores a serem usados ao pré-classificar o pool de tickets antes do lote. Essa propriedade só é usada quando pré-classificar com a estratégia de pesquisa exaustiva.

A ordem da lista de atributos determina a ordem de classificação. O FlexMatch usa convenção de classificação padrão para valores alfa e numéricos.

Valores permitidos: Qualquer atributo declarado em`playerAttributes`.

Obrigatório? Sim, se`batchingPreference`= “classificado”.

`backfillPriority`

O método de priorização para combinar tíquetes de aterro. Essa propriedade determina quando o FlexMatch processa os tickets de preenchimento em um lote. Ele só é usado quando pré-classificar com a estratégia de busca exaustiva. Se essa propriedade não for definida, o comportamento padrão será “normal”.

Valores permitidos:

- “normal” — O tipo de solicitação de um ticket (preenchimento ou nova correspondência) não é considerado ao formar correspondências.
- “alto” — Um lote de tíquetes é classificado por tipo de solicitação (e depois por idade), e o FlexMatch tenta combinar os tíquetes de aterro primeiro.
- “baixo” — Um lote de tíquetes é classificado por tipo de solicitação (e depois por idade), e o FlexMatch tenta fazer a correspondência de tíquetes sem preenchimento antes.

Obrigatório? Não

`expansionAgeSelection`

O método para calcular o tempo de espera para uma expansão de regra de correspondência. As expansões são usadas para relaxar os requisitos de correspondência se uma partida não tiver sido concluída após um certo período de tempo. O tempo de espera é calculado com base na idade dos ingressos que já estão na partida parcialmente preenchida. Se essa propriedade não for definida, o comportamento padrão será “mais novo”.

Valores permitidos:

- “mais novo” — O tempo de espera de expansão é calculado com base no ticket com o carimbo de data/hora de criação mais recente na partida parcialmente concluída. As expansões tendem a ser acionadas mais lentamente, porque um ticket mais novo pode reiniciar o relógio de tempo de espera.
- “mais antigo” — O tempo de espera de expansão é calculado com base no ticket com o carimbo de data/hora de criação mais antigo na partida. As expansões tendem a ser acionadas mais rapidamente.

Obrigatório? Não

`teams`

A configuração das equipes em uma correspondência. Forneça um nome e uma faixa de tamanho da equipe para cada equipe. Um conjunto de regras deve definir pelo menos uma equipe.

`name`

Um nome exclusivo para a equipe. Os nomes de equipe podem ser referidos em regras e expansões. Em uma partida bem-sucedida, os jogadores são atribuídos pelo nome da equipe nos dados de matchmaking.

Valores permitidos: String

Obrigatório? Sim

`maxPlayers`

O número máximo de jogadores que podem ser atribuídos ao grupo.

Valores permitidos: Número

Obrigatório? Sim

`minPlayers`

O número mínimo de jogadores que devem ser atribuídos ao grupo antes do jogo é viável.

Valores permitidos: Número

Obrigatório? Sim

`quantity`

O número de equipes desse tipo a serem criadas em uma partida. Equipes com quantidades maiores que 1 são designadas com um número anexado (“Red_1", “Red_2", etc.). Se essa propriedade não for definida, o valor padrão será “1".

Valores permitidos: Número

Obrigatório? Não

`rules`

Uma coleção de instruções de regras que definem como avaliar os jogadores para uma correspondência.

Obrigatório? Não

`name`

Um nome exclusivo para a regra. Todas as regras em um conjunto de regras devem ter nomes exclusivos. Os nomes de regra são referenciados em logs de eventos e métricas que controlam as atividades relacionadas à regra.

Valores permitidos: String

Obrigatório? Sim

`description`

Uma descrição de texto para a regra. Essas informações podem ser usadas para identificar o propósito de uma regra. Ele não é usado no processo de criação de jogos.

Valores permitidos: String

Obrigatório? Não

`type`

O tipo de instrução de regra. Cada tipo de regra tem propriedades adicionais que devem ser definidas. Para obter mais detalhes sobre a estrutura e o uso de cada tipo de regra, consulteFlexMatch Tipos de regras (p. 67).

