aws

FlexMatch Guía para desarrolladores

Amazon GameLift Servers

Version

Amazon GameLift Servers: FlexMatch Guía para desarrolladores

Table of Contents

¿Qué es Amazon GameLift ServersFlexMatch?

Amazon GameLift ServersFlexMatches un servicio de emparejamiento personalizable para juegos multijugador. Con FlexMatch él, puedes crear un conjunto personalizado de reglas que definan cómo será una partida multijugador para tu juego y determinen cómo evaluar y seleccionar a los jugadores compatibles para cada partida. También puede afinar los aspectos clave del algoritmo de emparejamiento para que se adapten a las necesidades de su juego.

Úsalo FlexMatch como un servicio de búsqueda de parejas independiente o integrado con una solución de alojamiento de Amazon GameLift Servers juegos. Por ejemplo, puedes implementarla FlexMatch como una función independiente con juegos con una peer-to-peer arquitectura o juegos que usen otras soluciones de computación en la nube. O bien, puedes añadir FlexMatch a tus contenedores Amazon GameLift Servers gestionados EC2 o gestionados un alojamiento local con Amazon GameLift Servers Anywhere. Esta guía proporciona información detallada sobre cómo crear un sistema de FlexMatch emparejamiento para tu situación particular.

FlexMatchte da la flexibilidad de establecer prioridades de emparejamiento en función de los requisitos de tu juego. Por ejemplo, puede hacer lo siguiente:

- Encuentre un equilibrio entre la velocidad y la calidad de los emparejamientos. Establezca reglas de juego para encontrar rápidamente emparejamientos que sean lo suficientemente buenos, o haga que los jugadores esperen un poco más para encontrar la mejor combinación posible y así disfrutar de una experiencia de juego óptima.
- Realice emparejamientos en función de jugadores o equipos bien emparejados. Cree un emparejamiento en el que todos los jugadores tengan características similares, como la habilidad o la experiencia. También puede formar emparejamientos en los que las características combinadas de cada equipo cumplan con un criterio común.
- Priorice el efecto de la latencia de los jugadores en el emparejamiento. ¿Quiere establecer un límite estricto de latencia para todos los jugadores o prefiere aceptar latencias más altas siempre que todos los participantes tengan una latencia similar?

> ¿Estás listo para empezar a trabajar con él? FlexMatch
>
> Para step-by-step obtener información sobre cómo poner en marcha tu juegoFlexMatch, consulta los siguientes temas:

- Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers
- Hoja de ruta: Cree una solución de emparejamiento independiente con FlexMatch

FlexMatchCaracterísticas principales

Las siguientes funciones están disponibles en todos los FlexMatch escenarios, ya sea que las utilices FlexMatch como un servicio independiente o como alojamiento de Amazon GameLift Servers juegos.

- Emparejamiento de jugadores personalizable. Diseñe y compile emparejadores que se adapten a todos los modos de juego que ofrece a los jugadores. Cree un conjunto de reglas personalizadas para evaluar los atributos de los jugadores clave (por ejemplo, nivel de habilidad o rol) y datos de latencia geográfica para crear los mejores emparejamientos posibles para el juego.
- Emparejamiento basado en latencia. Proporcione datos de latencia de los jugadores y cree reglas de emparejamiento que exijan que los jugadores de un emparejamiento tengan tiempos de respuesta similares. Esta característica resulta útil cuando los grupos de emparejamiento de jugadores abarcan varias regiones geográficas.
- Compatibilidad con emparejamientos de hasta 200 jugadores. Cree emparejamientos de hasta 40 jugadores mediante las reglas de emparejamiento personalizadas para el juego. Cree emparejamientos de hasta 200 jugadores mediante un proceso de emparejamiento que utiliza un proceso de emparejamiento personalizado simplificado para que los tiempos de espera de los jugadores sean gestionables.
- Aceptación del jugador. Solicite a los jugadores que opten por participar en un emparejamiento propuesto antes de finalizarlo e iniciar una sesión de juego. Usa esta función para iniciar tu proceso de aceptación personalizado e informar sobre las respuestas de los jugadores FlexMatch antes de iniciar una nueva sesión de juego para el partido. Si no todos los jugadores aceptan un emparejamiento, el emparejamiento propuesto falla y los jugadores que sí lo acepten volverán automáticamente al grupo de emparejamientos.
- Compatibilidad con grupos de jugadores. Genere emparejamientos para grupos de jugadores que quieran jugar juntos en el mismo equipo. FlexMatchÚsala para encontrar jugadores adicionales con los que completar la partida según sea necesario.
- Reglas de emparejamiento ampliables. Flexibilice gradualmente los requisitos de emparejamiento después de que haya pasado cierto tiempo sin encontrar un emparejamiento correcto. La expansión de reglas le permite decidir dónde y cuándo flexibilizar las reglas iniciales del

emparejamiento para que los jugadores puedan entrar en los juegos a los que se puede jugar con mayor rapidez.

- Reposición de emparejamientos. Llene las ranuras de jugadores vacías de una sesión de juego existente con jugadores nuevos bien emparejados. Personalice cuándo y cómo solicitar nuevos jugadores, y utilice las mismas reglas de emparejamiento personalizadas para encontrar más jugadores.

FlexMatchcon Amazon GameLift Servers alojamiento

FlexMatchofrece las siguientes funciones adicionales para usarlas con los juegos con los que alojasAmazon GameLift Servers. Esto incluye juegos con servidores de juegos personalizados o Amazon GameLift ServersRealtime.

- Ubicación de sesión de juego. Cuando una partida se realiza correctamente, solicita FlexMatch automáticamente una nueva ubicación en la sesión de juego aAmazon GameLift Servers. Los datos generados durante el emparejamiento, incluidas las asignaciones de jugadores IDs y equipos, se proporcionan al servidor del juego para que pueda utilizarlos para iniciar la sesión de juego de la partida. FlexMatcha continuación, transfiere la información de conexión de la sesión de juego para que los clientes del juego puedan unirse a la partida. Para minimizar la latencia que experimentan los jugadores en una partida, la ubicación de las sesiones de juego también Amazon GameLift Servers puede utilizar los datos de latencia de los jugadores regionales, si los proporciona.
- Reposición automática de emparejamientos. Con esta función habilitada, envía FlexMatch automáticamente una solicitud de reposición de partidas cuando se inicia una nueva sesión de juego con los espacios vacíos para los jugadores. El sistema de emparejamiento inicia el proceso de ubicación de la sesión de juego con un número mínimo de jugadores y, a continuación, llena rápidamente las ranuras restantes. No puede utilizar la reposición automática para reemplazar a los jugadores que abandonen una sesión de juego emparejada.

Si los usa Amazon GameLift Servers FleetIQ con juegos que están alojados con recursos de Amazon Elastic Compute Cloud (Amazon EC2), impleméntelos FlexMatch como un servicio independiente.

Precios para Amazon GameLift ServersFlexMatch

Amazon GameLift Serverslos cargos de las instancias se calculan según la duración del uso y el ancho de banda según la cantidad de datos transferidos. Si alojas tus juegos en Amazon GameLift

Servers servidores, el FlexMatch uso está incluido en las tarifasAmazon GameLift Servers. Si alojas tus juegos en otra solución de servidor, el FlexMatch uso se cobra por separado. Para obtener una lista completa de los costos y precios de Amazon GameLift Servers, consulte Precios de Amazon GameLift Servers.

Para obtener información sobre cómo calcular el coste de alojar tus juegos o crear parejas con ellosAmazon GameLift Servers, consulta Cómo generar estimaciones de Amazon GameLift Servers precios, donde se describe cómo utilizar la Calculadora de precios de AWS.

Cómo Amazon GameLift ServersFlexMatch funciona

En este tema se proporciona una descripción general del Amazon GameLift Servers FlexMatch servicio, incluidos los componentes principales de un FlexMatch sistema y la forma en que interactúan.

Puedes usarlo FlexMatch con juegos que utilizan alojamiento Amazon GameLift Servers gestionado o con juegos que utilizan otra solución de alojamiento. Los juegos que se alojan en Amazon GameLift Servers servidores, incluso Amazon GameLift ServersRealtime, utilizan el Amazon GameLift Servers servicio integrado para localizar automáticamente los servidores de juegos disponibles e iniciar las sesiones de juego para los partidos. Los juegos que se utilizan FlexMatch como un servicio independiente, incluido Amazon GameLift Servers FleetiQ, deben coordinarse con el sistema de alojamiento existente para asignar los recursos de alojamiento e iniciar las sesiones de juego para los partidos.

Para obtener instrucciones detalladas sobre cómo configurar FlexMatch tus juegos, consulta. Introducción al FlexMatch

Componentes de emparejamiento

Un sistema de FlexMatch emparejamiento incluye algunos o todos los siguientes componentes.

Componentes de Amazon GameLift Servers

Estos son Amazon GameLift Servers recursos que controlan la forma en que el FlexMatch servicio realiza el emparejamiento para tu juego. Se crean y mantienen mediante Amazon GameLift Servers herramientas, incluidas la consola y la AWS CLI o, alternativamente, mediante programación mediante el AWS SDK para. Amazon GameLift Servers

- FlexMatchConfiguración de emparejamiento (también llamada emparejador): un emparejador es un conjunto de valores de configuración que personaliza el proceso de emparejamiento de

tu juego. Un juego puede tener varios emparejadores, cada uno configurado para diferentes modos de juego o experiencias, según sea necesario. Cuando tu juego envía una solicitud de emparejamiento aFlexMatch, especifica qué emparejador usar.

- FlexMatchconjunto de reglas de emparejamiento: un conjunto de reglas contiene toda la información necesaria para evaluar a los jugadores en busca de posibles coincidencias y aprobarlas o rechazarlas. El conjunto de reglas define la estructura del equipo de un emparejamiento, declara los atributos de los jugadores que se utilizan para la evaluación y proporciona reglas que describen los criterios para que un emparejamiento sea aceptable. Las reglas se pueden aplicar a jugadores individuales, equipos o a todo el emparejamiento. Por ejemplo, una regla puede requerir que todos los jugadores del emparejamiento elijan el mismo mapa de juego o que todos los equipos tengan un promedio de habilidad similar.
- Amazon GameLift Serverscola de sesiones de juego (solo para los anfitriones FlexMatch Amazon GameLift Servers gestionados): una cola de sesión de juego localiza los recursos de alojamiento disponibles e inicia una nueva sesión de juego para la partida. La configuración de la cola determina dónde buscar Amazon GameLift Servers los recursos de alojamiento disponibles y cómo seleccionar el mejor anfitrión disponible para una partida.

Componentes personalizados

Los siguientes componentes incluyen las funciones necesarias para un FlexMatch sistema completo y que debes implementar en función de la arquitectura del juego.

- Interfaz de jugador para el emparejamiento: esta interfaz permite a los jugadores unirse a un emparejamiento. Como mínimo, inicia una solicitud de emparejamiento a través del componente del servicio de emparejamiento del cliente y proporciona datos específicos del jugador, como el nivel de habilidad y los datos de latencia, según sea necesario para el proceso de emparejamiento.

  > **Note**
  >
  > Como práctica recomendada, la comunicación con el FlexMatch servicio debe realizarse mediante un servicio de back-end, no desde un cliente de juego.

- Servicio de emparejamiento de clientes: este servicio archiva las solicitudes de unión de los jugadores desde la interfaz del jugador, genera las solicitudes de emparejamiento y las envía al servicio. FlexMatch En el caso de las solicitudes en proceso, supervisa los eventos de emparejamiento, rastrea el estado del emparejamiento y toma las medidas necesarias. En función de cómo administre el alojamiento de las sesiones de juego en el juego, este servicio puede

devolver la información de conexión de la sesión de juego a los jugadores. Este componente usa el AWS SDK con la Amazon GameLift Servers API para comunicarse con el FlexMatch servicio.

- Servicio de ubicación de partidos (solo FlexMatch como servicio independiente): este componente funciona con tu sistema de alojamiento de juegos existente para localizar los recursos de alojamiento disponibles e iniciar nuevas sesiones de juego para los partidos. El componente debe obtener los resultados del emparejamiento y extraer la información necesaria para iniciar una nueva sesión de juego, incluidos los jugadores IDs, los atributos y las asignaciones de equipo de todos los jugadores de la partida.

FlexMatchproceso de emparejamiento

En este tema se describe la secuencia de eventos en un escenario de emparejamiento básico, incluidas las interacciones entre los distintos componentes del juego y el FlexMatch servicio.

Paso 1: Solicita el emparejamiento para los jugadores

Un jugador que utilice el cliente de juegos hace clic en el botón “Unirse a un juego”. Esta acción hace que el servicio de emparejamiento de su cliente envíe una solicitud de emparejamiento a. FlexMatch La solicitud identifica el FlexMatch emparejador que se utilizará al cumplir con la solicitud. La solicitud también incluye la información del jugador que necesite el emparejador personalizado, como el nivel de habilidad, las preferencias de juego o los datos de latencia geográfica. Puede realizar solicitudes de emparejamiento para uno o varios jugadores.

Paso 2: Agrega las solicitudes al grupo de búsqueda de parejas

Cuando FlexMatch recibe la solicitud de emparejamiento, genera un ticket de emparejamiento y lo agrega al grupo de boletos del emparejador. El ticket permanece en el grupo hasta que se realice el emparejamiento o se alcance el límite de tiempo máximo. El servicio de emparejamiento de clientes recibe notificaciones periódicas sobre los eventos de emparejamiento, incluidos los cambios en el estado de los tickets.

Paso 3: Construye una partida

Tu FlexMatch emparejador ejecuta continuamente el siguiente proceso con todos los boletos de su grupo:

1. El emparejador ordena el grupo por la antigüedad de los tickets y, a continuación, comienza a crear un posible emparejamiento empezando por el ticket más antiguo.

2. El emparejador añade un segundo ticket al posible emparejamiento y evalúa el resultado según las reglas de emparejamiento personalizadas. Si el posible emparejamiento pasa la evaluación, los jugadores del ticket pasan a formar parte de un equipo.
3. El emparejador añade el siguiente ticket de la secuencia y repite el proceso de evaluación. Cuando se hayan ocupado todas las ranuras para los jugadores, el emparejamiento estará listo.

En el caso de emparejamientos grandes (de 41 a 200 jugadores), el sistema de emparejamiento utiliza una versión modificada del proceso descrito anteriormente, de forma que se puedan compilar emparejamientos en un plazo de tiempo razonable. En lugar de evaluar cada ticket de forma individual, el emparejador divide un grupo de tickets preestablecido en posibles emparejamientos y, a continuación, equilibra cada emparejamiento en función de una característica del jugador que haya especificado. Por ejemplo, un emparejador podría clasificar previamente los tickets en función de ubicaciones similares de baja latencia y, después, utilizar el equilibrio posterior al emparejamiento para asegurarse de que los equipos estén emparejados equitativamente según la habilidad del jugador.

Paso 4: Reporta los resultados del emparejamiento

Cuando se encuentra un emparejamiento aceptable, se actualizan todos los tickets emparejados y se genera un evento de emparejamiento correcto para cada ticket de emparejamiento.

- FlexMatchcomo servicio independiente: Tu juego recibe los resultados de las partidas en un evento de emparejamiento exitoso. Los datos de los resultados incluyen una lista de todos los jugadores emparejados y las asignaciones de sus equipos. Si las solicitudes de emparejamiento contienen información sobre la latencia de los jugadores, los resultados también sugieren una ubicación geográfica óptima para el emparejamiento.
- FlexMatchcon una solución de Amazon GameLift Servers alojamiento: los resultados de las partidas se pasan automáticamente a una Amazon GameLift Servers cola para situarlos en las sesiones de juego. El emparejador determina qué cola se utiliza para la ubicación de las sesiones de juego.

Paso 5: Inicia una sesión de juego para el partido

Una vez que el emparejamiento propuesto se haya formado satisfactoriamente, se iniciará una nueva sesión de juego. Tus servidores de juego deben poder utilizar los datos de los resultados del emparejamiento, incluidas las asignaciones de jugadores IDs y equipos, al configurar una sesión de juego para la partida.

- FlexMatchcomo servicio independiente: tu servicio personalizado de ubicación de partidas obtiene los datos de los resultados de las partidas de los eventos de matchmaking que se han

llevado a cabo con éxito y se conecta a tu sistema de ubicación de sesiones de juego existente para localizar un recurso de alojamiento disponible para la partida. Una vez encontrado un recurso de alojamiento, el servicio de ubicación de emparejamientos se coordina con el sistema de alojamiento actual para iniciar una nueva sesión de juego y obtener información de conexión.

- FlexMatchcon una solución de Amazon GameLift Servers alojamiento: la cola de sesiones de juego localiza el mejor servidor de juego disponible para la partida. En función de cómo esté configurada la cola, intentará situar la sesión de juego con los recursos de menor costo y en un lugar en el que los jugadores experimenten una latencia baja (si se proporcionan datos sobre la latencia de los jugadores). Una vez finalizada correctamente la sesión de juego, el Amazon GameLift Servers servicio solicita al servidor del juego que inicie una nueva sesión de juego y le transmite los resultados del emparejamiento y otros datos opcionales del juego.

Paso 6: Conecta a los jugadores al partido

Tras iniciar una sesión de juego, los jugadores se conectan a la sesión, reclaman su asignación de equipo y comienzan a jugar.

- FlexMatchcomo servicio independiente: tu juego utiliza el sistema de gestión de sesiones de juego existente para proporcionar información de conexión a los jugadores.
- FlexMatchcon una solución de Amazon GameLift Servers alojamiento: si la sesión de juego se coloca correctamente, FlexMatch actualiza todas las entradas coincidentes con la información de conexión de la sesión de juego y un identificador de sesión del jugador.

FlexMatchcompatible Regiones de AWS

Si utilizas una solución FlexMatch de Amazon GameLift Servers alojamiento, puedes organizar sesiones de juego combinadas en cualquier ubicación en la que estés alojando juegos. Consulta la lista completa Regiones de AWS y las ubicaciones de Amazon GameLift Servers alojamiento.

Introducción al FlexMatch

Usa los recursos de esta sección para ayudarte a empezar a crear un sistema de emparejamiento con FlexMatch.

Temas

Configura un formulario Cuenta de AWSFlexMatch

Amazon GameLift Servers FlexMatch es un AWS servicio y debe tener una AWS cuenta para utilizar este servicio. Crear una AWS cuenta es gratuito. Para obtener más información sobre lo que puede hacer con una cuenta de AWS , consulte Introducción a AWS.

Si estás usando FlexMatch con otros Amazon GameLift Servers soluciones, consulte los siguientes temas:

- Configurar el acceso para Amazon GameLift Servers hospedaje
- Configurar el acceso al alojamiento con Amazon GameLift Servers FleetIQ

Para configurar tu cuenta para Amazon GameLift Servers

1. Obtenga una cuenta. Abra Amazon Web Services y elija Iniciar sesión en la consola. Siga las instrucciones para crear una cuenta nueva o inicie sesión en una existente.
2. Configure un grupo de usuarios administrativos. Abra la consola de servicio de AWS Identity and Access Management (IAM) y siga los pasos para crear o actualizar usuarios o grupos de usuarios. IAM gestiona el acceso a sus AWS servicios y recursos. Todos los que acceden a su FlexMatch recursos, utilizando el Amazon GameLift Servers consola o llamando Amazon GameLift Servers APIs, debe tener acceso explícito. Para obtener instrucciones detalladas sobre el uso de la consola (o la AWS CLI u otras herramientas) para configurar grupos de usuarios, consulte Creación de usuarios de IAM.
3. Asocie una política de permisos a un usuario o grupo de su cuenta.El acceso a AWS los servicios y recursos se gestiona adjuntando una política de IAM a un usuario o grupo de

usuarios. Las políticas de permisos especifican un conjunto de AWS servicios y acciones a los que el usuario debe tener acceso.

En Amazon GameLift Servers, debe crear una política de permisos personalizada y adjuntarla a cada usuario o grupo de usuarios. Una política es un documento JSON. Utilice el siguiente ejemplo para crear su política.

El siguiente ejemplo ilustra una política de permisos en línea con permisos administrativos para todos Amazon GameLift Servers recursos y acciones. Puede optar por limitar el acceso especificando únicamente FlexMatch-artículos específicos.

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

Hoja de ruta: Cree una solución de emparejamiento independiente con FlexMatch

En este tema se describe el proceso de integración completo para implementarlo FlexMatch como un servicio de emparejamiento independiente. Usa este proceso si tu juego multijugador se aloja con peer-to-peer hardware local configurado a medida u otras primitivas de computación en la nube. Este proceso también se usa con Amazon GameLift ServersFleetIQ, que es una solución de optimización de alojamiento para juegos alojados en Amazon EC2. Si alojas tu juego mediante alojamiento Amazon GameLift Servers gestionado (incluido Amazon GameLift ServersRealtime), consultaHoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers.

Antes de iniciar la integración, debes tener una AWS cuenta y configurar los permisos de acceso al Amazon GameLift Servers servicio. Para obtener más información, consulte Configura un formulario Cuenta de AWSFlexMatch. Todas las tareas esenciales relacionadas con la creación y administración de Amazon GameLift Servers FlexMatch emparejadores y conjuntos de reglas se pueden realizar desde la Amazon GameLift Servers consola.