Valores permitidos:

- “AbsoluteSort” — Classifica usando um método de classificação explícito que solicita tickets em um lote com base em se um atributo de jogador especificado se compara ao ticket mais antigo do lote.
- “coleção” — Avalia os valores em uma coleção, como um atributo de jogador que é uma coleção ou um conjunto de valores para vários jogadores.
- “comparação” — Compara dois valores.
- “compound” — Define uma regra de matchmaking composta usando uma combinação lógica de outras regras no conjunto de regras. Suportado apenas para partidas de 40 ou menos jogadores.
- “distance” — Mede a distância entre os valores numéricos.

- “BatchDistance” — Mede a diferença entre um valor de atributo e o usa para agrupar solicitações de correspondência.
- “DistanceSort” — Classifica usando um método de classificação explícito que solicita tickets em um lote com base em como um atributo de jogador especificado com um valor numérico se compara ao ticket mais antigo do lote.
- “latência” — Avalia os dados de latência regional que são relatados para uma solicitação de pareamento.

Obrigatório? Sim

`expansions`

Regras para requisitos de partida relaxante ao longo do tempo em que uma partida não pode ser concluída. Configure expansões como uma série de etapas que se aplicam gradualmente para facilitar a localização das correspondências. Por padrão, o FlexMatch calcula o tempo de espera com base na idade do ticket mais novo adicionado a uma partida. Você pode alterar a forma como os tempos de espera de expansão são calculados usando a propriedade algoritmo`expansionAgeSelection`.

Os tempos de espera de expansão são valores absolutos, portanto, cada etapa deve ter um tempo de espera maior que o passo anterior. Por exemplo, para agendar uma série gradual de expansão, você pode usar tempos de espera de 30 segundos, 40 segundos e 50 segundos. Os tempos de espera não podem exceder o tempo máximo permitido para uma solicitação de correspondência, definido na configuração de criação de jogos.

Obrigatório? Não

`target`

O elemento do conjunto de regras a ser relaxado. Você pode relaxar as propriedades de tamanho da equipe ou qualquer propriedade de instrução de regra. A sintaxe é "<component name>[<rule/team name>]. <property name>”. Por exemplo, para alterar os tamanhos mínimos da equipe: `teams[Red, Yellow].minPlayers`. Para alterar o requisito mínimo de habilidade em uma instrução de regra de comparação chamada “minSkill”:`rules[minSkill].referenceValue`.

Obrigatório? Sim

`steps`

`waitTimeSeconds`

O período de tempo, em segundos, para esperar antes de aplicar o novo valor para o elemento do conjunto de regras de destino.

Obrigatório? Sim

`value`

O novo valor para o elemento de conjunto de regras de destino.

FlexMatch Tipos de regras

Regra de distância da

```
batchDistance
```

As regras de distância do Batch medem a diferença entre dois valores de atributo. Você pode usar o tipo de regra de distância do lote com correspondências grandes e pequenas. Há dois tipos de regras de distância em lote:

- Comparar valores de atributos numéricos. Por exemplo, uma regra de distância em lote desse tipo deve exigir que todos os jogadores em uma correspondência precisam estar dentro de dois

níveis de habilidades em relação ao outro. Para esse tipo, defina uma distância máxima entre o`batchAttribute`de todos os ingressos.

- Comparar valores de atributo de string. Por exemplo, uma regra de distância em lote desse tipo deve exigir que todos os jogadores em uma correspondência precisam solicitar o mesmo modo de jogos. Para esse tipo, defina um`batchAttribute`Valorize isso FlexMatch usa para formar lotes.

Propriedades das regras de distância do Batch

- **`batchAttribute`**— O valor do atributo player usado para formar lotes.
- **`maxDistance`**— O valor máximo da distância para uma correspondência em bem-sucedida. Usado para comparar atributos numéricos.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`), máximo (`max`) e média (`avg`) valores para os jogadores de um ticket. O padrão é `avg`.