1. Crea un conjunto de reglas FlexMatch de emparejamiento. Su conjunto de reglas personalizado proporciona instrucciones completas sobre cómo crear un emparejamiento. En él, debe definir la estructura y el tamaño de cada equipo. También estableces un conjunto de requisitos que debe cumplir una partida para ser válida, que FlexMatch suele incluir o excluir a los jugadores de una partida. Estos requisitos pueden aplicarse a jugadores individuales. También puedes personalizar el FlexMatch algoritmo del conjunto de reglas, por ejemplo, para organizar partidas grandes con hasta 200 jugadores. Consulte estos temas:

   - Construye un FlexMatch conjunto de reglas
   - FlexMatchejemplos de conjuntos de reglas

2. Configuración de notificaciones de eventos de emparejamiento. Usa las notificaciones para hacer un seguimiento de la actividad de FlexMatch emparejamiento, incluido el estado de las solicitudes de partidos pendientes. Este es el mecanismo que se utiliza para proporcionar los resultados de un emparejamiento propuesto. Dado que las solicitudes de emparejamiento son asíncronas, necesitará una forma de controlar el estado de las solicitudes. Usar las notificaciones es la opción preferida. Consulte estos temas:

   - Configuración FlexMatch notificaciones de eventos
   - FlexMatch eventos de emparejamiento

3. Configura una configuración de FlexMatch emparejamiento. También denominado “emparejador”, este componente recibe solicitudes de emparejamiento y las procesa. Para configurar un emparejador, especifique un conjunto de reglas, un objetivo de notificación y un tiempo máximo de espera. También puede habilitar características opcionales. Consulte estos temas:

   - Diseña un FlexMatch emparejador
   - Creación de una configuración de emparejamiento

4. Creación de un servicio de emparejamiento de clientes. Cree o amplíe un servicio de cliente de juego con funciones para crear y enviar solicitudes de emparejamiento. FlexMatch Para crear solicitudes de emparejamiento, este componente debe tener mecanismos para obtener los datos de los jugadores requeridos por el conjunto de reglas de emparejamiento y, opcionalmente, la información de latencia regional. También debe tener un método para crear y asignar un ticket único IDs para cada solicitud. También puede optar por compilar un flujo de trabajo de aceptación de jugadores que requiera que los jugadores opten por participar en un emparejamiento propuesto. Este servicio también debe supervisar los eventos de

emparejamiento para obtener los resultados del emparejamiento e iniciar la ubicación de las sesiones de juego para que los emparejamientos sean correctos. Consulte este tema:

- Añadir FlexMatch a un cliente de juego

5. Creación de un servicio de ubicación de emparejamientos. Cree un mecanismo que funcione con el sistema de alojamiento de juegos existente para localizar los recursos de alojamiento disponibles e iniciar nuevas sesiones de juego para que los emparejamientos sean correctos. Este componente debe poder utilizar la información de los resultados de los emparejamientos para obtener un servidor de juegos disponible e iniciar una nueva sesión de juego para el emparejamiento. También puede implementar un flujo de trabajo para completar las solicitudes de reposición, que utilice el emparejamiento para rellenar las ranuras abiertas en las sesiones de juego emparejadas que ya estén en marcha.

Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers

FlexMatchestá disponible con el Amazon GameLift Servers alojamiento gestionado para servidores de juegos personalizados y Amazon GameLift ServersRealtime. Para añadir el emparejamiento de FlexMatch a un juego, realice las siguientes tareas.

- Configurar un creador de emparejamientos. Un creador de emparejamientos recibe solicitudes de emparejamiento de jugadores y las procesa. Agrupa a los jugadores en función de un conjunto de reglas definidas y crea nuevas sesiones de juego y de jugadores por cada emparejamiento correcto. Siga estos pasos para configurar un creador de emparejamientos:
  - Crear un conjunto de reglas. Un conjunto de reglas indica al creador de emparejamientos cómo conformar un emparejamiento válido. Especifica la estructura del equipo y cómo evaluar a los jugadores para su inclusión en un emparejamiento. Consulte estos temas:
    - Construye un FlexMatch conjunto de reglas
    - FlexMatchejemplos de conjuntos de reglas
  - Crear una cola de sesión de juego. Una cola localiza la mejor región para cada emparejamiento y crea una sesión de juego nueva en esa región. Utilice una cola existente o cree una nueva para la creación de emparejamientos. Consulte este tema:
    - Creación de una cola

  - Configurar notificaciones (opcional). Dado que las solicitudes de emparejamiento son asíncronas, necesitará una forma de controlar el estado de las solicitudes. Las notificaciones son la opción preferida. Consulte este tema:
    - Configuración FlexMatch notificaciones de eventos
  - Configurar un creador de emparejamientos. Tras contar con un conjunto de reglas, una cola y un destino de notificaciones, puede crear la configuración para su creador de emparejamientos. Consulte estos temas:
    - Diseña un FlexMatch emparejador
    - Creación de una configuración de emparejamiento
- Integrar FlexMatch en el servicio del cliente de juego. Añada la funcionalidad a su servicio de cliente de juego para iniciar nuevas sesiones de juegos con la creación de emparejamientos. Las solicitudes para la creación de emparejamientos especifican el creador de emparejamientos que se debe utilizar y proporcionan los datos de los jugadores necesarios para el emparejamiento. Consulte este tema:
  - Añadir FlexMatch a un cliente de juego
- Integrar FlexMatch en el servidor de juegos. Añada la funcionalidad al servidor de juegos para iniciar sesiones de juegos creadas mediante la creación de emparejamientos. Las solicitudes para este tipo de sesión de juego incluyen información específica del emparejamiento, incluidos los jugadores y las asignaciones de equipo. El servidor de juegos necesita acceder a esta información y utilizarla cuando crea una sesión de juego para el emparejamiento. Consulte este tema:
  - Añadir FlexMatch a un servidor Amazon GameLift Servers de juegos hospedado
- Configurar la reposición de FlexMatch (opcional). Solicite emparejamientos de jugadores adicionales para completar las ranuras de jugadores abiertas en los juegos existentes. Puede activar la reposición automática para que Amazon GameLift Servers administre las solicitudes de reposición. También puede administrar manualmente la reposición añadiendo la funcionalidad a su servicio de cliente de juego o servidor de juegos para iniciar las solicitudes de reposición de emparejamiento. Consulte este tema:
  - Rellena los juegos existentes con FlexMatch

> **Note**
>
> FlexMatchEl relleno no está disponible actualmente para los juegos que se utilizan Amazon GameLift ServersRealtime.

Construir un casamentero Amazon GameLift ServersFlexMatch

En esta sección, se describen los elementos clave de un emparejador y cómo crear y personalizar uno para tu juego. Esto incluye configurar una configuración de emparejamiento y un conjunto de reglas de emparejamiento.

Crear tu matchmaker es el primer paso de las hojas de ruta: FlexMatch

- Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers
- Hoja de ruta: Cree una solución de emparejamiento independiente con FlexMatch

Un FlexMatch emparejador hace el trabajo de crear una partida de juego. Administra el conjunto de solicitudes de emparejamiento recibidas, procesa y selecciona a los jugadores para encontrar los mejores grupos de jugadores posibles y forma equipos para una partida. En el caso de los juegos que se utilizan Amazon GameLift Servers como anfitriones, también organiza e inicia una sesión de juego para el partido.

FlexMatch combina el servicio de emparejamiento con un motor de reglas personalizable. Permite diseñar el emparejamiento de jugadores en función de los atributos de los jugadores y los modos de juego relevantes para el juego, así como confiar en FlexMatch para que administre las cuestiones prácticas de formación de grupos de jugadores y su ubicación en los juegos. Vea más detalles sobre el emparejamiento personalizado en FlexMatchejemplos de conjuntos de reglas.

Tras formar una partida, FlexMatch proporciona los datos de la partida para la ubicación de la sesión de juego. En el caso de los juegos que se utilizan Amazon GameLift Servers como alojamiento, FlexMatch envía una solicitud de ubicación de la sesión de juego con los jugadores coincidentes a la cola de una sesión de juego. La cola busca recursos de alojamiento disponibles en sus flotas de Amazon GameLift Servers y comienza una nueva sesión de juego para el emparejamiento. En el caso de los juegos que utilizan otra solución de alojamiento, FlexMatch proporciona los datos de las partidas para que los proporciones a tu propio componente de ubicación de sesiones de juego.

Para obtener una descripción detallada de cómo los creadores de emparejamientos de FlexMatch procesan las solicitudes de emparejamiento que reciben, consulte FlexMatchproceso de emparejamiento.

Temas

Diseña un FlexMatch emparejador

En este tema se proporciona orientación sobre cómo diseñar un emparejador que se adapte a su juego.

Temas

Configuración de un emparejador básico

Como mínimo, un emparejador necesita los siguientes elementos:

- El conjunto de reglas determina el tamaño y el alcance de los equipos para un emparejamiento, y define un conjunto de reglas para utilizar a la hora de evaluar los jugadores para un emparejamiento. Cada creador de emparejamientos está configurado para utilizar un conjunto de reglas. Consulte Construye un FlexMatch conjunto de reglas y FlexMatchejemplos de conjuntos de reglas.
- El destinatario de las notificaciones recibe todas las notificaciones de los eventos de emparejamiento. Debe configurar un tema de Amazon Simple Notification Service (SNS) y, a continuación, añadir el ID de tema al emparejador. Consulte más información sobre la configuración de notificaciones en Configuración FlexMatch notificaciones de eventos.
- El tiempo de espera de la solicitud determina el periodo durante el cual las solicitudes de emparejamiento pueden permanecer en el grupo de solicitudes y ser evaluadas para comprobar los emparejamientos potenciales. Cuando una solicitud agota el tiempo de espera, significa que no se ha podido realizar un emparejamiento y se retira del grupo.
- Cuando se usa FlexMatch por Amazon GameLift Servers en el alojamiento gestionado, la cola de sesiones de juego busca los mejores recursos disponibles para organizar una sesión de juego durante el partido e inicia una nueva sesión de juego. Cada cola está configurada con una lista de

ubicaciones y tipos de recursos (incluidas las instancias de spot o bajo demanda) que determinan dónde se pueden ubicar las sesiones de juego. Para obtener más información sobre las colas, consulte Uso de colas con varias ubicaciones.

Elección de una ubicación para el emparejador

Decida dónde quiere que se lleve a cabo la actividad de emparejamiento y cree la configuración y el conjunto de reglas de emparejamiento en esa ubicación. Amazon GameLift Servers mantiene reservas de entradas para las solicitudes de partidas de tu juego, donde se clasifican y evalúan para determinar si son viables. Tras hacer una partida, Amazon GameLift Servers envía los detalles de la partida para la ubicación de la sesión de juego. Puede ejecutar las sesiones de juego emparejadas en cualquier ubicación compatible con la solución de alojamiento.

Consulta FlexMatchcompatible Regiones de AWS las ubicaciones en las que puedes crear FlexMatch recursos.

A la hora de elegir Región de AWS uno para tu matchmaker, ten en cuenta cómo la ubicación puede afectar al rendimiento y cómo puede optimizar la experiencia de juego para los jugadores. Recomendamos que siga las siguientes prácticas recomendadas:

- Coloca un emparejador en una ubicación que esté cerca de tus jugadores y del servicio de atención al cliente que envía FlexMatch solicitudes de emparejamiento. Este enfoque reduce el efecto de latencia en el flujo de trabajo de las solicitudes de emparejamiento y lo hace más eficiente.
- Si el juego llega a una audiencia global, considere la posibilidad de crear emparejadores en varias ubicaciones y enviar las solicitudes de emparejamiento al emparejador que esté más cerca del jugador. Además de aumentar la eficiencia, esto hace que se acumulen grupos de tickets que formar con jugadores que se encuentran geográficamente cerca unos de otros, lo que mejora la capacidad del emparejador para emparejar jugadores en función de los requisitos de latencia.
- Cuando se usa FlexMatch por Amazon GameLift Servers alojamiento gestionado, coloca tu emparejador y la cola de sesiones de juego que utiliza en la misma ubicación. Esto ayuda a minimizar la latencia de la comunicación entre el creador de emparejamientos y la cola.

Adición de elementos opcionales

Además de estos requisitos mínimos, puede configurar su creador de emparejamientos con las siguientes opciones adicionales. Si estás usando FlexMatch con un Amazon GameLift Servers

solución de alojamiento, muchas funciones están integradas. Si estás usando FlexMatch como servicio de búsqueda de pareja independiente, es posible que desee incorporar estas funciones a su sistema.

Aceptación del jugador

Puede configurar un emparejador para que exija que todos los jugadores seleccionados para un emparejamiento acepten la participación. Si el sistema requiere la aceptación, se debe dar a todos los jugadores la opción de aceptar o rechazar un emparejamiento propuesto. Un emparejamiento debe recibir aceptaciones de todos los jugadores en el emparejamiento propuesto para que pueda completarse. Si un jugador rechaza o no acepta un emparejamiento, se descarta el emparejamiento propuesto y los tickets se gestionan de la siguiente manera. Los tickets en los que todos los jugadores aceptaron el emparejamiento se devuelven al grupo de emparejamiento para continuar su procesamiento. Los tickets en los que al menos un jugador rechazó el emparejamiento o no respondió pasan a un estado de error y dejan de procesarse. La aceptación del jugador requiere un plazo. Todos los jugadores deben aceptar un emparejamiento propuesto dentro del plazo establecido para que el emparejamiento continúe.

Modo de reposición

Uso FlexMatch rellena tus sesiones de juego para que tus sesiones de juego estén repletas de nuevos jugadores bien emparejados durante toda la sesión de juego. Al tramitar las solicitudes de relleno, FlexMatch usa el mismo emparejador que se usó para emparejar a los jugadores originales. Puede personalizar la forma en que se priorizan los tickets de reposición con los tickets para nuevos emparejamientos colocando los tickets de reposición al principio o al final de la fila. Esto significa que, a medida que entran nuevos jugadores en el grupo de emparejamiento, es más o menos probable que se les coloque en un juego ya existente que en uno recién formado.

El relleno manual está disponible tanto si tu juego usa FlexMatch con gestionado Amazon GameLift Servers hosting o con otras soluciones de hosting. La reposición manual le ofrece la flexibilidad de decidir cuándo activar una solicitud de reposición. Por ejemplo, es posible que desee añadir jugadores nuevos solo durante determinadas fases del juego o solo cuando existan determinadas condiciones.

El relleno automático solo está disponible para los juegos que se utilizan de forma gestionada Amazon GameLift Servers alojamiento. Con esta función habilitada, si una sesión de juego comienza con tragaperras abiertas para jugadores, Amazon GameLift Servers comienza a generar automáticamente las solicitudes de reposición. Esta característica le permite configurar el sistema de emparejamiento para que los nuevos juegos se inicien con un número mínimo de

jugadores y, después, se llenen rápidamente a medida que entran nuevos jugadores en el grupo de emparejamiento. Puede desactivar la reposición automática en cualquier momento durante la duración de la sesión de juego.

Propiedades del juego

Para juegos que utilizan FlexMatch por Amazon GameLift Servers en el alojamiento gestionado, puedes proporcionar información adicional para pasarla a un servidor de juegos cada vez que se solicite una nueva sesión de juego. Esta puede ser una forma útil de modificar las configuraciones de modo de juego necesarias para iniciar una sesión de juego según el tipo de emparejamientos que se estén creando. Todas las sesiones de juego de los emparejamientos creados por un emparejador reciben el mismo conjunto de propiedades del juego. Puede variar la información de las propiedades del juego creando diferentes configuraciones de emparejamiento.

Ranuras de jugadores reservadas

Puede designar que determinadas ranuras de jugadores de cada emparejamiento se reserven y se rellenen en otro momento. Esto se hace configurando la propiedad "additional player count" de una configuración de emparejamiento.

Datos de eventos personalizados

Utilice esta propiedad para incluir un conjunto de información personalizada en todos los eventos relacionados con el emparejamiento para el creador de emparejamientos. Esta característica puede resultar útil para realizar un seguimiento de determinada actividad exclusiva de su juego, incluido el seguimiento del desempeño de sus creadores de emparejamientos.

Construye un FlexMatch conjunto de reglas

Cada FlexMatch El casamentero debe tener un conjunto de reglas. El conjunto de reglas determina los dos elementos clave de un emparejamiento: la estructura y el tamaño del equipo del juego, y cómo agrupar los jugadores para lograr el mejor emparejamiento posible.

Por ejemplo, un conjunto de reglas podría describir un emparejamiento de este tipo: Crear un emparejamiento con dos equipos de cinco jugadores cada uno (un equipo de defensores y un equipo de invasores). Un equipo puede tener jugadores novatos y experimentados, pero la habilidad media de los dos equipos debe estar dentro de los 10 puntos de diferencia entre sí. Si después de 30 segundos no se ha realizado ningún emparejamiento, reduzca gradualmente los requisitos de habilidad.

Los temas de esta sección describen cómo diseñar y crear un conjunto de reglas de creación de emparejamientos. Al crear un conjunto de reglas, puede usar cualquiera de las dos Amazon GameLift Servers consola o AWS CLI.

Temas

Diseñar un conjunto de FlexMatch reglas

En este tema se trata la estructura básica de un conjunto de reglas y cómo compilar un conjunto de reglas para emparejamientos pequeños de hasta 40 jugadores. Un conjunto de reglas de emparejamiento hace dos cosas: establece una estructura y tamaño del equipo de un emparejamiento, e indica al emparejador cómo elegir a los jugadores para formar el mejor emparejamiento posible.

Sin embargo, el conjunto de reglas de emparejamiento puede hacer más. Por ejemplo, puede hacer lo siguiente:

- Optimizar el algoritmo de emparejamiento para el juego.
- Establecer requisitos mínimos de latencia de los jugadores para proteger la calidad del juego.
- Con el tiempo, flexibilizar los requisitos del equipo y las reglas de los emparejamientos para que todos los jugadores activos puedan encontrar un emparejamiento aceptable cuando lo deseen.
- Definir la gestión de las solicitudes de emparejamiento grupal mediante la agregación de grupos.
- Procesar emparejamientos de gran tamaño de 40 o más jugadores. Para obtener más información sobre la compilación de emparejamientos de gran tamaño, consulte Diseña un FlexMatch conjunto de reglas para partidos grandes.

Al compilar un conjunto de reglas de emparejamiento, tenga en cuenta las siguientes tareas opcionales y obligatorias:

- Descripción del conjunto de reglas (obligatorio)
- Personalización del algoritmo de coincidencia

- Declaración de atributos de jugador
- Definición de los equipos de emparejamiento
- Establecimiento de reglas para el emparejamiento de jugador
- Permiso para que los requisitos se flexibilicen con el tiempo

Puede crear su conjunto de reglas mediante la Amazon GameLift Servers consola o la `CreateMatchmakingRuleSet` operación.

Descripción del conjunto de reglas (obligatorio)

Proporcione detalles para el conjunto de reglas.

- nombre (opcional): una etiqueta descriptiva para su uso personal. Este valor no está asociado al nombre del conjunto de reglas que especificó al crear el conjunto de reglas con élAmazon GameLift Servers.
- ruleLanguageVersion— La versión del lenguaje de expresiones de propiedades que se utiliza para crear FlexMatch reglas. El valor debe ser `1.0`.

Personalización del algoritmo de coincidencia

FlexMatchoptimiza el algoritmo predeterminado de la mayoría de los juegos para que los jugadores participen en partidas aceptables con un tiempo de espera mínimo. Puede personalizar el algoritmo y ajustar el emparejamiento para el juego.

El siguiente es el algoritmo de FlexMatch emparejamiento predeterminado:

1. FlexMatchcoloca todas las entradas de matchmaking abiertas y las entradas rellenas en una bolsa de entradas.
2. FlexMatchagrupa aleatoriamente las entradas de la piscina en uno o más lotes. A medida que aumenta el número de boletos, FlexMatch forma lotes adicionales para mantener un tamaño de lote óptimo.
3. FlexMatchclasifica las entradas por antigüedad, dentro de cada lote.
4. FlexMatchcrea una combinación basada en el billete más antiguo de cada lote.

Para personalizar el algoritmo de emparejamiento, añada un componente de `algorithm` a su esquema de conjunto de reglas. Para obtener información de referencia completa, consulte FlexMatch esquema de conjunto de reglas.

Utilice las siguientes personalizaciones opcionales para influir en las diferentes etapas del proceso de emparejamiento.

- Adición de la clasificación previa a los lotes
- Formación de lotes en función de los atributos de batchDistance
- Priorización los tickets de reposición
- Preferencia de los tickets más antiguos con expansiones

Adición de la clasificación previa a los lotes

Puede ordenar el grupo de tickets antes de formar los lotes. Este tipo de personalización es más eficaz en juegos con grandes grupos de tickets. La clasificación previa por lotes puede ayudar a acelerar el proceso de emparejamiento y a aumentar la uniformidad de los jugadores en cuanto a las características definidas.

Defina los métodos de clasificación previa a los lotes mediante la propiedad del algoritmo `batchingPreference`. El ajuste predeterminado es `random`.

Entre las opciones para personalizar la clasificación previa a los lotes se incluyen las siguientes:

- Ordenar por atributos de jugador. Proporcione una lista de los atributos de los jugadores para ordenar previamente el grupo de tickets.

  Para ordenar por atributos de jugador, establezca `batchingPreference` en `sorted` y defina su lista de atributos de jugador en `sortByAttributes`. Para usar un atributo, primero declare el atributo en el componente `playerAttributes` del conjunto de reglas.