Example

Exemplos

```
{
  "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"
}
```

Regra de comparação

```
comparison
```

As regras de comparação comparam um valor de atributo do jogador com outro valor. Há dois tipos de regras de comparação:

- Compare com o valor de referência. Por exemplo, uma regra de comparação desse tipo deve exigir que os jogadores correspondidos precisam ter um determinado nível de habilidades ou superior. Para esse tipo, especifique um atributo de jogador, um valor de referência e uma operação de comparação.
- Comparar os jogadores. Por exemplo, uma regra de comparação desse tipo deve exigir que todos os jogadores na correspondência precisam usar personagens diferentes. Para este tipo, especifique um atributo de jogador e o igual (=) ou não igual (!=) operação de comparação. Não especifique um valor de referência.

> **Note**
>
> As regras de distância em Batch são mais eficientes para comparar os atributos do jogador. Para reduzir a latência de combinação, use uma regra de distância do lote quando possível.

Propriedades das regras de comparação

- **`measurements`**— O valor do atributo do jogador para comparar.
- **`referenceValue`**— O valor para comparar a medida para uma correspondência em potencial.
- **`operation`**— O valor que determina como comparar a medição com o valor de referência. Entre as operações válidas estão:<,<=,=,!=,>,>=.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`), máximo (`max`) e média (`avg`) valores para os jogadores de um ticket. O padrão é `avg`.

Regra de distância

```
distance
```

As regras de distância medem a diferença entre dois valores numéricos, como a distância entre níveis de habilidades do jogador. Por exemplo, uma regra de distância deve exigir que todos os jogadores joguem o jogo por pelo menos 30 horas.

> Note
>
> As regras de distância em Batch são mais eficientes para comparar os atributos do jogador. Para reduzir a latência de combinação, use uma regra de distância do lote quando possível.

Propriedades das regras de distância

- **`measurements`**— O valor do atributo do jogador para o qual medir a distância. Deve ser um atributo com um valor numérico.
- **`referenceValue`**— O valor numérico para medir a distância em relação a uma correspondência em potencial.
- **`minDistance`/`maxDistance`**— O valor mínimo ou máximo da distância para uma correspondência em bem-sucedida.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`), máximo (`max`) e média (`avg`) valores para os jogadores de um ticket. O padrão é `avg`.

Regra de coleta

```
collection
```

As regras de coleta comparam um grupo de valores de atributos de jogadores com os de outros jogadores no lote ou com um valor de referência. Uma coleção pode conter valores de atributos para vários jogadores, um atributo de jogador como uma lista de strings ou ambos. Por exemplo, uma regra de coleção pode olhar para os personagens que os jogadores de uma equipe escolhem. A regra pode exigir que a equipe tenha pelo menos um de um determinado personagem.

Propriedades das regras de coleta

- **`measurements`**— A coleção de valores de atributos do jogador para comparar. Os valores de atributo precisam ser listas de string.
- **`referenceValue`**— O valor (ou coleção de valores) a ser usado para comparar medidas para uma correspondência prospectiva.
- **`operation`**— O valor que determina como comparar uma coleção de medidas. Entre as operações válidas estão as seguintes:

  - `intersection`— Esta operação mede o número de valores que são os mesmos nas coleções de todos os jogadores. Para obter um exemplo de uma regra que usa a operação de interseção, consulteExemplo 4: Use a classificação explícita para encontrar as melhores correspondências (p. 32).
  - `contains`— Esta operação mede o número de coleções de atributos do jogador que contêm o valor de referência especificado. Para obter um exemplo de uma regra que usa a operação contains, consulteExemplo 3: Definir requisitos de nível de equipe e limites de latência (p. 30).
  - `reference_intersection_count`— Esta operação mede o número de itens em uma coleção de atributos do jogador que correspondem aos itens na coleção de valores de referência. Você pode usar essa operação para comparar vários atributos de jogadores diferentes. Para ver um exemplo de uma regra que compara várias coleções de atributos de jogadores, consulteExemplo 5: Encontre interseções em vários atributos de jogador (p. 34).
- **`minCount`**/**`maxCount`**— O valor mínimo ou máximo da contagem para uma correspondência em bem-sucedida.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Para esse valor, você pode usar`union`para combinar os atributos do jogador de todos os jogadores do grupo. Ou você pode usar`intersection`para usar atributos de jogador que o grupo tem em comum. O padrão é `union`.