  En el siguiente ejemplo, FlexMatch ordena la reserva de entradas según el mapa de juego preferido de los jugadores y, a continuación, según la habilidad del jugador. Es más probable que los lotes resultantes contengan jugadores con habilidades similares que quieran utilizar el mismo mapa.

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

```
    "strategy": "exhaustiveSearch"
},
```

- Ordenar por latencia. Cree coincidencias con la latencia más baja disponible o cree rápidamente emparejamientos con una latencia aceptable. Esta personalización resulta útil para conjuntos de reglas que forman emparejamientos grandes de más de 40 jugadores.

  Establezca la propiedad del algoritmo `strategy` en`balanced`. La estrategia equilibrada limita los tipos de declaraciones de reglas disponibles. Para obtener más información, consulte Diseña un FlexMatch conjunto de reglas para partidos grandes.

  FlexMatchordena las entradas en función de los datos de latencia notificados por los jugadores de una de las siguientes maneras:

  - Ubicaciones de latencia más baja. La bolsa de entradas se clasifica previamente según las ubicaciones en las que los jugadores indican sus valores de latencia más bajos. FlexMatchLuego, agrupa los boletos con baja latencia en las mismas ubicaciones, lo que crea una mejor experiencia de juego. También reduce la cantidad de tickets en cada lote, por lo que el emparejamiento puede llevar más tiempo. Para utilizar esta personalización, establezca `batchingPreference` en `fastestRegion`, como se muestra en el siguiente ejemplo.

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

  - La latencia aceptable realiza el emparejamiento rápidamente. El grupo de tickets se clasifica previamente mediante ubicaciones en las que los jugadores informan de un valor de latencia aceptable. Esto forma menos lotes con más tickets. Con más tickets en cada lote, es más rápido encontrar emparejamientos aceptables. Para utilizar esta personalización, establezca la propiedad `batchingPreference` en `largestPopulation`, como se muestra en el siguiente ejemplo.

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

> **Note**
>
> El valor predeterminado para la estrategia equilibrada es `largestPopulation`.

Priorización los tickets de reposición

Si tu juego implementa el relleno automático o el relleno manual, puedes personalizar la forma en que se procesan FlexMatch los tickets de emparejamiento según el tipo de solicitud. El tipo de solicitud puede ser un nuevo emparejamiento o una solicitud de relleno. De forma predeterminada, FlexMatch trata ambos tipos de solicitudes de la misma manera.

La priorización del relleno afecta a la forma en que se gestionan FlexMatch los tickets después de agruparlos por lotes. La priorización de reposición requiere conjuntos de reglas para utilizar la estrategia de búsqueda exhaustiva.

FlexMatchno hace coincidir varios tickets de repostaje.

Para cambiar la prioridad de los tickets de reposición, configure la propiedad `backfillPriority`.

- Emparejar primero los tickets de reposición. Con esta opción se intenta igualar los tickets de reposición antes de crear nuevos emparejamientos. Esto significa que los jugadores entrantes tienen más probabilidades de unirse a un juego existente.

  Es mejor usar esto si el juego utiliza la reposición automática. La reposición automática se suele utilizar en juegos con sesiones de juego cortas y una alta rotación de jugadores. El relleno automático ayuda a estos juegos a formar un mínimo de partidas viables y a que comiencen a jugar mientras buscan FlexMatch más jugadores para llenar los espacios disponibles.

  Configure el `backfillPriority` en `high`.

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

- Emparejar los tickets de reposición al final. Con esta opción se ignoran los tickets de reposición hasta que evalúa todos los demás tickets. Esto significa que los FlexMatch jugadores entrantes se reintegran en juegos existentes cuando no se pueden unir a juegos nuevos.

Esta opción resulta útil cuando quiere utilizar la reposición como última opción para atraer jugadores a un juego, por ejemplo, cuando no hay suficientes jugadores para formar un nuevo juego.

Establece `backfillPriority` en `low`.

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

Preferencia de los tickets más antiguos con expansiones

Las reglas de expansión flexibilizan los criterios de las partidas cuando las partidas son difíciles de completar. Amazon GameLift Serversaplica las reglas de expansión cuando las entradas de una partida parcialmente completada alcanzan una edad determinada. Las marcas de tiempo de creación de las entradas determinan cuándo Amazon GameLift Servers se aplican las reglas; de forma predeterminada, FlexMatch registra la fecha y hora de la última entrada igualada.

Para cambiar cuándo FlexMatch se aplican las reglas de expansión, defina la propiedad de la siguiente manera: `expansionAgeSelection`

- Ampliar en función de los tickets más recientes. Esta opción permite aplicar las reglas de expansión en función del ticket más reciente que se añada al posible emparejamiento. Cada vez que FlexMatch coincide con un billete nuevo, el reloj se restablece. Con esta opción, los emparejamientos resultantes suelen ser de mayor calidad, pero tardan más en igualarse; las solicitudes de emparejamiento pueden agotarse antes de completarse si tardan demasiado en emparejarse. Establezca `expansionAgeSelection` en `newest`. `newest` es el valor predeterminado.
- Ampliar en función de los tickets más antiguos. Esta opción permite aplicar las reglas de expansión en función del ticket más antiguo en el posible emparejamiento. Con esta opción, las expansiones FlexMatch se aplican más rápido, lo que mejora los tiempos de espera para los jugadores que se emparejan primero, pero reduce la calidad de la partida para todos los jugadores. Establece `expansionAgeSelection` en `oldest`.

```
"algorithm": {
    "expansionAgeSelection": "oldest",
```

```
    "strategy": "exhaustiveSearch"
},
```

Declaración de atributos de jugador

En esta sección aparecen los atributos individuales de los jugadores para incluirlos en las solicitudes de emparejamiento. Hay dos motivos por los que puede declarar los atributos de los jugadores en un conjunto de reglas:

- Cuando el conjunto de reglas contiene reglas que se basan en los atributos del jugador.
- Cuando quiere pasar un atributo de jugador a la sesión de juego a través de la solicitud de emparejamiento. Por ejemplo, es posible que desees transferir las opciones de personaje del jugador a la sesión de juego antes de que cada jugador se conecte.

Al declarar el atributo del jugador, incluya la siguiente información:

- nombre (obligatorio): este valor debe ser único en el conjunto de reglas.
- tipo (obligatorio): es el tipo de datos del valor de atributo. Los tipos de datos válidos son números, cadenas, listas de cadenas o mapa de cadenas.
- predeterminado (opcional): introduce un valor predeterminado para usarlo si una solicitud de emparejamiento no proporciona un valor de atributo. Si no se declara ningún valor predeterminado y la solicitud no incluye un valor, no FlexMatch se puede cumplir con la solicitud.

Definición de los equipos de emparejamiento

Describa la estructura y el tamaño de los equipos para un emparejamiento. Cada emparejamiento debe tener al menos un equipo, y puede definir tantos equipos como quiera. Los equipos pueden tener el mismo número de jugadores o ser asimétricos. Por ejemplo, puede definir un equipo monstruo de un jugador y un equipo de cazadores con 10 jugadores.

FlexMatch procesa las solicitudes de emparejamiento como emparejamientos reducidos o de gran tamaño en función de la forma en la que el conjunto de reglas define los tamaños de los equipos. Los emparejamientos potenciales de hasta 40 jugadores son emparejamientos reducidos, mientras que los emparejamientos con más de 40 jugadores son emparejamientos de gran tamaño. Para determinar el tamaño de emparejamiento potencial de un conjunto de reglas, sume la configuración de maxPlayer para todos los equipos definidos en el conjunto de reglas.

- nombre (obligatorio): permite asignar un nombre único a cada equipo. Este nombre se utiliza en las reglas y las expansiones, así como en FlexMatch las referencias a los datos de emparejamiento de una sesión de juego.
- maxPlayers (obligatorio): permite especificar el número máximo de jugadores que desea asignar al equipo.
- minPlayers (obligatorio): permite especificar el número máximo de jugadores que desea asignar al equipo.
- cantidad (opcional): permite especificar el número de equipos que quiere formar con esta definición. Cuando FlexMatch crea una partida, les da a estos equipos el nombre proporcionado con un número adjunto. Por ejemplo `Red-Team1`, `Red-Team2` y `Red-Team3`.

FlexMatchintenta llenar los equipos hasta el tamaño máximo de jugadores, pero crea equipos con menos jugadores. Si desea que todos los equipos del emparejamiento tengan el mismo tamaño, puede crear una regla para eso. Consulte el tema FlexMatchejemplos de conjuntos de reglas para ver un ejemplo de una regla de `EqualTeamSizes` .

Establecimiento de reglas para el emparejamiento de jugador

Cree un conjunto de declaraciones de reglas que evalúe a los jugadores para su aceptación en un emparejamiento. Las reglas pueden establecer requisitos aplicables a jugadores individuales, equipos o a todo el emparejamiento. Cuando Amazon GameLift Servers procesa una solicitud de emparejamiento, comienza con el jugador más antiguo del grupo de jugadores disponibles y crea un emparejamiento alrededor de ese jugador. Para obtener ayuda detallada sobre la creación de FlexMatch reglas, consulteFlexMatchtipos de reglas.

- nombre (obligatorio): es un nombre fácil de recordar, que identifica de forma exclusiva la regla dentro de un conjunto de reglas. También se hace referencia a los nombres de la reglas en registros de eventos y métricas que realizan un seguimiento de la actividad relacionada con esta regla.
- descripción (opcional): utilice este elemento para asociar una descripción de texto de formato libre.
- tipo (obligatorio): el elemento «tipo» identifica la operación que se debe utilizar al procesar la regla. Cada tipo de regla requiere un conjunto de propiedades adicionales. Consulte una lista de tipos de reglas y propiedades válidos en FlexMatch lenguaje de reglas.
- Propiedad de tipo de regla (puede ser obligatoria): según el tipo de regla definido, es posible que deba configurar determinadas propiedades de la regla. Para obtener más información sobre las

propiedades y cómo usar el lenguaje de expresión de propiedades FlexMatch, consulte FlexMatch lenguaje de reglas.

Permiso para que los requisitos se flexibilicen con el tiempo

Las expansiones te permiten relajar los criterios de las reglas a lo largo del tiempo cuando no FlexMatch puedes encontrar una que coincida. Esta función FlexMatch garantiza que una apuesta esté disponible cuando no sea una combinación perfecta. Al flexibilizar las reglas con una expansión, está ampliando gradualmente el grupo de jugadores que se puede emparejar.

Las expansiones comienzan cuando la antigüedad del ticket más reciente del emparejamiento incompleto coincide con el tiempo de espera de una expansión. Al FlexMatch añadir una nueva entrada al partido, es posible que se restablezca el tiempo de espera de la expansión. Puede personalizar el inicio de las expansiones en la sección `algorithm` del conjunto de reglas.

Este es un ejemplo de una expansión que aumenta gradualmente el nivel mínimo de habilidad necesario para el emparejamiento. El conjunto de reglas utiliza una regla de distancia, llamada así SkillDeltapara exigir que todos los jugadores de una partida tengan una diferencia de 5 niveles de habilidad entre sí. Si no se realizan nuevos emparejamientos durante quince segundos, esta expansión busca una diferencia de nivel de habilidad de 10 y, diez segundos después, busca una diferencia de 20.

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

Si un emparejador tiene habilitada la reposición automática, no flexibilice los requisitos de recuento de jugadores demasiado rápido. La nueva sesión del juego tarda unos segundos en arrancar y en iniciar la reposición automática. Es mejor iniciar la expansión después de que la reposición automática tienda a activarse para sus juegos. El tiempo de expansión varía en función de la composición del equipo, así que realice las pruebas para encontrar la mejor estrategia de expansión para el juego.

Diseña un FlexMatch conjunto de reglas para partidos grandes

Si el conjunto de reglas crea emparejamientos que permiten de 41 a 200 jugadores, tendrá que realizar algunos ajustes en la configuración del conjunto de reglas. Esos ajustes optimizan el algoritmo de emparejamiento para que pueda compilar emparejamientos grandes y viables y, al mismo tiempo, reducir los tiempos de espera de los jugadores. Como resultado, los conjuntos de reglas de partidos de gran tamaño sustituyen a las laboriosas reglas personalizadas por soluciones estándar optimizadas para las prioridades habituales en materia de emparejamiento.

A continuación, le explicamos cómo determinar si necesita optimizar el conjunto de reglas para emparejamientos numerosos:

1. Para cada equipo definido en el conjunto de reglas, obtenga el valor de MaxPlayer,
2. Sume todos los valores de MaxPlayer. Si el total es superior a 40, tiene conjunto de reglas de emparejamientos de gran tamaño.

Para optimizar el conjunto de reglas para emparejamientos grandes, realice los ajustes que se describen a continuación. Consulte el esquema para ver un conjunto de reglas de coincidencia de gran tamaño establecido en Esquema de conjuntos de reglas para emparejamientos de gran tamaño y ejemplos de conjuntos de reglas en Ejemplo: crea una partida grande.

Personalización del algoritmo de emparejamiento para emparejamientos grandes

Añada un componente de algoritmo al conjunto de reglas, si aún no existe ninguno. Establezca las siguientes propiedades.

- `strategy` (obligatorio): establezca la propiedad `strategy` en «equilibrado». Esta configuración se activa FlexMatch para realizar comprobaciones adicionales después del partido para encontrar el equilibrio óptimo del equipo en función de un atributo de jugador específico, que se define en la `balancedAttribute` propiedad. La estrategia equilibrada reemplaza la necesidad de reglas personalizadas para crear equipos emparejados.
- `balancedAttribute` (obligatorio): permite identificar un atributo del jugador para usarlo al equilibrar los equipos en un emparejamiento. Este atributo debe tener un tipo de datos numéricos (doble o entero). Por ejemplo, si eliges equilibrar la habilidad del jugador, FlexMatch intenta asignar jugadores de forma que todos los equipos tengan niveles de habilidad agregados lo más igualados posible. Asegúrese de declarar el atributo de equilibrio en los atributos de jugador del conjunto de reglas.

- `batchingPreference` (opcional): permite elegir cuánto énfasis quiere poner en formar emparejamientos con la menor latencia posible para los jugadores. Esta configuración afecta a la forma en que se ordenan los tickets para los emparejamientos antes de organizar los emparejamientos. Las opciones son:
  - Mayor población. FlexMatch permite utilizar todas las entradas del pool que tengan valores de latencia aceptables en al menos una ubicación en común. Como resultado, el grupo de tickets potencial tiende a ser grande, lo que facilita completar los emparejamientos con mayor rapidez. Es posible que los jugadores participen en juegos con una latencia aceptable, pero no siempre óptima. Si la propiedad `batchingPreference` no está establecida, este es el comportamiento predeterminado cuando `strategy` se establece en «equilibrado».
  - Ubicación más rápida. FlexMatch clasifica previamente todas las entradas del grupo en función de dónde presenten los valores de latencia más bajos. Como resultado, los emparejamientos suelen estar formadas por jugadores que muestran una baja latencia en las mismas ubicaciones. Al mismo tiempo, la cantidad potencial de tickets para cada emparejamiento es menor, lo que puede aumentar el tiempo necesario para completar el emparejamiento. Además, puesto que se da mayor prioridad a la latencia, los jugadores de los emparejamientos pueden variar más en función del atributo de equilibrio.

En el siguiente ejemplo, se configura el algoritmo de emparejamientos para que se comporte de la siguiente manera: (1) Clasifique previamente los grupos de tickets para agrupar los tickets por ubicación donde tengan valores de latencia aceptables; (2) Forme lotes de tickets clasificados para igualarlos; (3) Cree emparejamientos con los tickets de un lote y equilibre los equipos para igualar la habilidad media de los jugadores.

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

Declaración de atributos de jugador

Asegúrese de declarar el atributo de jugador que se utiliza como atributo de equilibrio en el algoritmo del conjunto de reglas. Este atributo debe incluirse para cada jugador en una solicitud de emparejamiento. Puede proporcionar un valor predeterminado para el atributo del jugador, pero el equilibrio de atributos funciona mejor cuando se proporcionan valores específicos del jugador.

Definición de equipos

El proceso de definir el tamaño y la estructura del equipo es el mismo que en los partidos pequeños, pero la forma FlexMatch llena los equipos es diferente. Esto afecta al aspecto que puedan tener los emparejamientos cuando se completen solo parcialmente. Por eso, es posible que desee ajustar los tamaños mínimos del equipo.

FlexMatch utiliza las siguientes reglas al asignar un jugador a un equipo. Primero: busque equipos que aún no hayan alcanzado su requisito mínimo de jugador. Segundo: de esos equipos, encuentre el que tenga más ranuras abiertas.

Para los emparejamientos que definen Varios equipos de igual tamaño, los jugadores se añaden secuencialmente a cada equipo hasta que estén completos. Como resultado, los equipos de un emparejamiento siempre tienen casi el mismo número de jugadores, incluso cuando el emparejamiento no está completo. Actualmente no hay forma de forzar equipos de igual tamaño en emparejamientos de gran tamaño. Para emparejamientos con equipos de tamaño asimétrico, el proceso es un poco más complejo. En este caso, los jugadores se asignan inicialmente a los equipos más grandes con el mayor número de ranuras abiertas. A medida que el número de ranuras abiertas se distribuye de manera más uniforme en todos los equipos, los jugadores se agrupan en una ranura en equipos más pequeños.

Por ejemplo, supongamos que tiene un conjunto de reglas con tres equipos. Los equipos rojo y azul están configurados en `maxPlayers`=10, `minPlayers`=5. El equipo verde tiene establecido `maxPlayers`=3, `minPlayers`=2. Esta es la secuencia de llenado:

1. Ningún equipo ha llegado a `minPlayers`. Los equipos Rojo y Azul tienen 10 ranuras abiertas, mientras que el Verde tiene 3. Se asignan los primeros 10 jugadores (5 en cada uno) a los equipos Rojo y Azul. Ambos equipos han llegado a `minPlayers`.
2. El equipo verde aún no ha llegado a `minPlayers`. Se asignan los siguientes 2 jugadores al equipo Verde. El equipo verde ya ha llegado a `minPlayers`.
3. Con todos los equipos en `minPlayers`, ahora se asignan jugadores adicionales en función del número de ranuras disponibles. Los equipos rojo y azul tienen 5 ranuras abiertas, mientras que el verde tiene 1. Se asignan los primeros 8 jugadores (4 en cada uno) a los equipos rojo y azul. Ahora todos los equipos tienen una ranura libre.
4. Las tres ranuras para jugadores restantes se asignan (una para cada uno) a los equipos sin ningún orden en particular.

Establecimiento de reglas para emparejamientos grandes

El emparejamiento para emparejamientos grandes se basa principalmente en la estrategia de equilibrio y en las optimizaciones de la latencia por lotes. la mayoría de las reglas personalizadas no están disponibles. Sin embargo, puede incorporar los siguientes tipos de reglas:

- Regla que establece un límite estricto en la latencia de los jugadores. Utilice el tipo de regla `latency` con la propiedad `maxLatency`. Consulte la referencia Latency rule. A continuación se muestra un ejemplo que establece la latencia máxima del jugador en 200 milisegundos:

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

- Regla para agrupar a los jugadores en función de la cercanía de un atributo de jugador específico. Esto no es lo mismo que definir un atributo de equilibrio como parte del algoritmo de emparejamientos grandes, que se centra en formar equipos igualados. Esta regla clasifica los tickets de los emparejamientos en función de la similitud entre los valores de los atributos especificados, como la habilidad de principiante o experto, lo que suele llevar a que los jugadores de los emparejamientos estén muy alineados con respecto al atributo especificado. Utilice el tipo de regla `batchDistance`, identifique un atributo numérico y especifique el rango más amplio que desee permitir. Consulte la referencia Regla de distancia por lotes. Este es un ejemplo en el que se exige que los jugadores de un emparejamiento estén separados por un nivel de habilidad entre sí:

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

Reducción de los requisitos de emparejamientos de gran tamaño

Al igual que con los pequeños emparejamientos, puede usar expansiones para reducir los requisitos de los emparejamientos con el paso del tiempo cuando no sea posible conseguir emparejamientos válidos. Con los emparejamientos de gran tamaño, tiene la opción de reducir las reglas de latencia o el recuento de jugadores del equipo.

Si utiliza la reposición automática de emparejamientos para emparejamientos de gran tamaño, evite flexibilizar el recuento de jugadores del equipo demasiado rápido. FlexMatch comienza a generar solicitudes de relleno solo después de que comience una sesión de juego, lo que puede no ocurrir hasta varios segundos después de que se haya creado una partida. Durante ese tiempo, FlexMatch puede crear varias sesiones de juego nuevas parcialmente rellenadas, especialmente cuando se reducen las reglas del recuento de jugadores. Como resultado, dispondrá de más sesiones de juego de las que necesita y los jugadores se extenderán demasiado entre ellas. La práctica recomendada es proporcionar al primer paso de la expansión del recuento de jugadores un tiempo de espera más largo, lo suficiente para que su sesión de juego comience. Puesto que las solicitudes de reposición tienen mayor prioridad con los emparejamientos de gran tamaño, los jugadores entrantes se incluirán en los juegos existentes antes de comenzar un nuevo juego. Es posible que tenga que hacer varias pruebas hasta encontrar el tiempo de espera ideal para su juego.

Aquí tiene un ejemplo que reduce gradualmente el recuento de jugadores del equipo Amarillo con un tiempo de espera inicial más largo. Tenga en cuenta que los tiempos de espera en las expansiones de conjuntos de reglas son absolutos, no compuestos. Por lo tanto, la primera expansión se produce los cinco segundos, y la segunda expansión cinco segundos después, a los diez segundos.

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

Tutorial: Crear un conjunto de reglas de emparejamiento

Antes de crear un conjunto de reglas de emparejamiento para tu Amazon GameLift Servers FlexMatch emparejador, te recomendamos comprobar la sintaxis del conjunto de reglas. Después de crear un conjunto de reglas con la Amazon GameLift Servers consola o el AWS Command Line Interface (AWS CLI), no podrás cambiarlo.

Ten en cuenta que hay una cuota de servicio para el número máximo de conjuntos de reglas que puedes tener en una AWS región, por lo que es recomendable eliminar los conjuntos de reglas que no utilices.

Console

Creación de un conjunto de reglas

1. Abra la consola de Amazon GameLift Servers en https://console.aws.amazon.com/gamelift/.
2. Cambie a la AWS región en la que desee crear su conjunto de reglas. Defina conjuntos de reglas en la misma región que la configuración de emparejamiento que los utiliza.
3. En el panel de navegación FlexMatch, selecciona Conjuntos de reglas de emparejamiento.
4. En la página Conjuntos de reglas de emparejamiento, elija Crear conjunto de reglas.
5. En la página Crear un conjunto de reglas de emparejamiento, realice el siguiente procedimiento:

   a. En Configuración del conjunto de reglas, en Nombre, especifique un nombre descriptivo único que pueda utilizar para identificarlo en una lista o en las tablas de eventos y métricas.

   b. En Conjunto de reglas, especifique el conjunto de reglas en JSON. Para obtener información sobre el diseño de un conjunto de reglas, consulte Diseñar un conjunto de FlexMatch reglas. También puede utilizar uno de los conjuntos de reglas de ejemplo de FlexMatchejemplos de conjuntos de reglas.

   c. Elija Validar para verificar que la sintaxis del conjunto de reglas sea correcta. No puede editar los conjuntos de reglas una vez creados, por lo que es una buena idea validarlos primero.

   d. (Opcional) En Etiquetas, añade etiquetas para ayudarte a gestionar y realizar un seguimiento de tus AWS recursos.
6. Seleccione Crear. Si la creación se realiza correctamente, podrá utilizar el conjunto de reglas con un emparejador.

AWS CLI

Creación de un conjunto de reglas

Abre una ventana de línea de comandos y usa el comando create-matchmaking-rule-set.

Este comando de ejemplo crea un sencillo conjunto de reglas de emparejamiento para configurar un equipo. Asegúrese de crear el conjunto de reglas en la misma AWS región que las configuraciones de emparejamiento que lo utilizan.

```
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}]}'
```

Si la solicitud de creación se ha realizado correctamente, Amazon GameLift Servers devuelve un MatchmakingRuleSetobjeto que incluye la configuración que especificó. Un emparejador puede utilizar ahora el conjunto de reglas nuevo.

Console

Eliminación de un conjunto de reglas

1. Abra la consola de Amazon GameLift Servers en https://console.aws.amazon.com/gamelift/.
2. Cambie a la región en la que creó el conjunto de reglas.
3. En el panel de navegación FlexMatch, elija Conjuntos de reglas de emparejamiento.
4. En la página Conjuntos de reglas de emparejamiento, seleccione el conjunto de reglas que desee eliminar y, a continuación, elija Eliminar.
5. En el cuadro de diálogo Eliminar conjunto de reglas, elija Eliminar para confirmar la eliminación.

> **Note**
>
> Si una configuración de emparejamiento utiliza el conjunto de reglas, Amazon GameLift Servers muestra un mensaje de error (No se puede eliminar el conjunto de reglas). Si esto ocurre, cambie la configuración de emparejamiento para utilizar un conjunto de reglas diferente e inténtelo de nuevo. Para saber qué configuraciones de emparejamiento utilizan actualmente un conjunto de reglas, elija el nombre de un conjunto de reglas para ver la página de detalles correspondiente.

AWS CLI

Eliminación de un conjunto de reglas

Abre una ventana de línea de comandos y usa el comando delete-matchmaking-rule-setpara eliminar un conjunto de reglas de emparejamiento.

Si una configuración de emparejamiento utiliza el conjunto de reglas, Amazon GameLift Servers devuelve un mensaje de error. Si esto ocurre, cambie la configuración de emparejamiento para utilizar un conjunto de reglas diferente e inténtelo de nuevo. Para obtener una lista de las configuraciones de emparejamiento que utilizan un conjunto de reglas, usa el comando describe-matchmaking-configurationsy especifica el nombre del conjunto de reglas.

En este comando de ejemplo, primero se comprueba el uso del conjunto de reglas de emparejamiento y, a continuación, se elimina el conjunto de reglas.

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

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

FlexMatchejemplos de conjuntos de reglas

Los conjuntos de reglas de FlexMatch pueden cubrir una amplia variedad de situaciones de emparejamiento. Los siguientes ejemplos se ajustan a la estructura de configuración y el lenguaje de expresiones de propiedad de FlexMatch. Copie estos conjuntos de reglas en su totalidad o elija los componentes según sea necesario.

Para obtener más información sobre cómo utilizar las reglas y los conjuntos de reglas de FlexMatch, consulte los siguientes temas:

> **Note**
>
> Cuando evalúe un ticket de emparejamiento que incluya varios jugadores, todos los jugadores de la solicitud deben cumplir los requisitos del emparejamiento.

Temas

Ejemplo: crea dos equipos con jugadores igualados

Este ejemplo ilustra cómo configurar dos equipos de jugadores emparejados de manera uniforme con las siguientes instrucciones.

- Cree dos equipos de jugadores.
  - Incluya entre cuatro y ocho jugadores en cada equipo.
  - Los equipos definitivos deben tener el mismo número de jugadores.
- Incluya el nivel de habilidad de un jugador (si no se proporciona, de manera predeterminada es 10).
- Elija jugadores en función de si su nivel de habilidad es similar al de otros jugadores. Asegúrese de que la habilidad promedio de los jugadores de ambos equipos esté dentro de 10 puntos entre sí.
- Si el emparejamiento no se completa rápidamente, relaje el requisito de habilidades de los jugadores para completar un emparejamiento dentro de un tiempo razonable.
  - Transcurridos 5 segundos, expanda la búsqueda para permitir equipos con habilidades promedio de los jugadores dentro de los 50 puntos.
  - Transcurridos 15 segundos, expanda la búsqueda para permitir equipos con habilidades promedio de los jugadores dentro de los 100 puntos.

Notas sobre el uso de este conjunto de reglas:

- Este ejemplo permite que los equipos tengan cualquier tamaño entre cuatro y ocho jugadores (aunque deben ser del mismo tamaño). En el caso de los equipos con un rango de tamaños válido, el creador de emparejamientos hace todo lo posible para emparejar el número máximo de jugadores permitidos.

- La regla `FairTeamSkill` garantiza que los equipos estén emparejados de manera uniforme según las habilidades de los jugadores. Para evaluar esta regla con cada posible jugador nuevo, FlexMatch prueba a añadir el jugador a un equipo y calcula las medias. Si la regla genera un error, no se agrega el posible jugador al emparejamiento.
- Puesto que ambos equipos tienen estructuras idénticas, puede optar por crear una sola definición de equipo y establecer la cantidad de equipos en "2". En este caso, si le ha asignado el nombre "extranjeros" al equipo, sus equipos se denominarán "extranjeros_1" y "extranjeros_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
        }]
    }]
}
```

Ejemplo: crea equipos desiguales (Cazadores contra monstruos)

Este ejemplo describe un modo de juego en el que un grupo de jugadores da caza a un único monstruo. Las personas eligen el rol de cazador o de monstruo. Los cazadores especifican el nivel de habilidad mínimo del monstruo al que quieren enfrentarse. El tamaño mínimo del equipo de cazadores puede relajarse a lo largo del tiempo para completar el emparejamiento. Esta situación establece las siguientes instrucciones:

- Cree un equipo con cinco cazadores.
- Cree otro equipo con un monstruo.
- Incluya los siguientes atributos de los jugadores:
  - El nivel de habilidad de un jugador (si no se proporciona, de manera predeterminada es 10).
  - El nivel de habilidad de un monstruo preferido del jugador (si no se proporciona, de manera predeterminada es 10).
  - Si el jugador quiere ser el monstruo (si no se proporciona, de manera predeterminada es 0 o false).
- Elija un jugador para que sea el monstruo en función de los siguientes criterios:
  - El jugador debe solicitar el rol del monstruo.
  - El jugador debe cumplir o superar el nivel de habilidad más alto preferido por los jugadores que ya se han agregado al equipo de cazadores.
- Elija jugadores para el equipo de cazadores en función de los siguientes criterios:

  - Los jugadores que soliciten el rol del monstruo no pueden unirse al equipo de cazadores.
  - Si el rol del monstruo ya se ha asignado, el jugador debe desear un nivel de habilidad de monstruo inferior a la habilidad del monstruo propuesto.
- Si un emparejamiento no se completa rápidamente, relaje el tamaño mínimo del equipo de cazadores de la siguiente manera:
  - Después de 30 segundos, permita que el juego comience con tan solo cuatro jugadores en el equipo de cazadores.
  - Después de 60 segundos, permita que el juego comience con tan solo tres personas en el equipo de cazadores.

Notas sobre el uso de este conjunto de reglas:

- Al utilizar dos equipos independientes para cazadores y monstruo, puede evaluar la pertenencia según diferentes conjuntos de criterios.

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

Ejemplo: establece los requisitos y los límites de latencia a nivel de equipo

En este ejemplo se muestra cómo configurar equipos de jugadores y aplicar un conjunto de reglas a cada equipo, en lugar de a cada jugador. En él se utiliza una definición única para crear tres equipos igualmente emparejados. También establece una latencia máxima para todos los jugadores. Los

máximos de latencia pueden relajarse con el tiempo para completar el emparejamiento. En este ejemplo se establecen las siguientes instrucciones:

- Cree tres equipos de jugadores.
  - Incluya entre tres y cinco jugadores en cada equipo.
  - Los equipos definitivos deben contener el mismo o casi el mismo número de jugadores (dentro de uno).
- Incluya los siguientes atributos de los jugadores:
  - El nivel de habilidad de un jugador (si no se proporciona, de manera predeterminada es 10).
  - Un rol de personaje de jugador (si no se proporciona, de manera predeterminada es "ignorante").
- Elija jugadores en función de si su nivel de habilidad es similar al de otros jugadores del emparejamiento.
  - Asegúrese de que la habilidad promedio de los jugadores de cada equipo esté dentro de 10 puntos entre sí.
- Limite los equipos al siguiente número de personajes de "médicos":
  - Un emparejamiento completo puede tener un máximo de cinco médicos.
- Solo empareje los jugadores que cuenten con una latencia de 50 milisegundos o menos.
- Si un emparejamiento no se completa rápidamente, relaje el requisito de latencia de los jugadores de la siguiente manera:
  - Transcurridos 10 segundos, permita valores de latencia de los jugadores de hasta 100 ms.
  - Transcurridos 20 segundos, permita valores de latencia de los jugadores de hasta 150 ms.

Notas sobre el uso de este conjunto de reglas:

- El conjunto de reglas garantiza que los equipos estén emparejados de manera uniforme según las habilidades de los jugadores. Para evaluar la regla `FairTeamSkill`, FlexMatch añade provisionalmente el posible jugador a un equipo y calcula la habilidad media de los jugadores del equipo. A continuación, la compara con las habilidades medias de los jugadores de ambos equipos. Si la regla genera un error, no se agrega el posible jugador al emparejamiento.
- Los requisitos del equipo y del nivel de emparejamiento (número total de médicos) se logran mediante una regla de colección. Este tipo de regla toma una lista de atributos de personajes para todos los jugadores y la compara con los recuentos máximos. Utilice `flatten` para crear una lista de todos los jugadores en todos los equipos.

- Cuando realice la evaluación en función de la latencia, tenga en cuenta lo siguiente:
  - Se proporcionan datos de latencia en la solicitud de emparejamiento como parte del objeto Player. No se trata de un atributo del jugador, por lo que no necesita enumerarse como tal. Para obtener mediciones de latencia precisas, usa las balizas Amazon GameLift Servers de ping UDP. Estos puntos de conexión permiten medir la latencia real de la red UDP entre los dispositivos reproductores y cada una de las posibles ubicaciones de alojamiento, lo que permite tomar decisiones de ubicación más precisas que con los pings ICMP. Para obtener más información sobre el uso de balizas de ping UDP para medir la latencia, consulta la sección Balizas de ping UDP.
  - El creador de emparejamientos evalúa la latencia por región. Cualquier región con una latencia superior al máximo se ignora. Para ser aceptado para un emparejamiento, un jugador debe tener al menos una región con una latencia por debajo del máximo.
  - Si en una solicitud de emparejamiento no se proporcionan los datos de latencia de uno o varios jugadores, la solicitud se rechaza para todos los emparejamientos.

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

Ejemplo: usa una clasificación explícita para encontrar las mejores coincidencias

Este ejemplo configura un emparejamiento sencillo con dos equipos de tres jugadores. Muestra cómo utilizar reglas de ordenación explícitas para encontrar los mejores emparejamientos posibles lo más rápido posible. Estas reglas ordenan todos los tickets de emparejamiento activos para crear los mejores emparejamientos según determinados requisitos clave. Este ejemplo se implementa con las siguientes instrucciones:

- Cree dos equipos de jugadores.
- Incluya exactamente tres jugadores en cada equipo.
- Incluya los siguientes atributos de los jugadores:
  - Nivel de experiencia (si no se proporciona, de manera predeterminada es 50).
  - Modos de juego preferidos (puede enumerar varios valores) (si no se proporcionan, de manera predeterminada son "coop" y "deathmatch").
  - Mapas de juego preferidos, se incluyen el nombre del mapa y la ponderación de preferencias (si no se proporciona, de manera predeterminada es `"defaultMap"` con un peso de 100).
- Configure la ordenación previa:
  - Ordene los jugadores en función de su preferencia por el mismo mapa de juego que el jugador de referencia. Los jugadores pueden tener varios mapas de juego favoritos, por lo que este ejemplo utiliza un valor de preferencia.
  - Ordene los jugadores según el grado de coincidencia de su nivel de experiencia con el del jugador de referencia. Con esta ordenación, todos los jugadores de todos los equipos tendrían niveles de experiencia lo más similares posible.
- Todos los jugadores de todos los equipos deben haber seleccionado al menos un modo de juego en común.
- Todos los jugadores de todos los equipos deben haber seleccionado al menos un mapa de juego en común.

Notas sobre el uso de este conjunto de reglas:

- La ordenación de mapa de juego utiliza una ordenación absoluta que compara el valor del atributo mapPreference. Puesto que es la primera del conjunto de reglas, se realiza esta ordenación en primer lugar.
- La ordenación por experiencia utiliza una ordenación por distancia para comparar el nivel de habilidades de un posible jugador en relación con las habilidades del jugador de referencia.

- Las ordenaciones se realizan según la disposición en la que aparecen en un conjunto de reglas. En esta situación, se ordena a los jugadores por preferencia de mapa de juego y, a continuación, por el nivel de experiencia.

```
{
    "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
    }]
}
```

Ejemplo: encuentra intersecciones entre los atributos de varios jugadores

Este ejemplo ilustra cómo utilizar una regla de colección para encontrar intersecciones en dos o más atributos de los jugadores. Cuando trabaje con colecciones, puede utilizar la operación `intersection` para un único atributo y la operación `reference_intersection_count` para varios atributos.

Para ilustrar este enfoque, este ejemplo evalúa los jugadores de un emparejamiento en función de sus personajes preferidos. El juego de ejemplo es un estilo «free-for-all» en el que todos los jugadores de una partida son oponentes. A cada jugador se le pide que (1) elija un personaje para sí mismo, y (2) elija los personajes contra los que desea jugar. Necesitamos una regla que garantice que cada jugador de un emparejamiento utiliza un personaje que está la lista de oponentes preferidos de todos los demás jugadores.

El conjunto de reglas del ejemplo describe un emparejamiento con las siguientes características:

- Estructura del equipo: un equipo de cinco jugadores
- Atributos de los jugadores:
  - myCharacter: el personaje elegido por el jugador.
  - preferredOpponents: lista de personajes contra los que desea jugar el jugador.
- Reglas de emparejamiento: un posible emparejamiento es aceptable si cada uno de los personajes en uso se encuentra en la lista de oponentes preferidos de cada jugador.

Para implementar la regla de emparejamiento, en este ejemplo se utiliza una regla de colección con los siguientes valores de propiedades:

- Operación: utiliza la operación para evaluar el grado de coincidencia de cada lista de cadenas del valor de medición con la lista de cadenas del valor de referencia.
- Medición: utiliza la expresión de propiedad `flatten` para crear una lista de listas de cadenas, cada una de las cuales contiene el valor del atributo myCharacter de un jugador.
- Valor de referencia: utiliza la expresión de propiedad `set_intersection` para crear una lista de cadenas de todos los valores del atributo preferredOpponents que son comunes a cada jugador del emparejamiento.
- Restricciones: se establece en 1 para garantizar que el personaje elegido por cada jugador (una lista de cadenas en la medición) coincide al menos con uno de los oponentes preferidos comunes a todos los jugadores (una cadena en el valor de referencia).
- Expansión: si no se consigue realizar un emparejamiento en 15 segundos, flexibilizar el requisito mínimo de intersección.

El flujo de proceso para esta regla es el siguiente:

1. Se añade un jugador al posible emparejamiento. El valor de referencia (una lista de cadenas) se vuelve a calcular para incluir las intersecciones con la lista de oponentes preferidos del nuevo

jugador. El valor de la medición (una lista de listas de cadenas) se vuelve a calcular para añadir el personaje elegido del jugador nuevo como una lista de cadenas nueva.

2. Amazon GameLift Servers verifica que cada lista de cadenas del valor de medición (los personajes elegidos por los jugadores) coincide al menos con una cadena del valor de referencia (los oponentes preferidos de los jugadores). Dado que en este ejemplo cada lista de cadenas de la medición contiene solo un valor, la intersección es 0 o 1.
3. Si cualquier lista de cadenas de la medición no tiene un valor que coincide con la lista de cadenas de valores de referencia, la regla no se cumple y el jugador nuevo se elimina del posible emparejamiento.
4. Si no se consigue un emparejamiento en un plazo de 15 segundos, se elimina el requisito de emparejamiento con el contrincante para rellenar el resto de ranuras de jugadores en el emparejamiento.

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

Ejemplo: compara los atributos de todos los jugadores

Este ejemplo ilustra cómo comparar los atributos de los jugadores entre un grupo de jugadores.

El conjunto de reglas del ejemplo describe un emparejamiento con las siguientes características:

- Estructura del equipo: dos equipos de un solo jugador
- Atributos de los jugadores:
  - gameMode: tipo de juego elegido por el jugador (si no se proporciona, el valor predeterminado es "turn-based" (por turnos)).
  - gameMap: mundo del juego elegido por el jugador (si no se proporciona, el valor predeterminado es 1).
  - character (personaje): personaje elegido por el jugador (si no hay ningún valor predeterminado, significa que los jugadores deben especificar un personaje).
- Reglas de emparejamiento: los jugadores deben emparejarse de acuerdo con los requisitos siguientes:
  - Los jugadores deben elegir el mismo modo de juego.
  - Los jugadores deben elegir el mismo mapa de juego.
  - Los jugadores deben elegir personajes distintos.

Notas sobre el uso de este conjunto de reglas:

- Para implementar la regla de emparejamiento, en este ejemplo se utilizan reglas de comparación para comprobar todos los valores de los atributos de los jugadores. Para el modo y el mapa de juego, la regla verifica que los valores sean los mismos. Para el personaje, la regla verifica que los valores sean distintos.

- En este ejemplo se utiliza una definición de jugador con una propiedad de cantidad para crear ambos equipos de jugadores. Al equipo se le asignan los siguientes nombres: "jugador_1" y "jugador_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])"]
    }]
}
```

Ejemplo: crea una partida grande

En este ejemplo se muestra cómo configurar un conjunto de reglas para emparejamientos que pueden superar los 40 jugadores. Cuando un conjunto de reglas describe equipos con un recuento de maxPlayer total superior a 40, se procesa como un emparejamiento de gran tamaño. Obtenga más información en Diseña un FlexMatch conjunto de reglas para partidos grandes.

El conjunto de reglas de ejemplo crea un emparejamiento mediante las siguientes instrucciones:

- Cree un equipo con un máximo de 200 jugadores con un requisito mínimo de 175 jugadores.
- Criterios de equilibrio: seleccione jugadores en función del nivel de habilidades similar. Todos los jugadores deben informar sobre su nivel de habilidades para realizar el emparejamiento.
- Preferencia por lotes: agrupe los jugadores por criterios de equilibrio similares cuando cree los emparejamientos.
- Reglas de latencia: configure el máximo aceptable de latencia de jugadores de 150 milisegundos.
- Si el emparejamiento no se completa rápidamente, suavice los requisitos para completar un emparejamiento dentro de un tiempo razonable.
  - Después de 10 segundos, acepte un equipo con 150 jugadores.
  - Después de 12 segundos, aumente el máximo aceptable de latencia a 200 milisegundos.
  - Después de 15 segundos, acepte un equipo con 100 jugadores.

Notas sobre el uso de este conjunto de reglas:

- Puesto que el algoritmo utiliza la preferencia por lotes "largest population" (mayor población), los jugadores se ordenan primero según los criterios de equilibrio. Como resultado, los emparejamientos suelen estar más completos y contener jugadores con habilidades más similares. Todos los jugadores cumplirán los requisitos de latencia aceptable, pero es posible que no obtengan la mejor latencia posible para su ubicación.