Regra composta

```
compound
```

As regras compostas usam declarações lógicas para formar partidas de 40 ou menos jogadores. Você pode usar várias regras compostas em um único conjunto de regras. Ao usar várias regras compostas, todas as regras compostas devem ser verdadeiras para formar uma correspondência.

Você não pode expandir uma regra composta usandoregras de Expansão (p. 19), mas você pode expandir as regras subjacentes ou de suporte.

Propriedades das regras compostas

- **`statement`**— A lógica usada para combinar regras individuais para formar a regra composta. As regras especificadas nessa propriedade devem ter sido definidas anteriormente no conjunto de regras. Você não pode usar`batchDistance`regras em uma regra composta.

  Essa propriedade é compatível com os seguintes operadores lógicos:
  - `and`— A expressão é verdadeira se os dois argumentos fornecidos forem verdadeiros.
  - `or`— A expressão é verdadeira se um dos dois argumentos fornecidos for verdadeiro.
  - `not`— Inverte o resultado do argumento na expressão.
  - `xor`— A expressão é verdadeira se apenas um dos argumentos for verdadeiro.

Example Exemplo

O exemplo a seguir combina jogadores de vários níveis de habilidade com base no modo de jogo que eles selecionam.

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

Regra de latência

```
latency
```

As regras de latência medem a latência do jogador por local. Uma regra de latência ignora qualquer local com uma latência mais alta do que o máximo. Um jogador deve ter um valor de latência abaixo do máximo em pelo menos um local para que a regra de latência os aceite. Você pode usar esse tipo de regra com correspondências grandes especificando a opção`maxLatency`propriedade.

Propriedades das regras de latência

- **`maxLatency`**— O valor máximo da latência para um local. Se um ticket não tiver locais com latência abaixo do máximo, o ticket não corresponderá à regra de latência.
- **`maxDistance`**— O valor máximo entre a latência de cada ticket e o valor de referência da distância.
- **`distanceReference`**— O valor de latência com o qual comparar a latência do ticket. Tíquetes dentro da distância máxima do valor de referência da distância resultam em uma correspondência bem-sucedida. Entre as opções válidas estão o mínimo (`min`) e média (`avg`) valores de latência do jogador.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`), máximo (`max`) e média (`avg`) valores para os jogadores de um ticket. O padrão é `avg`.

> **Note**
>
> Uma fila pode colocar uma sessão de jogo em uma região que não corresponde a uma regra de latência. Para obter mais informações sobre políticas de latência para filas, consulteCrie uma política de latência de jogadores.

Regra de classificação absoluta

```
absoluteSort
```

As regras de classificação absoluta classificam um lote de casamento tickets com base em um atributo de jogador especificado em comparação com o primeiro tíquete adicionado ao lote.

Propriedades das regras de classificação absoluta

- **`sortDirection`**- A ordem para classificar o casamento ingressos em. As opções válidas incluem`ascending`e`descending`.
- **`sortAttribute`**— O atributo do jogador para classificar os tíquetes por.
- **`mapKey`**— As opções para classificar o atributo de jogador se este for um mapa. Entre as opções válidas estão:
  - `minValue`— A chave com o valor mais baixo é a primeira.
  - `maxValue`— A chave com o valor mais alto é a primeira.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`) atributo player, o máximo (`max`) atributo de jogador e a média (`avg`) de todos os atributos dos jogadores do grupo. O padrão é `avg`.

Example

Exemplo

A regra de exemplo a seguir classifica os jogadores por nível de habilidade e calcula a média do nível de habilidade dos grupos.

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

Regra de classificação de

```
distanceSort
```

As regras de classificação de distância classificam um lote de casamento tíquetes com base na distância de um atributo de jogador especificado do primeiro tíquete adicionado ao lote.