- La estrategia de algoritmo que se utiliza en este conjunto de reglas, "largest population" (mayor población), es la configuración predeterminada. Para utilizar el valor predeterminado, puede optar por omitir la configuración.
- Si ha habilitado la reposición de emparejamiento, no suavice los requisitos de número de jugadores demasiado rápido. De ser así, podría acabar con demasiadas sesiones de juego parcialmente completas. Obtenga más información en Reducción de los requisitos de emparejamientos de gran tamaño.

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

Ejemplo: crea una partida grande con varios equipos

En este ejemplo se muestra cómo configurar un conjunto de reglas para emparejamientos con varios equipos que pueden superar los 40 jugadores. En él se explica cómo crear varios equipos idénticos con una definición y cómo se pueblan los equipos con un tamaño asimétrico durante la creación del emparejamiento.

El conjunto de reglas de ejemplo crea un emparejamiento mediante las siguientes instrucciones:

- Cree diez equipos de "cazadores" idénticos con un máximo de 15 jugadores y un equipo de "monstruos" con exactamente 5 jugadores.
- Criterios de equilibrio: seleccione jugadores en función del número de monstruos matados. Si los jugadores no informan del número de muertes, utilice el valor predeterminado de 5.
- Preferencia por lotes: agrupe los jugadores en función de las regiones en las que informan de la latencia de jugador más rápida.
- Regla de latencia: establece un máximo de latencia de jugadores aceptable de 200 milisegundos.
- Si el emparejamiento no se completa rápidamente, suavice los requisitos para completar un emparejamiento dentro de un tiempo razonable.
  - Después de 15 segundos, acepte los equipos con 10 jugadores.
  - Después de 20 segundos, acepte los equipos con 8 jugadores.

Notas sobre el uso de este conjunto de reglas:

- Este conjunto de reglas define a los equipos que pueden albergar hasta 155 jugadores, lo que los convierte en un emparejamiento de gran tamaño. (10 x 15 cazadores + 5 monstruos = 155)
- Puesto que el algoritmo utiliza la preferencia por lotes de "región más rápida", los jugadores suelen colocarse en regiones en las que informan de una latencia más rápida y no en regiones en las que informan de una latencia alta (pero aceptable). Al mismo tiempo, es probable que los

emparejamientos tengan menos jugadores y que los criterios de equilibrio (número de habilidades de los monstruos) varíen de forma más amplia.

- Cuando se define una expansión para una definición de varios equipos (cantidad > 1), la expansión se aplica a todos los equipos creados con esa definición. Por lo tanto, si se reduce la configuración de jugadores mínimos del equipo de cazadores, los diez equipos de cazadores se verán afectados de igual manera.
- Puesto que este conjunto de reglas está optimizado para minimizar la latencia de los jugadores, la regla de latencia funciona como un cajón para excluir a los jugadores que no tienen opciones de conexión aceptables. No tenemos que reducir este requisito.
- A continuación se explica cómo FlexMatch realiza los emparejamientos para este conjunto de reglas antes de que las expansiones surtan efecto.
  - Ningún equipo ha alcanzado aún el recuento de minPlayers. Los equipos de cazadores cuentan con 15 ranuras abiertas, mientras que el equipo de monstruos dispone de 5.
    - Se asignan los primeros 100 jugadores (10 en cada uno) en los diez equipos de cazadores.
    - Los siguientes 22 jugadores se asignarán de forma secuencial (2 en cada uno) en los equipos de cazadores y de monstruos.
  - Los equipos de cazadores alcanzaron el recuento de minPlayers de 12 jugadores cada uno. El equipo de monstruos dispone de dos jugadores y no ha alcanzado el recuento de minPlayers.
    - Los siguientes tres jugadores se asignan al equipo de monstruos.
  - Todos los equipos han alcanzado el recuento de minPlayers. Los equipos de cazadores tienen tres ranuras abiertas cada uno. El equipo de monstruos está completo.
    - Los últimos 30 jugadores se asignan de forma secuencial a los equipos de cazadores, lo que garantiza que todos los equipos de cazadores tienen casi el mismo tamaño (un jugador más o menos).
- Si ha habilitado la reposición para emparejamientos creados con este conjunto de reglas, no reduzca los requisitos de número de jugadores demasiado rápido. De ser así, podría acabar con demasiadas sesiones de juego parcialmente completas. Obtenga más información en Reducción de los requisitos de emparejamientos de gran tamaño.

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

Ejemplo: crea una partida grande con jugadores con atributos similares

En este ejemplo se muestra cómo configurar un conjunto de reglas para emparejamientos con dos equipos que utilizan `batchDistance`. En el ejemplo:

- La regla `SimilarLeague` garantiza que todos los jugadores de un emparejamiento tengan una `league` dentro de otros dos jugadores.

- La regla `SimilarSkill` garantiza que todos los jugadores de un emparejamiento tengan una `skill` dentro de otros diez jugadores Si un jugador ha estado esperando 10 segundos, la distancia se amplía a 20. Si un jugador ha estado esperando 20 segundos, la distancia se amplía a 40.
- La regla `SameMap` garantiza que todos los jugadores de un emparejamiento hayan solicitado el mismo `map`.
- La regla `SameMode` garantiza que todos los jugadores de un emparejamiento hayan solicitado el mismo `mode`.

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

Ejemplo: usa una regla compuesta para crear una partida con jugadores con atributos o selecciones similares

En este ejemplo se muestra cómo configurar un conjunto de reglas para emparejamientos con dos equipos que utilizan `compound`. En el ejemplo:

- La regla `SimilarLeagueDistance` garantiza que todos los jugadores de un emparejamiento tengan una `league` dentro de otros dos jugadores.
- La regla `SimilarSkillDistance` garantiza que todos los jugadores de un emparejamiento tengan una `skill` dentro de otros diez jugadores Si un jugador ha estado esperando 10 segundos, la distancia se amplía a 20. Si un jugador ha estado esperando 20 segundos, la distancia se amplía a 40.

- La regla `SameMapComparison` garantiza que todos los jugadores de un emparejamiento hayan solicitado el mismo `map`.
- La regla `SameModeComparison` garantiza que todos los jugadores de un emparejamiento hayan solicitado el mismo `mode`.
- La regla `CompoundRuleMatchmaker` garantiza un emparejamiento si se cumple al menos una de las siguientes condiciones:
  - Los jugadores de un emparejamiento han solicitado el mismo `map` y el mismo `mode`.
  - Los jugadores de un emparejamiento tienen atributos `skill` y `league` comparables.

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

Ejemplo: crea una regla que utilice la lista de bloqueados de un jugador

Este ejemplo muestra un conjunto de reglas que permite a los jugadores evitar el emparejamiento con otros jugadores. Los jugadores pueden crear una lista de bloqueados, que el emparejador evalúa durante la selección del jugador para un emparejamiento. Para obtener más información

sobre cómo añadir una lista de bloqueados o una característica de lista de evitación, consulte el blog de AWS para videojuegos.

En este ejemplo se establecen las siguientes instrucciones:

- Cree dos equipos de exactamente cinco jugadores.
- Introduce la lista de bloqueados de un jugador, que es una lista de jugadores IDs (hasta 100).
- Compara a todos los jugadores con la lista de bloqueados de cada jugador y rechaza una partida propuesta si IDs encuentras algún jugador bloqueado.

Notas sobre el uso de este conjunto de reglas:

- Al evaluar a un nuevo jugador para añadirlo a un emparejamiento propuesto (o para cubrir una plaza en un emparejamiento ya existente), es posible que el jugador sea rechazado por alguna de las siguientes razones:
  - Si el nuevo jugador está en la lista de bloqueados de algún jugador que ya esté seleccionado para el emparejamiento.
  - Si los jugadores que ya están seleccionados en el emparejamiento están en la lista de bloqueados del nuevo jugador.
- Como se muestra, este conjunto de reglas impide emparejar a un jugador con cualquier jugador de su lista de bloqueados. Puede cambiar este requisito por una lista de preferencias (también llamada lista de «evitación») añadiendo una expansión de reglas y aumentando el valor `maxCount`.

```
{
    "name": "Player Block List",
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "red"
    }, {
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "blue"
    }],
    "playerAttributes": [{
        "name": "BlockList",
```

```
        "type": "string_list",
        "default": []
    }],
    "rules": [{
        "name": "PlayerIdNotInBlockList",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": "flatten(teams[*].players.attributes[BlockList])",
        "referenceValue": "flatten(teams[*].players[playerId])",
        "maxCount": 0
    }]
}
```

Creación de una configuración de emparejamiento

Para configurar un Amazon GameLift Servers FlexMatch emparejador para procesar las solicitudes de emparejamiento, crea una configuración de emparejamiento. Usa la Amazon GameLift Servers consola o el (). AWS Command Line Interface AWS CLI Para obtener más información sobre la creación de un emparejamiento, consulte Diseña un FlexMatch emparejador.

Temas

- Tutorial: Crea un emparejador para hospedar Amazon GameLift Servers
- Tutorial: Crea un emparejador de forma independiente FlexMatch
- Tutorial: Edición de una configuración de emparejamiento

Tutorial: Crea un emparejador para hospedar Amazon GameLift Servers

Antes de crear una configuración de emparejamiento, crea un conjunto de reglas y una cola de sesiones de Amazon GameLift Servers juego para usarlos con el emparejador.

Console

1. En la Amazon GameLift Serversconsola, en el panel de navegación, selecciona Configuraciones de emparejamiento.
2. Cambia a la AWS región en la que quieras crear tu matchmaker.
3. En la página Configuraciones de emparejamiento, elija Crear una configuración de emparejamiento.

4. En la página Definir detalles de configuración, en Detalles de la configuración de emparejamiento, realice el siguiente procedimiento:

   a. En Nombre, introduzca un nombre de emparejador que le ayude a identificarlo en una lista y en las métricas. El nombre del emparejador debe ser único dentro de una región. Las solicitudes de emparejamiento identifican el emparejador que se debe utilizar por su nombre y región.

   b. En Descripción, añada una descripción que ayude a identificar al emparejador (opcional).

   c. En Conjunto de reglas, elija un conjunto de reglas de la lista para utilizarlo con el emparejador. La lista contiene todos los conjuntos de reglas que se han creado en la región actual.

   d. Para el FlexMatchmodo, selecciona Gestionado para el alojamiento Amazon GameLift Servers gestionado. Este modo solicita que se FlexMatch pasen las partidas exitosas a la cola de sesiones de juego especificada.

   e. En AWS Región, elija la región en la que configuró la cola de sesiones de juego que quiera utilizar con el emparejador.

   f. En Cola, elija la cola de la sesión de juego que quiera utilizar con el emparejador.

5. Elija Siguiente.

6. En la página Configurar los ajustes, en Configuración de emparejamiento, realice el siguiente procedimiento:

   a. En el apartado Tiempo de espera de las solicitudes, establece el tiempo máximo, en segundos, para que el emparejador complete una partida para cada solicitud. FlexMatchcancela las solicitudes de emparejamiento que superen este tiempo.

   b. En Modo de reposición, elija un modo para la gestión de las reposiciones de emparejamiento.

      - Para activar la característica de reposición automática, elija Automático.
      - Para crear su propia administración de solicitudes de reposición o no utilizar la característica de reposición, seleccione Manual.

   c. (Opcional) Para aumentar el número de jugadores, establece el número de casillas que se deben mantener abiertas en una partida. FlexMatchpuede llenar estos espacios con jugadores en el futuro.

   d. En Opciones de aceptación de coincidencias, en Aceptación necesaria, si desea que todos los jugadores de un emparejamiento propuesto acepten activamente su

participación en el emparejamiento, seleccione Obligatorio (opcional). Si selecciona esta opción, establezca en Tiempo de espera de aceptación cuánto tiempo, en segundos, quiere que el emparejador espere la aceptación de los jugadores antes de cancelar el emparejamiento.

7. En Configuración de notificaciones de eventos, realice el siguiente procedimiento:

   a. En Tema de SNS, elija un tema de Amazon Simple Notification Service (Amazon SNS) para recibir notificaciones de eventos de emparejamiento (opcional). Si aún no ha configurado un tema de SNS, puede añadirlo más tarde editando la configuración de emparejamiento. Para obtener más información, consulte Configuración FlexMatch notificaciones de eventos.

   b. (Opcional) Para los datos de eventos personalizados, introduce cualquier dato personalizado que quieras asociar a este emparejador en la mensajería del evento. FlexMatchincluye estos datos en todos los eventos asociados al emparejador.

8. Expanda Datos adicionales del juego y, a continuación, realice el siguiente procedimiento (opcional):

   a. (Opcional) En cuanto a los datos de la sesión de juego, introduce cualquier información adicional relacionada con el juego que quieras incluir FlexMatch en las nuevas sesiones de juego que comiencen con partidas realizadas con esta configuración de emparejamiento.

   b. En Propiedades del juego, añada propiedades de pares clave-valor que contengan información sobre una nueva sesión de juego (opcional).

9. (Opcional) En Etiquetas, añade etiquetas para ayudarte a gestionar y hacer un seguimiento de tus recursos. AWS

10. Elija Siguiente.

11. En la página Revisar y crear, revise sus opciones y, a continuación, elija Crear. Si la creación se realiza correctamente, el emparejador estará listo para aceptar solicitudes de emparejamiento.

AWS CLI

Para crear una configuración de emparejamiento con AWS CLI, abre una ventana de línea de comandos y usa el create-matchmaking-configurationcomando para definir un nuevo emparejador.

Este comando de ejemplo crea una nueva configuración de emparejamiento que requiere la aceptación del jugador y habilita la reposición automática. También reserva espacios para dos jugadores para FlexMatch añadirlos más tarde y proporciona algunos datos de la sesión del juego.

```
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"
```

Si la solicitud de creación de la configuración de emparejamiento es correcta, Amazon GameLift Servers devuelve un MatchmakingConfigurationobjeto con los ajustes que solicitaste para el emparejador. El nuevo creador de emparejamientos está listo para aceptar solicitudes de emparejamiento.

Tutorial: Crea un emparejador de forma independiente FlexMatch

Antes de crear una configuración de emparejamiento, cree un conjunto de reglas que utilizar con el emparejador.

Console

1. Abre la Amazon GameLift Servers consola en casa. https://console.aws.amazon.com/gamelift/
2. Cambia a la AWS región en la que quieras crear tu matchmaker. Para ver una lista de las regiones que admiten configuraciones de FlexMatch emparejamiento, consulta. Elección de una ubicación para el emparejador
3. En el panel de navegación FlexMatch, selecciona Configuraciones de emparejamiento.

4. En la página Configuraciones de emparejamiento, elija Crear una configuración de emparejamiento.
5. En la página Definir detalles de configuración, en Detalles de la configuración de emparejamiento, realice el siguiente procedimiento:

   a. En Nombre, introduzca un nombre de emparejador que le ayude a identificarlo en una lista y en las métricas. El nombre del emparejador debe ser único dentro de una región. Las solicitudes de emparejamiento identifican el emparejador que se debe utilizar por su nombre y región.

   b. En Descripción, añada una descripción que ayude a identificar al emparejador (opcional).

   c. En Conjunto de reglas, elija un conjunto de reglas de la lista para utilizarlo con el emparejador. La lista contiene todos los conjuntos de reglas que se han creado en la región actual.

   d. Para FlexMatchel modo, selecciona Independiente. Esto indica que tienes un mecanismo personalizado para iniciar nuevas sesiones de juego en una solución de Amazon GameLift Servers alojamiento externa a.
6. Elija Siguiente.
7. En la página Configurar los ajustes, en Configuración de emparejamiento, realice el siguiente procedimiento:

   a. En Solicitar tiempo de espera, establezca el tiempo máximo, en segundos, para que el emparejador complete un emparejamiento para cada solicitud. Las solicitudes de emparejamiento que superan este tiempo se rechazan.

   b. En Opciones de aceptación de coincidencias, en Aceptación necesaria, si desea que todos los jugadores de un emparejamiento propuesto acepten activamente su participación en el emparejamiento, seleccione Obligatorio (opcional). Si selecciona esta opción, establezca en Tiempo de espera de aceptación cuánto tiempo, en segundos, quiere que el emparejador espere la aceptación de los jugadores antes de cancelar el emparejamiento.
8. En Configuración de notificaciones de eventos, realice el siguiente procedimiento:

   a. En Tema de SNS, elija un tema de Amazon SNS para recibir las notificaciones de eventos de emparejamiento (opcional). Si aún no ha configurado un tema de SNS, puede añadirlo más tarde editando la configuración de emparejamiento. Para obtener más información, consulte Configuración FlexMatch notificaciones de eventos.

b. (Opcional) En el caso de los datos de eventos personalizados, introduce cualquier dato personalizado que desees asociar a este emparejador en la mensajería del evento. FlexMatchincluye estos datos en todos los eventos asociados al emparejador.

9. (Opcional) En Etiquetas, añade etiquetas para ayudarte a gestionar y realizar un seguimiento de tus AWS recursos.
10. Elija Siguiente.
11. En la página Revisar y crear, revise sus opciones y, a continuación, elija Crear. Si la creación se realiza correctamente, el emparejador estará listo para aceptar solicitudes de emparejamiento.

AWS CLI

Para crear una configuración de emparejamiento con AWS CLI, abre una ventana de línea de comandos y usa el create-matchmaking-configurationcomando para definir un nuevo emparejador.

Este comando de ejemplo crea una nueva configuración de emparejamiento para un emparejador independiente que requiere la aceptación del jugador.

```
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"
```

Si la solicitud de creación de la configuración de emparejamiento es correcta, Amazon GameLift Servers devuelve un MatchmakingConfigurationobjeto con los ajustes que solicitaste para el emparejador. El nuevo creador de emparejamientos está listo para aceptar solicitudes de emparejamiento.

Tutorial: Edición de una configuración de emparejamiento

Para editar una configuración de emparejamiento, seleccione Configuraciones de emparejamiento en la barra de navegación y elija la configuración que desee editar. Puede actualizar cualquier campo de una configuración existente excepto su nombre.

Al actualizar un conjunto de reglas de configuración, un nuevo conjunto de reglas puede ser incompatible si hay tickets de emparejamiento activos por los siguientes motivos:

- Nombres o número de equipos nuevos o diferentes
- Nuevos atributos de jugador
- Cambios en los tipos de atributos de los jugadores existentes

Para realizar cualquiera de estos cambios en el conjunto de reglas, cree una nueva configuración de emparejamiento con el conjunto de reglas actualizado.

Configuración FlexMatch notificaciones de eventos

Puede utilizar notificaciones de eventos para realizar un seguimiento del estado de las solicitudes de emparejamiento individuales. Todos los juegos en producción, o en preproducción con actividad de emparejamiento de alto volumen, deben utilizar las notificaciones de eventos.

Existen dos opciones para configurar las notificaciones de eventos.

- Haga que el emparejador publique notificaciones de eventos en un tema de Amazon Simple Notification Service (Amazon SNS).
- Utilice EventBridge los eventos de Amazon publicados automáticamente y su conjunto de herramientas para gestionar los eventos.

Para obtener una lista de FlexMatch eventos que Amazon GameLift Servers emite, miraFlexMatch eventos de emparejamiento.

Temas

- Configura eventos EventBridge
- Tutorial: Configuración de un tema de Amazon SNS
- Configuración de un tema de SNS con cifrado del servidor
- Configuración de una suscripción a un tema para invocar una función de Lambda

Configura eventos EventBridge

Amazon GameLift Servers publica automáticamente todos los eventos de emparejamiento en Amazon EventBridge. Con EventBridge, puedes configurar reglas para que los eventos de emparejamiento se envíen a los objetivos para su procesamiento. Por ejemplo, puedes establecer una regla para dirigir el evento «PotentialMatchCreated» a una AWS Lambda función que gestione las aceptaciones de los jugadores. Para obtener más información, consulta ¿Qué es Amazon EventBridge?

> **Note**
>
> Cuando configures tus matchmakers, deja vacío el campo de destino de la notificación o haz referencia a un tema de SNS si quieres usar tanto EventBridge Amazon SNS como Amazon SNS.

Tutorial: Configuración de un tema de Amazon SNS

Puedes tener Amazon GameLift Servers publicar todos los eventos que un FlexMatch matchmaker genera en un tema de Amazon SNS.

Para crear un tema de SNS para Amazon GameLift Servers notificaciones de eventos

1. Abra la consola de Amazon SNS.
2. En el panel de navegación, elija Temas.
3. En la página Temas, elija Crear tema.
4. Cree un tema en la consola de . Para obtener más información, consulte Para crear un tema mediante la AWS Management Console en la Guía para desarrolladores de Amazon Simple Notification Service.
5. En la página Detalles del tema, elija Editar.
6. En la página Editar del tema, expanda Política de acceso y, a continuación, añada la sintaxis en negrita de la siguiente declaración de política de AWS Identity and Access Management (IAM) al final de la política existente (opcional). Se muestra la política completa aquí para mayor claridad. Asegúrese de utilizar los detalles del nombre de recurso de Amazon (ARN) para su propio tema de SNS y Amazon GameLift Servers configuración de emparejamiento.

```
{
```

```
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
        }
      }
    }
  ]
```

```
}
```

7. Elija Guardar cambios.

Configuración de un tema de SNS con cifrado del servidor

Puede utilizar el cifrado del servidor (SSE) para almacenar datos confidenciales en temas cifrados. SSE protege el contenido de los mensajes de los temas de Amazon SNS mediante claves administradas en AWS Key Management Service ()AWS KMS. Para obtener más información sobre el cifrado del servidor con Amazon SNS, consulte Cifrado en reposo en la Guía para desarrolladores de Amazon Simple Notification Service.