Propriedades das regras de classificação de distância

- **`sortDirection`**- A direção para classificar casamento ingressos. As opções válidas incluem`ascending`e`descending`.
- **`sortAttribute`**— O atributo do jogador para classificar os tíquetes por.
- **`mapKey`**— As opções para classificar o atributo de jogador se este for um mapa. Entre as opções válidas estão:
  - `minValue`— Para obter o primeiro ticket adicionado ao lote, encontre a chave com o valor mais baixo.
  - `maxValue`— Para obter o primeiro ticket adicionado ao lote, encontre a chave com o valor mais alto.
- **`partyAggregation`**— O valor que determina como FlexMatch lida com ingressos com vários jogadores (grupos). Entre as opções válidas estão o mínimo (`min`), máximo (`max`) e média (`avg`) valores para os jogadores de um ticket. O padrão é `avg`.

Expressões de propriedade Flex

As expressões de propriedade podem ser usadas para definir determinadas propriedades relacionadas à criação de jogos. Eles permitem que você use cálculos e lógica ao definir um valor de propriedade. As expressões de propriedade geralmente resultam em uma dessas duas formas:

- Dados de jogador individual.
- Coleções calculadas de dados individuais do jogador.

Expressões de propriedade de matchmaking comuns

Uma expressão de propriedade identifica um valor específico para um jogador, equipe ou correspondência. As expressões parciais a seguir ilustram como identificar equipes e jogadores:

Objetivo	Entrada	Significado	Resultado
Para identificar uma equipe em uma correspondência:	`teams[red]`	A equipe Red	Team
Para identificar um conjunto de equipes em uma correspondência:	`teams[red,blue]`	A equipe Red e a equipe Blue	List<Team>

Objetivo	Entrada	Significado	Resultado
Para identificar todas as equipes em uma correspondência:	`teams[*]`	Todas as equipes	List<Team>
Para identificar jogadores em um jogo específico:	`team[red].players`	Jogadores da equipe Red	List<Player>
Para identificar jogadores em um conjunto de equipes específicas em uma correspondência:	`team[red,blue].players`	Jogadores na correspondência, agrupados por equipe	List<List<Player>>
Para identificar jogadores em uma correspondência:	`team[*].players`	Jogadores na correspondência, agrupados por equipe	List<List<Player>>

Exemplos de expressões de propriedade

A tabela a seguir ilustra algumas expressões de propriedade criadas nos exemplos anteriores:

Expressão	Significado	Tipo resultante
`teams[red].players[playerId]`	Os IDs de jogador de todos da equipe red	List<string>
`teams[red].players.attributes['skill']`	Os atributos "skill" de todos os jogadores do time red	List<number>
`teams[red,blue].players.attributes['skill']`	Os atributos "skill" de todos os jogadores da equipe Red e do time Blue, agrupados por equipe	List<List<number>>
`teams[*].players.attributes['skill']`	Os atributos "skill" de todos os jogadores na correspondência, agrupados por equipe	List<List<number>>

Agregações de propriedade

As expressões de propriedade podem ser usadas para agregar dados da equipe, usando as funções ou combinações de funções a seguir:

Agregação	Entrada	Significado	Resultado
`min`	List<number>	Obtenha o mínimo de todos os números na lista.	number
`max`	List<number>	Obtenha o máximo de todos os números na lista.	number

Agregação	Entrada	Significado	Resultado
`avg`	List<number>	Obtenha a média de todos os números na lista.	number
`median`	List<number>	Obtenha o mediano de todos os números na lista.	number
`sum`	List<number>	Obtenha a soma de todos os números na lista.	number
`count`	List<?>	Obtenha o número de elementos na lista.	number
`stddev`	List<number>	Obtenha o desvio padrão de todos os números na lista.	number
`flatten`	List<List<?>>	Transforme uma coleção de listas aninhadas em uma lista única contendo todos os elementos.	List<?>
`set_intersection`	List<List<string>>	Veja as strings encontradas em todas as listas de string de uma coleção.	List<string>
Todas acima	List<List<?>>	Todas as operações em uma lista aninhada operam em cada sublista individualmente para produzir uma lista de resultados.	List<?>

A tabela a seguir ilustra algumas expressões de propriedade válidas que usam funções de agregação:

Expressão	Significado	Tipo resultante
flatten(teams[*].players.attributes[skill])	Os atributos "skill" de todos os jogadores na correspondência (não agrupados)	List<number>
avg(teams[red].players.attributes[skill])	A habilidade média dos jogadores da equipe red	number
avg (teams [*] .players.attributes [skill])	A habilidade média de cada equipe na correspondência	List<number>
avg(flatten(teams[*].players.attributes[skill]))	O nível de habilidade médio de todos os jogadores na	number

Expressão	Significado	Tipo resultante
	correspondência. Esta expressão obtém uma lista nivelada de habilidades do jogador e faz uma média.	
count(teams[red].players)	O número de jogadores da equipe red	number
count (teams[*].players)	O número de jogadores em cada equipe na correspondência	List<number>
max(avg(teams[*].players.attributes[skill]))	O nível mais alto de habilidade da equipe na correspondência	number

Eventos de marcação do FlexMatch

O GameLift FlexMatch emite eventos para cada ticket de matchmaking à medida que ele é processado. Você pode publicar esses eventos em um tópico do Amazon SNS, conforme descrito emConfigurar FlexMatch Notificações de eventos do (p. 42). Esses eventos também são emitidos para o Amazon CloudWatch Events quase em tempo real e com o melhor esforço.

Este tópico descreve a estrutura dos eventos FlexMatch e fornece um exemplo para cada tipo de evento. Para obter mais informações sobre os status do tíquete de marcação de jogos, consulteMatchmakingTicketnoReferência da API do Amazon GameLift.

MatchmakingSearching

O tíquete foi inserido na marcação de jogos. Isso inclui solicitações novas e aquelas que foram parte de uma correspondência proposta que falhou.

Recurso:ConfigurationArn

Detalhes: type, tickets, estimatedWaitMillis, gameSessionInfo

Exemplo

```
{
  "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

Uma correspondência em potencial foi criada. Ela é emitida para todas as novas correspondências potenciais, independentemente de a aceitação ser necessária.

Recurso: ConfigurationArn

Detalhes: type, tickets, acceptanceTimeout, acceptanceRequired, ruleEvaluationMetrics, gameSessionInfo, matchId

Exemplo

```
{
  "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

Os jogadores aceitaram uma correspondência em potencial. Este evento contém o status de aceitação atual de cada jogador na correspondência. Os dados ausentes significam que AcceptMatch não foi chamado para esse jogador.

Recurso: ConfigurationArn

Detalhes: type, tickets, matchId, gameSessionInfo

Exemplo

```
{
  "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

A aceitação da correspondência é concluída devido à aceitação, rejeição do jogador ou tempo limite da aceitação.

Recurso: ConfigurationArn

Detalhes: type, tickets, acceptance, matchId, gameSessionInfo

Exemplo

```
{
  "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

A marcação de jogos foi concluída com êxito e uma sessão de jogo foi criada.

Recurso: ConfigurationArn

Detalhes: type, tickets, matchId, gameSessionInfo

Exemplo

```
{
  "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

O tíquete da marcação de jogos falhou devido ao tempo limite.

Recurso: ConfigurationArn

Detalhes: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

Exemplo

```
{
  "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

O tíquete de matchmaking foi cancelado.

Recurso: ConfigurationArn

Detalhes: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

Exemplo


```
{
  "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

O tíquete da marcação de jogos encontrou um erro. Isso pode ser devido à fila de sessões de jogos não acessível ou a um erro interno.

Recurso: ConfigurationArn

Detalhes: type, tickets, ruleEvaluationMetrics, message, matchId, gameSessionInfo

Exemplo

```
{
  "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"
  }
}
```

Segurança com o FlexMatch

A segurança da nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de datacenters e arquiteturas de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. Para obter informações sobre como aplicar o modelo de responsabilidade compartilhada ao usar o FlexMatch, consulteSegurança no Amazon GameLift.

Notas de versão e versões do SDK do GameLift FlexMatch

As notas de release do GameLift fornecem detalhes sobre novos recursos, atualizações e correções do FlexMatch relacionadas ao serviço. Esta página também inclui o histórico de versões do GameLift SDK.

Recursos do desenvolvedor do GameLift

Para ver toda a documentação e recursos do desenvolvedor do GameLift, consulte aDocumentação do Amazon GameLiftPágina inicial do.

Glossário da AWS

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência geral da AWS.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.