Para configurar un tema de SNS con cifrado del servidor, revise los temas siguientes:

- Creación de claves en la Guía para desarrolladores de AWS Key Management Service .
- Habilitación de SSE para un tema en la Guía para desarrolladores de Amazon Simple Notification Service

Al crear la clave de KMS, utilice la siguiente política de claves de KMS:

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

```
}
```

Configuración de una suscripción a un tema para invocar una función de Lambda

Puede invocar una función de Lambda mediante notificaciones de evento publicadas en su tema de Amazon SNS. Al configurar el emparejador, asegúrese de establecer el destino de notificación en un ARN del tema de SNS.

La siguiente AWS CloudFormation plantilla configura una suscripción a un tema de SNS denominado para `MyFlexMatchEventTopic` invocar una función de Lambda denominada. `FlexMatchEventHandlerLambdaFunction` La plantilla crea una política de permisos de IAM que permite Amazon GameLift Servers para escribir en el tema de SNS. A continuación, la plantilla añade permisos para que el tema de SNS invoque la función de 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 tu juego para FlexMatch

Uso Amazon GameLift Servers FlexMatch para añadir la funcionalidad de emparejamiento de jugadores a tus juegos. Puede usar… FlexMatch con un gestionado Amazon GameLift Servers solución de alojamiento o como un servicio independiente con otra solución de alojamiento. Si quieres añadir FlexMatch a un Amazon GameLift Servers FleetIQ solución, utilícela como un servicio independiente. Para obtener más información sobre cómo FlexMatch funciona, consulteCómo Amazon GameLift ServersFlexMatch funciona.

Las soluciones de emparejamiento requieren el siguiente trabajo:

- Crea un emparejador con tus reglas de emparejamiento personalizadas Para obtener más información sobre cómo crear el emparejador, ver. Construir un casamentero Amazon GameLift ServersFlexMatch
- Actualiza tu cliente de juego para permitir que los jugadores soliciten una partida.
- Para juegos que utilizan Amazon GameLift Servers hosting, actualiza tu servidor de juegos para gestionar los datos de los partidos y, si lo prefieres, rellena los espacios vacíos en los partidos.

En los temas de esta sección se explica cómo añadir compatibilidad con el matchmaking a tus clientes y servidores de juegos.

Consulta la hoja de ruta que prefieras FlexMatch solución de emparejamiento:

- Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers
- Hoja de ruta: Cree una solución de emparejamiento independiente con FlexMatch

Añadir FlexMatch a un cliente de juego

En este tema se describe cómo añadir la funcionalidad de FlexMatch emparejamiento a los componentes del juego del lado del cliente.

Te recomendamos encarecidamente que tu cliente de juego realice las solicitudes de emparejamiento a través de un servicio de juego secundario. Si utilizas esta fuente de confianza para comunicarte con el Amazon GameLift Servers servicio, podrás protegerte más fácilmente de los intentos de hackeo y de los datos falsos de los jugadores. Si su juego dispone de un servicio de directorio de sesiones, esta es una buena opción para controlar las solicitudes de creación de emparejamientos. Se recomienda utilizar un servicio de juegos de fondo para todas las llamadas al

Amazon GameLift Servers servicio si se utiliza FlexMatch junto con el Amazon GameLift Servers alojamiento y como un servicio independiente.

Las actualizaciones del lado del cliente son obligatorias tanto si se utiliza FlexMatch con un alojamiento Amazon GameLift Servers gestionado como si se trata de un servicio independiente con otra solución de alojamiento. Con la API de servicioAmazon GameLift Servers, que forma parte del AWS SDK, agrega las siguientes funciones:

- Solicita partidas para uno o varios jugadores (obligatorio). En función de tu conjunto de reglas de emparejamiento, esta solicitud puede requerir ciertos datos específicos del jugador, incluidos los atributos del jugador y la latencia.
- Realiza un seguimiento del estado de una solicitud de emparejamiento (obligatorio). En general, esta tarea requiere configurar la notificación de eventos.
- Solicite la aceptación de un jugador para una propuesta de emparejamiento (opcional). Esta función requiere una interacción adicional con el jugador para mostrar los detalles del partido y permitirle aceptarlo o rechazarlo.
- Obtén información sobre la conexión de la sesión de juego y únete a la partida (obligatorio). Cuando se haya iniciado una sesión de juego para la nueva partida, recupera la información de conexión de la sesión de juego y úsala para conectarte a la sesión de juego.

Tareas imprescindibles del lado del cliente

Antes de poder añadir la funcionalidad del lado del cliente a tu juego, debes realizar estas tareas:

- Agrega el AWS SDK a tu servicio de backend. Tu servicio de backend utiliza la funcionalidad de la Amazon GameLift Servers API, que forma parte del AWS SDK. Consulta Amazon GameLift Servers SDKs los servicios de cliente para obtener más información sobre el AWS SDK y descargar la versión más reciente. Para ver las descripciones y la funcionalidad de las API, consulteAmazon GameLift ServersFlexMatchReferencia de API (AWS SDK).
- Configure un sistema de tickets de emparejamiento. Todas las solicitudes de emparejamiento deben tener un identificador de ticket único. Crea un mecanismo para generar entradas únicas IDs y asígnalas a las solicitudes de partidos. Un ID de ticket puede usar cualquier formato de cadena, hasta un máximo de 128 caracteres.
- Recopila información sobre tu casamentero. Obtén la siguiente información de tu configuración y conjunto de reglas de emparejamiento.
  - Nombre del recurso de configuración de emparejamiento.

  - La lista de atributos de los jugadores, que se definen en el conjunto de reglas.
- Recupera los datos del jugador. Configura una forma de obtener datos relevantes para que cada jugador los incluya en tus solicitudes de emparejamiento. Necesitas el ID del jugador y los valores de los atributos del jugador. Si tu conjunto de reglas tiene reglas de latencia o quieres usar los datos de latencia al organizar las sesiones de juego, recopila los datos de latencia de cada ubicación geográfica en la que es probable que el jugador esté incluido en una partida. Para obtener mediciones de latencia precisas, usa las balizas Amazon GameLift Servers de ping UDP. Estos puntos de conexión permiten medir la latencia real de la red UDP entre los dispositivos reproductores y cada una de las posibles ubicaciones de alojamiento, lo que permite tomar decisiones de ubicación más precisas que con los pings ICMP. Para obtener más información sobre el uso de balizas de ping UDP para medir la latencia, consulta la sección Balizas de ping UDP.

Solicitud del emparejamiento de jugadores

Agrega código al servicio de backend de tu juego para gestionar las solicitudes de emparejamiento a un emparejador. FlexMatch El proceso de solicitud de FlexMatch emparejamiento es idéntico para los juegos que se utilizan FlexMatch con Amazon GameLift Servers alojamiento y para los juegos que se utilizan FlexMatch como solución independiente.

Para crear una solicitud de emparejamiento, sigue estos pasos:

Llama a la Amazon GameLift Servers API. StartMatchmaking Cada solicitud debe incluir la siguiente información.

Creador de emparejamientos

El nombre de la configuración de la creación de emparejamiento que desea utilizar para la solicitud. FlexMatch coloca cada solicitud en el grupo del creador de emparejamientos especificado y la solicitud se procesa en función de la configuración del creador de emparejamientos. Esto incluye la aplicación de un plazo, ya sea para solicitar la aceptación del jugador de los emparejamientos, qué cola utilizar cuando se coloque una sesión de juego resultante, etc. Para obtener más información sobre los creadores de emparejamientos y las reglas, consulte Diseña un FlexMatch emparejador.

ID del ticket

Un ID de ticket único asignado a la solicitud. Todo lo relacionado con la solicitud, incluidos los eventos y las notificaciones, hará referencia al ID de ticket.

Datos del jugador

Lista de jugadores para los que se desea crear un emparejamiento. Si alguno de los jugadores de la solicitud no cumple los requisitos de emparejamiento, según las reglas de emparejamiento y los mínimos de latencia, la solicitud de emparejamiento nunca dará lugar a un emparejamiento correcto. Puede incluir hasta diez jugadores en una solicitud de emparejamiento. Cuando hay varios jugadores en una solicitud, FlexMatch intenta crear un único emparejamiento y asignar todos los jugadores al mismo equipo (seleccionado de forma aleatoria). Si una solicitud contiene demasiados jugadores, por lo que no caben en uno de los equipos de emparejamiento, la solicitud no podrá emparejarse. Por ejemplo, si ha configurado el creador de emparejamientos de tal forma que cree emparejamientos 2v2 (dos equipos de dos jugadores), no puede enviar una solicitud de emparejamiento que contenga más de dos jugadores.

> **Note**
>
> Un jugador (identificado por su ID de jugador) solo puede incluirse en una solicitud de emparejamiento activa en cada momento. Si se crea una solicitud nueva para un jugador, cualquier ticket de creación de emparejamientos activo con el mismo ID de jugador se cancela automáticamente.

Para cada jugador que aparezca, incluya los siguientes datos:

- ID de jugador: cada jugador debe tener un ID de jugador único generado por usted. Consulte Generar reproductor IDs.
- Atributos de jugador: si el emparejador en uso llama a atributos de jugador, la solicitud debe proporcionar esos atributos para cada jugador. Los atributos necesarios de los jugadores se definen en el conjunto de reglas del creador de emparejamientos, que también especifica el tipo de datos para el atributo. Un atributo de jugador es opcional solo cuando el conjunto de reglas especifica un valor predeterminado para el atributo. Si la solicitud de emparejamiento no proporciona atributos de jugador necesarios para todos los jugadores, la solicitud de creación de emparejamientos no podrá ser nunca correcta. Para obtener más información sobre los conjuntos de reglas del creador de emparejamiento y los atributos de jugador, consulte Construye un FlexMatch conjunto de reglas y FlexMatchejemplos de conjuntos de reglas.
- Latencias del jugador: si el emparejador en uso tiene una regla de latencias del jugador, la solicitud debe informar sobre la latencia de cada jugador. Los datos de latencias del jugador son una lista de uno o varios valores por jugador. Representa la latencia que el jugador experimenta para las regiones en la cola del creador de emparejamientos. Si no hay valores

de latencia para un jugador incluidos en la solicitud, el jugador no podrá emparejarse y se producirá un error en la solicitud. Para obtener mediciones de latencia precisas, utilice las balizas Amazon GameLift Servers de ping UDP. Estos puntos de conexión permiten medir la latencia real de la red UDP entre los dispositivos del reproductor y una posible ubicación de alojamiento, lo que permite tomar decisiones de ubicación más precisas que con los pings ICMP. Para obtener más información sobre el uso de balizas de ping UDP para medir la latencia, consulta la sección Balizas de ping UDP.

Para recuperar los detalles de la solicitud de coincidencia

Después de enviar una solicitud de partido, puedes ver los detalles de la solicitud llamando DescribeMatchmakingcon el identificador de la entrada de la solicitud. Esta llamada devuelve la información de solicitud, incluido el estado actual. Una vez que se ha completado correctamente la solicitud, el ticket también contiene la información que necesita un cliente de juego para conectarse al emparejamiento.

Para cancelar una solicitud de partido

Puedes cancelar una solicitud de emparejamiento en cualquier momento llamando StopMatchmakingcon el identificador de entrada de la solicitud.

Seguimiento de eventos de emparejamiento

Configure notificaciones para realizar un seguimiento de eventos que Amazon GameLift Servers emite para procesos de creación de emparejamientos. Puedes configurar las notificaciones directamente, mediante la creación de un tema de SNS o a través de Amazon EventBridge. Para obtener más información sobre la configuración de notificaciones, consulte Configuración FlexMatch notificaciones de eventos. Una vez configuradas las notificaciones, añada un agente de escucha en su servicio de cliente para detectar los eventos y responder según sea necesario.

También es recomendable hacer copias de seguridad de las notificaciones mediante el sondeo periódico de actualizaciones de estado cuando transcurre un período significativo de tiempo sin notificaciones. Para minimizar el impacto en el rendimiento de emparejamiento, asegúrese de sondear solo después de esperar al menos 30 segundos después de que se haya enviado el ticket de emparejamiento o después de la última notificación recibida.

Para recuperar un ticket de solicitud de emparejamiento, incluido el estado actual, llama DescribeMatchmakingcon el identificador del ticket de la solicitud. Recomendamos sondear no más

de una vez cada 10 segundos. Este enfoque se utiliza únicamente en escenarios de desarrollo de bajo volumen.

> **Note**
>
> Debe configurar su juego con notificaciones de eventos antes de tener un uso de emparejamiento de gran volumen, como, por ejemplo, con pruebas de carga de preproducción. Todos los juegos en la versión pública deben usar notificaciones independientemente del volumen. El enfoque de sondeo continuo solo resulta apropiado para juegos en desarrollo con bajo uso de emparejamiento.

Solicitud de aceptación del jugador

Si utiliza un creador de emparejamientos que tenga la aceptación de jugador activada, añada el código a su servicio de cliente para administrar el proceso de aceptación del jugador. El proceso de gestión de las aceptaciones de jugadores es el mismo para los juegos que se utilizan FlexMatch con un alojamiento Amazon GameLift Servers gestionado y para los juegos que se utilizan FlexMatch como una solución independiente.

Solicite la aceptación del jugador para una propuesta de emparejamiento.

1. Detecte cuando una propuesta de emparejamiento necesite la aceptación del jugador. Supervise el ticket de creación de emparejamientos para detectar cuándo el estado cambia a `REQUIRES_ACCEPTANCE`. Un cambio en este estado desencadena el evento. FlexMatch `MatchmakingRequiresAcceptance`
2. Obtener aceptaciones de todos los jugadores. Cree un mecanismo para presentar los detalles del emparejamiento propuesto a cada jugador en el ticket de creación de emparejamiento. Los jugadores deben poder indicar que aceptan o rechazan el emparejamiento propuesto. Puedes recuperar los detalles del partido llamando DescribeMatchmaking. Los jugadores tienen un tiempo limitado para responder antes de que el creador de emparejamientos retire el emparejamiento propuesto y continúe.
3. Informar de las respuestas de los jugadores a FlexMatch. Reporta las respuestas de los jugadores llamando AcceptMatchcon la opción de aceptar o rechazar. Todos los jugadores en una solicitud de emparejamiento deben aceptar el emparejamiento para que avance.
4. Administrar tickets con aceptaciones erróneas. Se produce un error en la solicitud cuando un jugador del emparejamiento propuesto rechaza el emparejamiento o no responde antes

de que transcurra el plazo de aceptación. Los tickets de los jugadores que aceptaron el emparejamiento se devuelven automáticamente al grupo de tickets. Los tickets de los jugadores que no aceptaron el emparejamiento pasan al estado de ERROR y dejarán de procesarse. En el caso de los tickets con varios jugadores, si alguno de los jugadores del ticket no acepta el emparejamiento, se pierde todo el ticket.

Conexión a un emparejamiento

Añada código a su servicio de cliente para administrar los emparejamientos formados correctamente (estado `COMPLETED` o evento `MatchmakingSucceeded`). Esto incluye la notificación de los jugadores del emparejamiento y el envío de información de la conexión a los clientes de juego.

En el caso de los juegos que utilizan un alojamiento Amazon GameLift Servers gestionado, cuando una solicitud de emparejamiento se tramita correctamente, la información de conexión de la sesión de juego se añade a la ficha de emparejamiento. Llama para recuperar un ticket de emparejamiento completo. DescribeMatchmaking La información de conexión incluye la dirección IP y el puerto de la sesión de juego, así como un ID de sesión de jugador para cada ID de jugador. Obtenga más información en GameSessionConnectionInfo. Su cliente de juegos puede utilizar esta información para conectarse directamente a la sesión de juego para el emparejamiento. Una solicitud de conexión debe incluir un ID de jugador y un ID de sesión de jugador. Estos datos asocian al jugador conectado a los datos del partido de la sesión de juego, que incluyen las asignaciones de los equipos (consulte GameSession).

En el caso de los juegos que utilizan otras soluciones de alojamiento Amazon GameLift ServersFleetIQ, por ejemplo, debes incorporar un mecanismo que permita a los jugadores conectarse a la sesión de juego correspondiente.

Ejemplo de solicitudes de emparejamiento

Los siguientes fragmentos de código crean solicitudes de emparejamiento para diferentes emparejamientos. Como se ha indicado anteriormente, una solicitud debe proporcionar los atributos de jugador que necesita el creador de emparejamientos en uso, tal y como se definió en el conjunto de reglas del creador de emparejamientos. El atributo proporcionado debe utilizar el mismo tipo de datos, el número (N) o la cadena (S) que se defina en el conjunto de reglas.

```
# 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)
```

Añadir FlexMatch a un servidor Amazon GameLift Servers de juegos hospedado

Cuando Amazon GameLift Servers crea una partida, genera un conjunto de datos de resultados de la partida que describen los detalles clave del emparejamiento, incluidas las asignaciones de los equipos. Un servidor de juego utiliza estos datos, así como otra información de la sesión de juego, al iniciar una nueva sesión de juego para organizar el partido.

Para los servidores de juegos que están alojados en Amazon GameLift Servers

Amazon GameLift ServersIndica a un servidor de juegos que procese iniciar una sesión de juego. Incluye un GameSessionobjeto que describe el tipo de sesión de juego que se va a crear e incluye información específica del jugador, incluidos los datos del partido.

Para servidores de juegos que están alojados en otras soluciones

Tras completar correctamente una solicitud de emparejamiento, Amazon GameLift Servers emite un evento que incluye los resultados de la partida. Puedes usar estos datos con tu propia solución de alojamiento para iniciar una sesión de juego para el partido.

Acerca de los datos de Matchmaker

Los datos del partido incluyen la siguiente información:

- Un identificador de coincidencia único
- El ID de la configuración de emparejamiento que se utilizó para crear la partida
- Los jugadores seleccionados para el partido
- Nombres y asignaciones de los equipos

- Valores de los atributos de los jugadores que se usaron para formar la partida. Los atributos también pueden proporcionar información que indique cómo se configura una sesión de juego. Por ejemplo, el servidor del juego puede asignar personajes a los jugadores en función de sus atributos o elegir una preferencia en el mapa del juego que sea común a todos los jugadores. O bien, es posible que tu juego desbloquee determinadas funciones o niveles en función del nivel medio de habilidad de los jugadores.

Los datos de las partidas no incluyen la latencia de los jugadores. Si necesitas datos de latencia de los jugadores actuales, por ejemplo, para rellenar partidas, te recomendamos que obtengas datos actualizados.

> **Note**
>
> Los datos de Matchmaker especifican el ARN de la configuración de emparejamiento completa, que identifica el nombre de la configuración, la AWS cuenta y la región. En el caso de los juegos que se alojen con Amazon GameLift Servers Match Backfill, solo necesitarás el nombre de la configuración. El nombre de la configuración es la cadena que sigue a «:matchmakingconfiguration/». En el siguiente ejemplo, el nombre de la configuración de emparejamiento es "». MyMatchmakerConfig

Este ejemplo de JSON muestra un conjunto de datos de matchmaker típico. Describe un juego para dos jugadores, en el que los jugadores se emparejan según las calificaciones de habilidad y el nivel más alto alcanzado.

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

Configura un servidor de juegos para FlexMatch

Los servidores de juegos alojados Amazon GameLift Servers deben estar integrados con el SDK Amazon GameLift Servers del servidor y tener las funciones básicas descritas en Añadir Amazon GameLift Servers al servidor de juegos. Esta funcionalidad permite que tu servidor de juegos funcione con recursos de Amazon GameLift Servers alojamiento y se comunique con el Amazon GameLift Servers servicio. En las siguientes instrucciones se describen las tareas adicionales que debe realizar para añadir FlexMatch funciones.

Para añadir FlexMatch a tu servidor de juegos

1. Usa los datos de emparejamiento al iniciar las sesiones de juego. Tu servidor de juegos implementa una función de devolución de llamada llamada. `onStartGameSession()` Tras crear una partida, Amazon GameLift Servers busca un proceso de servidor de juegos disponible y llama a esta función para pedirle que inicie una sesión de juego para la partida. Esta llamada incluye un objeto de sesión de juego (GameSession). Tu servidor de juego utiliza la información de la sesión de juego, incluidos los datos del emparejador, para iniciar la sesión de juego. Para obtener más información sobre cómo iniciar una sesión de juego, consulta Iniciar una sesión de juego. Para obtener más información sobre los datos de Matchmaker, consultaAcerca de los datos de Matchmaker.
2. Administre las conexiones de los jugadores. Cuando se conecte a un juego emparejado, el cliente de juegos hará referencia al ID de jugador y a un ID de sesión del jugador (consulte Validación de un jugador nuevo). Configura tu servidor de juego para usar el ID de jugador para asociar a un jugador entrante con la información del jugador en los datos del emparejador. Los datos de Matchmaker identifican la asignación del equipo de un jugador y otra información que representa al jugador en el juego.

3. Informe cuando los jugadores abandonan un juego. Asegúrate de que el servidor del juego llame al SDK del servidor RemovePlayerSessionpara denunciar la caída de un jugador. Este paso es especialmente importante si utilizas el FlexMatch relleno para llenar los espacios vacíos en juegos existentes. Obtén más información sobre cómo implementar el FlexMatch relleno. Rellena los juegos existentes con FlexMatch
4. Solicita a nuevos jugadores que completen las partidas existentes (opcional). Decide cómo quieres completar tus partidas en directo. Si tu emparejador tiene el modo de relleno configurado como «manual», quizás quieras añadir la función de relleno a tu juego. Si el modo de relleno está configurado como «automático», es posible que necesites una forma de desactivarlo para las sesiones de juego individuales. Por ejemplo, cuando una sesión de juego llegue a un punto determinado del juego, es posible que desees dejar de repostar. Obtén más información sobre cómo implementar el rellenado de partidas. Rellena los juegos existentes con FlexMatch

Rellena los juegos existentes con FlexMatch

La reposición de emparejamiento utiliza los mecanismos de FlexMatch para encontrar nuevos jugadores para sesiones de juego existentes que ya están emparejadas. Aunque siempre puede añadir jugadores a cualquier juego (consulte Conexión de un jugador a una sesión de juego), la reposición de emparejamiento garantiza que los nuevos jugadores cumplan con los mismos criterios de emparejamiento que los jugadores actuales. Además, la reposición de emparejamiento asigna los nuevos jugadores a equipos, administra la aceptación de los jugadores y envía información de emparejamiento actualizada al servidor de juegos. Encontrará más información sobre la reposición de emparejamientos en FlexMatchproceso de emparejamiento.

> **Note**
>
> FlexMatchEl relleno no está disponible actualmente para los juegos que se utilizan. Amazon GameLift Servers Realtime

Existen dos tipos de mecanismos de reposición:

- Activa el relleno automático para completar las sesiones de juego que comiencen con menos jugadores que el máximo permitido. La reposición automática no reemplaza a los jugadores que se unen al juego y luego se retiran.
- Configura un mecanismo de relleno manual para reemplazar a los jugadores que abandonen una sesión de juego en curso. Este mecanismo debe ser capaz de detectar una vacante y generar una solicitud de reposición para llenarla.

Activación de la reposición automática

Con la reposición automática de partidas, Amazon GameLift Servers desencadena automáticamente una solicitud de reposición cada vez que una sesión de juego comienza con una o más ranuras de jugadores sin rellenar. Esta característica permite que los juegos comiencen tan pronto como se encuentre el número mínimo de jugadores emparejados y llenen las ranuras restantes más tarde a medida que se emparejen jugadores adicionales. Puede optar por detener la reposición automática en cualquier momento.

Como ejemplo, consideremos un juego que puede albergar de seis a diez jugadores. FlexMatchinicialmente localiza a seis jugadores, forma el partido e inicia una nueva sesión de

juego. Con la reposición automática, la nueva sesión de juego puede solicitar inmediatamente cuatro jugadores adicionales. Dependiendo del estilo del juego, es posible que deseemos permitir que nuevos jugadores se unan en cualquier momento durante la sesión de juego. De forma alternativa, es posible que deseemos detener la reposición automática después de la fase inicial de configuración y antes de que comience la partida.

Para añadir la reposición automática a su juego, realice las siguientes actualizaciones en él.

1. Habilite la reposición automática. La reposición automática se administra en una configuración de creación de emparejamientos. Cuando está habilitada, se utiliza con todas las sesiones de juego emparejadas que se crean con ese creador de emparejamientos. Amazon GameLift Servers comienza a generar solicitudes de reposición para una sesión que no es de juego completo tan pronto como se inicia la sesión de juego en un servidor de juegos.

   Para activar la reposición automática, abra una configuración de emparejamiento y establezca el modo de reposición en "AUTOMÁTICO". Para obtener más información, consulte Creación de una configuración de emparejamiento.
2. Active la priorización de reposición. Personalice el proceso de emparejamiento para priorizar la tramitación de las solicitudes de reposición antes de crear nuevos emparejamientos. En el conjunto de reglas de emparejamiento, añada un componente de algoritmo y establezca la prioridad de reposición en «alta». Para obtener más información, consulta Personalización del algoritmo de coincidencia.
3. Actualiza la sesión de juego con nuevos datos de matchmaker. Amazon GameLift Serversactualiza tu servidor de juegos con la información de los partidos mediante la función de devolución de llamadas del SDK del servidor `onUpdateGameSession` (consulta Inicializar el proceso del servidor). Añada código al servidor de juegos para administrar los objetos de la sesión de juego actualizados como resultado de la actividad de reposición. Obtenga más información en Actualización de datos de emparejamientos en el servidor de juegos.
4. Desactive la reposición automática para una sesión de juego. Puede optar por detener la reposición automática en cualquier momento durante una sesión de juego individual. Para detener el relleno automático, añade código a tu cliente o servidor de juego para realizar la llamada a la Amazon GameLift Servers API. StopMatchmaking Esta llamada requiere un ID de ticket. Utilice el ID de ticket de reposición de la última solicitud de reposición. Puede obtener esta información a partir de los datos de creación de emparejamientos de sesión de juego, que se actualiza tal y como se describe en el paso anterior.

Genera solicitudes de relleno manual desde un servidor de juegos

Puedes iniciar manualmente las solicitudes de reposición de partidas desde el proceso del servidor del juego que aloja la sesión de juego. El proceso del servidor es el que up-to-date contiene más información sobre los jugadores conectados al juego y el estado de los espacios vacíos.

En este tema se supone que ya ha creado los componentes de FlexMatch necesarios y que ha añadido correctamente procesos de creación de emparejamientos al servidor de juegos y un servicio de juego del lado del cliente. Para obtener más información sobre la configuración de FlexMatch, consulte Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers.

Para habilitar la reposición de emparejamiento para su juego, añada la siguiente funcionalidad:

- Enviar solicitudes de reposición de emparejamiento a un creador de emparejamientos y hacer un seguimiento del estado de las solicitudes.
- Actualice la información de emparejamiento de la sesión de juego. Consulte Actualización de datos de emparejamientos en el servidor de juegos.

Al igual que con otras funcionalidades de servidor, los servidores de juegos utilizan el SDK del servidor de Amazon GameLift Servers. Este SDK está disponible en C++ y C#.

Para realizar solicitudes de reposición de emparejamiento desde su servidor de juegos, complete las siguientes tareas.

1. Dispare una solicitud de reposición de emparejamiento. Por lo general, es buena idea iniciar una solicitud de reposición siempre que un juego emparejado tiene una o varias ranuras de jugadores vacías. Es posible que desee vincular solicitudes de reposición a circunstancias específicas, por ejemplo, para suplir funciones de personaje cruciales o equilibrar equipos. También es probable que desee limitar la actividad de reposición en función de la antigüedad de una sesión de juego.
2. Cree una solicitud de reposición. Añada código para crear y enviar solicitudes de reposición de emparejamiento a un creador de emparejamientos de FlexMatch. Las solicitudes de relleno se gestionan mediante los siguientes servidores APIs:
    - StartMatchBackfill()
    - StopMatchBackfill()

Para crear una solicitud de reposición, llame a `StartMatchBackfill` con la siguiente información. Para cancelar una solicitud de reposición, llame a `StopMatchBackfill` con el ID del ticket de solicitud de reposición.

- ID de ticket: proporcione un ID de ticket del emparejador (o puede optar por que se generen automáticamente). Puedes usar el mismo mecanismo para asignar entradas tanto a las solicitudes de emparejamiento como IDs a las de relleno. Los tickets para el emparejamiento y la reposición se procesan de la misma forma.
- Emparejador: identifique qué emparejador utilizar para la solicitud de reposición. Por lo general, deberá utilizar el mismo creador de emparejamientos que se usó para crear el emparejamiento original. Esta solicitud toma un ARN de configuración de emparejamiento. Esta información se almacena en el objeto de sesión de juego (GameSession), que se proporcionó al proceso del servidor Amazon GameLift Servers al activar la sesión de juego. La el ARN de configuración de emparejamiento se incluye en la propiedad `MatchmakerData`.
- ARN de sesión de juego: identifique la sesión de juego que se va a reponer. Puedes obtener el ARN de la sesión del juego llamando a la API del servidor GetGameSessionId(). Durante el proceso de emparejamiento, los tickets de las nuevas solicitudes no tienen un ID de sesión de juego, mientras que los tickets de las solicitudes de reposición si lo tienen. La presencia del ID de sesión de juego es una forma de saber la diferencia entre los tickets de los nuevos emparejamientos y los tickets de las reposiciones.
- Datos del jugador: incluya información del jugador (Jugador) de todos los jugadores actuales en la sesión de juego que está en fase de reposición. Esta información permite a los creadores de emparejamientos localizar los mejores emparejamientos de jugador para los jugadores que se encuentren en la sesión de juego. Debe incluir la membresía del equipo de cada jugador. No especifique un equipo si no utiliza la reposición. Si el servidor de juegos ha comunicado de forma precisa el estado de conexión del jugador, tendría que poder adquirir estos datos como sigue:

  1. El proceso del servidor que aloja la sesión de juego debe contener la mayor cantidad de up-to-date información sobre los jugadores que están conectados actualmente a la sesión de juego.
  2. Para obtener las asignaciones de los jugadores IDs, los atributos y el equipo, extrae los datos de los jugadores del objeto de la sesión de juego (GameSession), `MatchmakerData` propiedad (consulteAcerca de los datos de Matchmaker). Los datos del creador de

emparejamientos incluyen a todos los jugadores emparejados en la sesión de juego, por lo que tendrá que extraer los datos solo de los jugadores actualmente conectados.

3. Para la latencia de los jugadores, si el creador de emparejamientos solicita datos de latencia, recopile nuevos valores de latencia de todos los jugadores actuales e inclúyalos en cada objeto `Player`. Si se omiten los datos de latencia y el creador de emparejamientos tiene una regla de latencia, la solicitud no se emparejará correctamente. Las solicitudes de reposición exigen datos de latencia solo para la región en la que se encuentra el juego en esos momentos. Puede obtener una región de sesión de juego de la propiedad `GameSessionId` del objeto `GameSession`; este valor es un ARN, que incluye la región.

3. Realiza un seguimiento del estado de una solicitud de relleno. Amazon GameLift Serversactualiza tu servidor de juegos sobre el estado de las solicitudes de reabastecimiento mediante la función de devolución de llamadas del SDK del servidor `onUpdateGameSession` (consulta Cómo inicializar el proceso del servidor). Añada código para gestionar los mensajes estado, así como los objetos de sesión de juego actualizados como resultado de solicitudes de reposición correctas en Actualización de datos de emparejamientos en el servidor de juegos.

   Un creador de emparejamientos solo puede procesar una solicitud de reposición de emparejamiento de una sesión de juego a la vez. Si necesitas cancelar una solicitud, llama StopMatchBackfill a (). Si necesita cambiar una solicitud, llame a `StopMatchBackfill` y envíe una solicitud actualizada.

Genera solicitudes de rellenado manual desde un servicio de back-end

Como alternativa a enviar solicitudes de reposición desde un servidor de juegos, puede enviarlas desde un servicio de juego del lado del cliente. Para utilizar esta opción, el servicio del lado del cliente debe tener acceso a datos actualizados de la actividad de la sesión de juego y las conexiones del jugador; si el juego utiliza un servicio de directorio de sesiones, esto podría ser una buena opción.

En este tema se supone que ya ha creado los componentes de FlexMatch necesarios y que ha añadido correctamente procesos de creación de emparejamientos al servidor de juegos y un servicio de juego del lado del cliente. Para obtener más información sobre la configuración de FlexMatch, consulte Hoja de ruta: Agregue el emparejamiento a una solución de alojamiento Amazon GameLift Servers.

Para habilitar la reposición de emparejamiento para su juego, añada la siguiente funcionalidad:

- Enviar solicitudes de reposición de emparejamiento a un creador de emparejamientos y hacer un seguimiento del estado de las solicitudes.
- Actualice la información de emparejamiento de la sesión de juego. Consulte Actualización de datos de emparejamientos en el servidor de juegos

Al igual que con otras funciones del cliente, un servicio de juegos del lado del cliente utiliza el SDK con la AWS API. Amazon GameLift Servers Este SDK está disponible en C++, C # y en otros lenguajes. Para obtener una descripción general del cliente APIs, consulta la referencia de la Amazon GameLift Servers API, que describe la API del servicio para Amazon GameLift Servers las acciones y enlaza con guías de referencia específicas de cada idioma.

Para configurar un servicio de juego del lado del cliente para reponer juegos emparejados, complete las siguientes tareas.

1. Dispare una solicitud de reposición. Por lo general, un juego inicia una solicitud de reposición siempre que un juego emparejado tiene una o varias ranuras de jugadores vacías. Es posible que desee vincular solicitudes de reposición a circunstancias específicas, por ejemplo, para suplir funciones de personaje cruciales o equilibrar equipos. También es probable que desee limitar las reposiciones en función de la antigüedad de una sesión de juego. Independientemente de lo que utilice como disparador, necesitará la siguiente información como mínimo. Puedes obtener esta información del objeto de sesión de juego (GameSession) llamando DescribeGameSessionscon un identificador de sesión de juego.

   - Número de ranuras de jugadores vacías en la actualidad. Este valor se puede calcular a partir del límite de jugadores máximos de una sesión de juego y el recuento actual de jugadores. El recuento actual de jugadores se actualiza siempre que el servidor de juegos contacta con el servicio Amazon GameLift Servers para validar la conexión de un nuevo jugador o para informar del abandono de un jugador.

   - Política de creación. Esta configuración indica si la sesión de juego acepta nuevos jugadores.

   El objeto de sesión de juego contiene otra información potencialmente útil, incluida la hora de inicio de la sesión de juego, propiedades personalizadas del juego y datos del creador de emparejamientos.

2. Cree una solicitud de reposición. Añada código para crear y enviar solicitudes de reposición de emparejamiento a un creador de emparejamientos de FlexMatch. Las solicitudes de relleno se gestionan mediante los siguientes clientes APIs:

   - StartMatchBackfill
   - StopMatchmaking

   Para crear una solicitud de reposición, llame a `StartMatchBackfill` con la siguiente información. Una solicitud de reposición es parecida a una solicitud de creador de emparejamientos (consulte Solicitud del emparejamiento de jugadores), pero además identifica la sesión de juego existente. Para cancelar una solicitud de reposición, llame a `StopMatchmaking` con el ID del ticket de solicitud de reposición.

   - ID de ticket: proporcione un ID de ticket del emparejador (o puede optar por que se generen automáticamente). Puedes usar el mismo mecanismo para asignar entradas tanto a las solicitudes de emparejamiento como IDs a las de relleno. Los tickets para el emparejamiento y la reposición se procesan de la misma forma.
   - Emparejador: identifique el nombre de la configuración del emparejador que se va a usar. Por lo general, deberá utilizar el mismo creador de emparejamientos para la reposición que el que se usó para crear el emparejamiento original. Esta información se encuentra en un objeto de sesión de juego (GameSession), `MatchmakerData` propiedad, bajo la configuración de emparejamiento ARN. El valor del nombre es la cadena después de ""matchmakingconfiguration/". (Por ejemplo, en el valor de ARN "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4", el nombre de la configuración del creador de emparejamientos es "MM-4v4".)
   - ARN de sesión de juego: especifique la sesión de juego que se va a reponer. Utilice la propiedad `GameSessionId` del objeto de sesión de juego; este ID utiliza el valor de ARN que usted necesita. Los boletos de emparejamiento (MatchmakingTicket) para las solicitudes de reposición tienen el ID de sesión de juego mientras se procesan; los boletos para nuevas solicitudes de emparejamiento no reciben un ID de sesión de juego hasta que se coloca el partido; la presencia de un ID de sesión de juego es una forma de diferenciar entre las entradas para nuevos partidos y las entradas para rellenarse.
   - Datos del jugador: incluya información del jugador (Jugador) de todos los jugadores actuales en la sesión de juego que está en fase de reposición. Esta información permite al creador de emparejamientos localizar los mejores emparejamientos de jugador para los jugadores que se encuentren en la sesión de juego. Debe incluir la membresía del equipo de cada jugador.

No especifique un equipo si no utiliza la reposición. Si el servidor de juegos ha comunicado de forma precisa el estado de conexión del jugador, tendría que poder adquirir estos datos como sigue:

1. Llama al DescribePlayerSessions() con el identificador de la sesión de juego para descubrir todos los jugadores que están conectados actualmente a la sesión de juego. Cada sesión de jugador incluye un ID de jugador. Puede añadir un filtro de estado para recuperar solo las sesiones de jugador activas.
2. Extrae los datos del jugador del objeto de la sesión de juego (GameSession), de la `MatchmakerData` propiedad (consulteAcerca de los datos de Matchmaker. Usa el jugador IDs adquirido en el paso anterior para obtener datos únicamente para los jugadores actualmente conectados. Como los datos del creador de emparejamientos no se actualizan cuando los jugadores abandonan el juego, tendrá que extraer los datos solo para los jugadores actuales.
3. Para la latencia de los jugadores, si el creador de emparejamientos solicita datos de latencia, recopile nuevos valores de latencia de todos los jugadores actuales e inclúyalos en el objeto `Player`. Si se omiten los datos de latencia y el creador de emparejamientos tiene una regla de latencia, la solicitud no se emparejará correctamente. Las solicitudes de reposición exigen datos de latencia solo para la región en la que se encuentra el juego en esos momentos. Puede obtener una región de sesión de juego de la propiedad `GameSessionId` del objeto `GameSession`; este valor es un ARN, que incluye la región.

3. Haga un seguimiento del estado de una solicitud de reposición. Añada código para permanecer a la escucha de actualizaciones del estado del ticket de emparejamiento. Puede utilizar la configuración del mecanismo para realizar un seguimiento de los tickets de las nuevas solicitudes de emparejamiento (consulte Seguimiento de eventos de emparejamiento) utilizando notificación (preferido) o sondeo de eventos. Aunque no necesita disparar la actividad de aceptación del jugador con solicitudes de reposición y la información del jugador se actualiza en el servidor de juegos, aún tiene que vigilar el estado del ticket para gestionar errores y repeticiones de envío de solicitudes.

   Un creador de emparejamientos solo puede procesar una solicitud de reposición de emparejamiento de una sesión de juego a la vez. Si necesita cancelar una solicitud, llame a StopMatchmaking. Si necesita cambiar una solicitud, llame a `StopMatchmaking` y envíe una solicitud actualizada.

   Una vez que una solicitud de reposición de emparejamiento tiene éxito, su servidor de juegos recibe un objeto `GameSession` actualizado y gestiona las tareas necesarias para

incorporar nuevos jugadores a la sesión de juego. Más información en Actualización de datos de emparejamientos en el servidor de juegos.

Actualización de datos de emparejamientos en el servidor de juegos

Independientemente de cómo inicie las solicitudes de reposición de emparejamiento en su juego, el servidor de juegos tiene que poder gestionar las actualizaciones de sesión de juego que Amazon GameLift Servers proporciona como resultado de solicitudes de reposición de emparejamiento.

Cuando Amazon GameLift Servers completa una solicitud de reposición de partidas (con éxito o no), llama al servidor del juego mediante la función de devolución de llamadas. on`UpdateGameSession` Esta llamada tiene tres parámetros de entrada: el identificador de un boleto para completar el partido, un mensaje de estado y un GameSession objeto que contiene la mayoría de los datos de emparejamiento, incluida la información del jugador. up-to-date Tiene que añadir el siguiente código a su servidor de juegos como parte de la integración del servidor de juegos:

1. Implemente la función on`UpdateGameSession`. Esta función tiene que poder gestionar los siguientes mensajes de estado (`updateReason`):
   - MATCHMAKING_DATA_UPDATED: se han emparejado correctamente nuevos jugadores a la sesión de juego. El objeto `GameSession` contiene datos actualizados del creador de emparejamientos, incluidos datos de jugador de los jugadores existentes y los jugadores recién emparejados.
   - BACKFILL_FAILED: el intento de reponer el emparejamiento falló debido a un error interno. El objeto `GameSession` no ha cambiado.
   - BACKFILL_TIMED_OUT: el emparejador no pudo encontrar una emparejamiento de reposición dentro del límite de tiempo. El objeto `GameSession` no ha cambiado.
   - BACKFILL_CANCELLED: la solicitud de reabastecimiento de partidas se canceló mediante una llamada a StopMatchmaking (cliente) o (servidor). StopMatchBackfill El objeto `GameSession` no ha cambiado.
2. Para emparejamientos de reposición correctos, utilice los datos actualizados del creador de emparejamientos para gestionar los nuevos jugadores cuando se conecten a la sesión de juego. Como mínimo, deberá utilizar las asignaciones de equipo para el o los nuevos jugadores, así como otros atributos del jugador necesarios para que el jugador inicie el juego.

3. En la acción de llamada de tu servidor de juegos al SDK del servidor ProcessReady(), añade el nombre del método de devolución de `onUpdateGameSession` llamada como parámetro del proceso.

Seguridad con FlexMatch

La seguridad en la nube AWS es la máxima prioridad. Como cliente de AWS , se beneficiará de una arquitectura de red y de centros de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWS programas de de . Para obtener más información sobre los programas de cumplimiento aplicablesAmazon GameLift Servers, consulte los AWS servicios incluidos en el ámbito de aplicación por programa de cumplimiento y .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También eres responsable de otros factores, como la confidencialidad de tus datos, los requisitos de tu empresa y las leyes AWS y reglamentos aplicables.

Para obtener información de seguridad relacionada conAmazon GameLift Servers, inclusoFlexMatch, consulte Seguridad en Amazon GameLift Servers. Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida a la hora de utilizar Amazon GameLift Servers. En los temas se muestra cómo realizar la configuración Amazon GameLift Servers para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus Amazon GameLift Servers recursos.

Amazon GameLift ServersreferenciaFlexMatch

Esta sección contiene la documentación de referencia del emparejamiento de FlexMatch de Amazon GameLift Servers.

Temas

Amazon GameLift ServersFlexMatchReferencia de API (AWS SDK)

En este tema se proporciona una lista basada en tareas de las operaciones de la API para. Amazon GameLift Servers FlexMatch La API del Amazon GameLift Servers FlexMatch servicio está empaquetada en el AWS SDK, en el espacio de nombres. `aws.gamelift` Descarga el AWS SDK o consulta la documentación de referencia de la Amazon GameLift Servers API.

Amazon GameLift ServersFlexMatchproporciona servicios de emparejamiento para su uso con juegos que están alojados en soluciones de Amazon GameLift Servers alojamiento (incluido el alojamiento gestionado para servidores de juegos personalizados o Amazon GameLift Servers Realtime el alojamiento en Amazon EC2 con Amazon GameLift ServersFleetIQ), así como con otros sistemas de alojamiento peer-to-peer, como primitivos de computación en la nube o locales. Consulta la Guía para Amazon GameLift Servers desarrolladores para obtener más información sobre otras Amazon GameLift Servers opciones de alojamiento.

Temas

Configuración de reglas y procesos de emparejamiento

Utiliza estas operaciones para crear un FlexMatch emparejador, configurar el proceso de emparejamiento de tu juego y definir un conjunto de reglas personalizadas para crear partidos y equipos.

Configuración de la creación de emparejamientos

- CreateMatchmakingConfiguration— Crea una configuración de emparejamiento con instrucciones para evaluar grupos de jugadores y crear equipos de jugadores. Cuando la Amazon GameLift Servers utilices como anfitrión, especifica también cómo crear una nueva sesión de juego para el partido.
- DescribeMatchmakingConfigurations— Recupera las configuraciones de emparejamiento definidas en una Amazon GameLift Servers región.
- UpdateMatchmakingConfiguration— Cambiar los ajustes de la configuración de emparejamiento. cola.
- DeleteMatchmakingConfiguration— Eliminar una configuración de emparejamiento de la región.

Conjunto de reglas de creación de emparejamientos

- CreateMatchmakingRuleSet— Crea un conjunto de reglas para usarlas cuando busques partidas de jugadores.
- DescribeMatchmakingRuleSets— Recupera los conjuntos de reglas de emparejamiento definidos en una Amazon GameLift Servers región.
- ValidateMatchmakingRuleSet— Verificar la sintaxis de un conjunto de reglas de emparejamiento.
- DeleteMatchmakingRuleSet— Eliminar un conjunto de reglas de emparejamiento de la región.

Solicitud de un emparejamiento para uno o varios jugadores

Llame a estas operaciones desde su servicio cliente de juego para administrar las solicitudes de emparejamiento de jugadores.

- StartMatchmaking— Solicita el emparejamiento para un jugador o un grupo que quiera jugar la misma partida.
- DescribeMatchmaking— Obtén detalles sobre una solicitud de emparejamiento, incluido el estado.
- AcceptMatch— En el caso de un partido que requiera la aceptación de un jugador, notifícalo Amazon GameLift Servers cuando un jugador acepte un partido propuesto.
- StopMatchmaking— Cancelar una solicitud de emparejamiento.
- StartMatchBackfill- Solicita partidas de jugadores adicionales para llenar los espacios vacíos en una sesión de juego existente.

Lenguajes de programación disponibles

El AWS SDK compatible con Amazon GameLift Servers está disponible en los siguientes idiomas. Para obtener información sobre la compatibilidad con los entornos de desarrollo, consulte la documentación de cada idioma.

- C++ (documentación del SDK) (Amazon GameLift Servers)
- Java (documentos del SDK) (Amazon GameLift Servers)
- .NET (documentos del SDK) (Amazon GameLift Servers)
- Go (documentación del SDK) (Amazon GameLift Servers)
- Python (documentación del SDK) (Amazon GameLift Servers)
- Ruby (documentos del SDK) (Amazon GameLift Servers)
- PHP (documentación del SDK) (Amazon GameLift Servers)
- JavaScript/Node.js (documentación del SDK) (Amazon GameLift Servers)

FlexMatch lenguaje de reglas

Los temas de referencia de esta sección describen la sintaxis y la semántica que se utilizan para crear reglas de emparejamiento para usarlas con Amazon GameLift Servers FlexMatch. Para obtener ayuda detallada para redactar reglas y conjuntos de reglas de emparejamiento, consulteConstruye un FlexMatch conjunto de reglas.

Temas

- FlexMatch esquema de conjunto de reglas
- FlexMatch definiciones de propiedades de conjuntos de reglas
- FlexMatchtipos de reglas
- FlexMatch expresiones de propiedades

FlexMatch esquema de conjunto de reglas

FlexMatch los conjuntos de reglas utilizan un esquema estándar para las reglas de coincidencia pequeña y grande. Para obtener descripciones detalladas de cada sección, consulte FlexMatch definiciones de propiedades de conjuntos de reglas.

Esquema de conjuntos de reglas para emparejamientos reducidos

El siguiente esquema documenta todas las propiedades posibles y los valores permitidos de un conjunto de reglas que se usa para crear emparejamientos de hasta 40 jugadores.

```
{
    "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 conjuntos de reglas para emparejamientos de gran tamaño

El siguiente esquema documenta todas las propiedades posibles y los valores permitidos de un conjunto de reglas que se usa para crear emparejamientos superiores a 40 jugadores. Si el total de `maxPlayers` valores de todos los equipos del conjunto de reglas supera los 40, entonces FlexMatch procesa las solicitudes de coincidencia que utilizan esta regla establecida según las pautas de partidos 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
        }]
    }]
}
```

FlexMatch definiciones de propiedades de conjuntos de reglas

En esta sección se define cada propiedad del esquema del conjunto de reglas. Para obtener ayuda adicional sobre la creación de un conjunto de reglas, consulte Construye un FlexMatch conjunto de reglas.

`name`

Una etiqueta descriptiva para el conjunto de reglas. Este valor no está asociado al nombre asignado al Amazon GameLift Servers MatchmakingRuleSet recurso. Este valor se incluye en los datos de emparejamiento que describen una partida completada, pero no lo utiliza ninguna Amazon GameLift Servers procesos.

Valores permitidos: string

¿Obligatorio? No

`ruleLanguageVersion`

La versión del lenguaje de expresión de FlexMatch propiedades que se utiliza.

Valores permitidos: «1.0»

¿Obligatorio? Sí

playerAttributes

Una recopilación de datos de los jugadores que se incluye en las solicitudes de emparejamiento y que se utiliza en el proceso de emparejamiento. También puede declarar los atributos aquí para incluir los datos de los jugadores en los datos de emparejamiento que se transmiten a los servidores de juegos, incluso si los datos no se utilizan en el proceso de emparejamiento.

¿Obligatorio? No

name

Un nombre único para el atributo del jugador que utilizará el emparejador. Este nombre debe coincidir con el nombre del atributo del jugador al que se hace referencia en las solicitudes de emparejamiento.

Valores permitidos: string

¿Obligatorio? Sí

type

El tipo de datos del valor de atributo del jugador.

Valores permitidos: «string», «number», «string_list» y «string_number_map»

¿Obligatorio? Sí

default

Un valor predeterminado que se va a utilizar cuando una solicitud de emparejamiento no proporciona uno para un jugador.

Valores permitidos: cualquier valor permitido para el atributo del jugador.

¿Obligatorio? No

algorithm

Ajustes de configuración opcionales para personalizar el proceso de emparejamiento.

¿Obligatorio? No

strategy

El método que se debe utilizar al crear emparejamientos. Si no se establece esta propiedad, el comportamiento predeterminado es «exhaustiveSearch».

Valores permitidos:

- «exhaustiveSearch»: método de búsqueda estándar. FlexMatch forma una combinación en torno al boleto más antiguo de un lote evaluando los otros boletos del grupo según un conjunto de reglas de juego personalizadas. Esta estrategia se utiliza para partidos de 40 jugadores o menos. Cuando utilice esta estrategia, `batchingPreference` debe configurarla como «aleatoria» u «ordenada».
- «equilibrado»: método que está optimizado para formar emparejamientos grandes de forma rápida. Esta estrategia se utiliza para partidos de 41 jugadores a 200 jugadores. Forma los emparejamientos clasificando previamente el grupo de tickets, compilando posibles emparejamientos y asignando jugadores a los equipos. Después, equilibra a cada equipo de un emparejamiento utilizando un atributo de jugador específico. Por ejemplo, esta estrategia se puede utilizar para igualar los niveles medios de habilidad de todos los equipos en un emparejamiento. Cuando utilice esta estrategia, `balancedAttribute` debe configurarse y `batchingPreference` debe establecerse como «largestPopulation» o «fastestRegion». La mayoría de los tipos de reglas personalizadas no se reconocen con esta estrategia.

¿Obligatorio? Sí

`batchingPreference`

El método de clasificación previa que se debe utilizar antes de agrupar los tickets para compilar emparejamientos. Clasificar previamente el grupo de tickets provoca que los tickets se agrupen por lotes en función de una característica específica, lo que tiende a aumentar la uniformidad entre los jugadores en los emparejamientos finales.

Valores permitidos:

- «aleatorio»: válido solo con `strategy` = «exhaustiveSearch». No se realiza una clasificación previa; los tickets del grupo se agrupan aleatoriamente. Este es el comportamiento predeterminado para una estrategia de búsqueda exhaustiva.
- «ordenado»: válido solo con `strategy` = «exhaustiveSearch». El grupo de tickets se clasifica previamente en función de los atributos de los jugadores que aparecen en `sortbyAttributes`.
- «largestPopulation»: válido solo con `strategy` = «equilibrado». El grupo de tickets se clasifica previamente mediante regiones en las que los jugadores muestran niveles de latencia aceptables. Este es el comportamiento predeterminado para una estrategia equilibrada.

- «fastestRegion»: válido solo con `strategy` = «equilibrado». El grupo de tickets se clasifica previamente mediante regiones en las que los jugadores muestran los niveles de latencia más bajos. Los emparejamientos resultantes tardan más en completarse, pero la latencia para todos los jugadores suele ser baja.

¿Obligatorio? Sí

`balancedAttribute`

El nombre del atributo de un jugador que se utilizará al compilar emparejamientos grandes con una estrategia equilibrada.

Valores permitidos: cualquier atributo declarado en `playerAttributes` con `type` = «number».

¿Obligatorio? Sí, si `strategy` = «equilibrado».

`sortByAttributes`

Una lista de los atributos de los jugadores para utilizarla al clasificar previamente el grupo de tickets antes de la agrupación por lotes. Esta propiedad solo se utiliza cuando se realiza una clasificación previa con la estrategia de búsqueda exhaustiva. El orden de la lista de atributos determina el orden de clasificación. FlexMatch utiliza una convención de clasificación estándar para los valores alfanuméricos y alfanuméricos.

Valores permitidos: cualquier atributo declarado en `playerAttributes`.

¿Obligatorio? Sí, si `batchingPreference` = «ordenado».

`backfillPriority`

El método de priorización para emparejar los tickets de reposición. Esta propiedad determina cuándo FlexMatch procesa los tickets de relleno en un lote. Solo se utiliza cuando se realiza una clasificación previa con la estrategia de búsqueda exhaustiva. Si no se establece esta propiedad, el comportamiento predeterminado es «normal».

Valores permitidos:

- «normal»: el tipo de solicitud del ticket (reposición o nuevo emparejamiento) no se tiene en cuenta a la hora de formar emparejamientos.
- «alto»: un lote de entradas se clasifica por tipo de solicitud (y luego por antigüedad) y FlexMatch intenta hacer coincidir primero las entradas rellenadas.

- «bajo»: un lote de entradas se clasifica por tipo de solicitud (y luego por antigüedad) y FlexMatch intenta encontrar primero las entradas no rellenadas.

¿Obligatorio? No

`expansionAgeSelection`

El método para calcular el tiempo de espera para la expansión de las reglas de un emparejamiento. Las expansiones se utilizan para flexibilizar los requisitos de un emparejamiento si este no se ha completado después de un cierto periodo. El tiempo de espera se calcula en función de la antigüedad de los tickets que ya están disponibles para el emparejamiento parcialmente lleno. Si no se establece esta propiedad, el comportamiento predeterminado es «más reciente».

Valores permitidos:

- «más reciente»: el tiempo de espera de la expansión se calcula en función del ticket con la marca de tiempo de creación más reciente en el emparejamiento que se haya completado parcialmente. Las expansiones suelen activarse más lentamente, ya que un ticket más reciente puede reactivar el tiempo de espera.
- «más antiguo»: el tiempo de espera de la expansión se calcula en función del ticket con la marca de tiempo de creación más antigua en el emparejamiento. Las expansiones suelen activarse más rápido.

¿Obligatorio? No

`teams`

La configuración de los equipos en un emparejamiento. Proporcione un nombre de equipo y un rango de tamaños para cada equipo. Un conjunto de reglas debe definir al menos un equipo.

`name`

Un nombre único para el equipo. Se puede hacer referencia a los nombres de los equipos en las reglas y las expansiones. Si un emparejamiento es correcto, los jugadores se asignan por nombre de equipo en los datos de emparejamiento.

Valores permitidos: string

¿Obligatorio? Sí

`maxPlayers`

El número máximo de jugadores que se puede asignar al equipo.

Valores permitidos: number

¿Obligatorio? Sí

`minPlayers`

El número mínimo de jugadores que deben asignarse al equipo antes de que el emparejamiento sea viable.

Valores permitidos: number

¿Obligatorio? Sí

`quantity`

El número de equipos de este tipo que se va a crear en un emparejamiento. Los equipos con cantidades superiores a 1 se designan con un número adjunto («Red_1», «Red_2», etc.). Si no se define esta propiedad, el valor predeterminado es «1».

Valores permitidos: number

¿Obligatorio? No

`rules`

Un conjunto de declaraciones de reglas que definan cómo evaluar a los jugadores para un emparejamiento.

¿Obligatorio? No

`name`

Un nombre único para la regla. Todas las reglas de un conjunto de reglas deben tener nombres únicos. Se hace referencia a los nombres de la reglas en registros de eventos y métricas que realizan un seguimiento de la actividad relacionada con la regla.

Valores permitidos: string

¿Obligatorio? Sí

`description`

Una descripción textual de la regla. Esta información se puede utilizar para identificar el propósito de una regla. No se utiliza en el proceso de emparejamiento.

Valores permitidos: string

¿Obligatorio? No

`type`

El tipo de declaración de reglas. Cada tipo de regla tiene propiedades adicionales que se deben establecer. Para obtener más información sobre la estructura y el uso de cada tipo de regla, consulte FlexMatchtipos de reglas.

Valores permitidos:

- «absoluteSort»: permite ordenar mediante un método de clasificación explícito que ordena los tickets en un lote en función de si un atributo de jugador específico se compara con el ticket más antiguo del lote.
- «colección»: permite evaluar los valores de una colección, como un atributo de un jugador que es una colección o un conjunto de valores para varios jugadores.
- «comparación»: permite comparar dos valores.
- “compound”: permite definir una regla de emparejamiento compuesta mediante una combinación lógica de otras reglas del conjunto de reglas. Solo se admite en emparejamientos de 40 jugadores o menos.
- “distance”: permite medir la distancia entre valores numéricos.
- “batchDistance”: permite medir la diferencia entre el valor de un atributo y lo utiliza para agrupar las solicitudes de coincidencia.
- «distanceSort»: permite ordenar mediante un método de clasificación explícito que ordena los tickets en un lote en función de la forma en la que un atributo de jugador específico con un valor numérico se compara con el ticket más antiguo del lote.
- “latency”: permite evaluar los datos de latencia regional que se muestran en una solicitud de emparejamiento.

¿Obligatorio? Sí

`expansions`

Reglas para flexibilizar los requisitos de los emparejamientos a lo largo del tiempo cuando no se pueda completar un emparejamiento. Configure las expansiones como una serie de pasos que se apliquen gradualmente para facilitar la búsqueda de emparejamientos. Por defecto, FlexMatch calcula el tiempo de espera en función de la antigüedad de la última entrada añadida a un partido.

Puede cambiar la forma en que se calculan los tiempos de espera de expansión mediante la propiedad del algoritmo `expansionAgeSelection`.

Los tiempos de espera de expansión son valores absolutos, por lo que cada paso debe tener un tiempo de espera superior al paso anterior. Por ejemplo, para programar una serie de expansión gradual, debe utilizar tiempos de espera de 30, 40 y 50 segundos. Los tiempos de espera no pueden superar el tiempo máximo permitido para una solicitud de emparejamiento, que se establece en la configuración de emparejamiento.

¿Obligatorio? No

`target`

El elemento del conjunto de reglas debe flexibilizarse. Puede flexibilizar las propiedades del tamaño de un equipo o cualquier propiedad de declaración de las reglas. La sintaxis es "<nombre componente>[<regla/nombre equipo>]. <nombre propiedad>". Por ejemplo, para cambiar el tamaño mínimo de los equipos:`teams[Red, Yellow].minPlayers`. Para cambiar el requisito de habilidad mínima en una declaración de reglas de comparación denominada «minSkill»:`rules[minSkill].referenceValue`.

¿Obligatorio? Sí

`steps`

`waitTimeSeconds`

Es la duración, en segundos, que debe esperar antes de aplicar el nuevo valor al elemento del conjunto de reglas de destino.

¿Obligatorio? Sí

`value`

El nuevo valor del elemento del conjunto de reglas de destino.

FlexMatchtipos de reglas

Regla de distancia por lotes

```
batchDistance
```

Las reglas de distancia por lotes miden la diferencia entre dos valores de atributos. Puede utilizar el tipo de regla de distancia por lotes con emparejamientos grandes y pequeños. Existen dos tipos de reglas de distancia de lotes:

- Compare los valores de los atributos numéricos. Por ejemplo, una regla de distancia por lotes de este tipo podría requerir que todos los jugadores de un emparejamiento estén situados en dos niveles de distancia entre sí como máximo. Para este tipo, defina una distancia máxima entre `batchAttribute` y todos los tickets.
- Compare los valores de los atributos de cadena. Por ejemplo, una regla de distancia por lotes de este tipo puede requerir que todos los jugadores de un emparejamiento soliciten el mismo modo de juego. Para este tipo, defina un `batchAttribute` valor que se FlexMatch utilice para formar lotes.

Propiedades de regla de distancia por lotes

- **`batchAttribute`**: el valor del atributo del jugador utilizado para formar lotes.
- **`maxDistance`**: el valor de distancia máxima para un emparejamiento correcto. Se utiliza para comparar atributos numéricos.
- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen los valores mínimo (`min`), máximo (`max`) y medio (`avg`) para los jugadores de un ticket. El valor predeterminado es `avg`.

Example

Ejemplos

```
{
  "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"
}
```

Regla de comparación

```
comparison
```

Las reglas de comparación comparan un valor de atributo de un jugador con otro valor. Existen dos tipos de reglas de comparación:

- Comparar con el valor de referencia. Por ejemplo, una regla de comparación de este tipo podría requerir que los jugadores emparejados tengan un nivel de habilidad determinado o superior. Para este tipo, especifique un atributo del jugador, un valor de referencia y una operación de comparación.
- Comparar entre jugadores. Por ejemplo, una regla de comparación de este tipo puede requerir que todos los jugadores del emparejamiento utilicen caracteres diferentes. Para este tipo, especifique el atributo de un jugador y la operación de comparación igual (=) o no igual (`!=`). No especifique un valor de referencia.

> **Note**
>
> Las reglas de distancia por lotes son más eficientes para comparar los atributos de los jugadores. Para reducir la latencia de los emparejamientos, utilice una regla de distancia por lotes siempre que sea posible.

Propiedades de regla de comparación

- **`measurements`**: valor del atributo del jugador que se va a comparar.
- **`referenceValue`**: valor con el que comparar la medición para un posible emparejamiento.
- **`operation`**: valor que determina cómo comparar la medición con el valor de referencia. Entre las operaciones válidas se incluyen: <, <=, =, `!=`, > y >=.
- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen los valores mínimo (`min`), máximo (`max`) y medio (`avg`) para los jugadores de un ticket. El valor predeterminado es `avg`.

Regla de distancia

```
distance
```

Las reglas de distancia miden la diferencia entre dos valores numéricos, como, por ejemplo, la distancia entre niveles de habilidad del jugador. Por ejemplo, una regla de distancia puede requerir que todos los jugadores hayan jugado al juego durante al menos 30 horas.

> **Note**
>
> Las reglas de distancia por lotes son más eficientes para comparar los atributos de los jugadores. Para reducir la latencia de los emparejamientos, utilice una regla de distancia por lotes siempre que sea posible.

Propiedades de regla de distancia

- **`measurements`**: valor del atributo del jugador para medir la distancia. Debe ser un atributo con un valor numérico.
- **`referenceValue`**: valor numérico con el que se mide la distancia para un posible emparejamiento.
- **`minDistance`/`maxDistance`**: el valor de distancia mínima o máxima para un emparejamiento correcto.
- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen los valores mínimo (`min`), máximo (`max`) y medio (`avg`) para los jugadores de un ticket. El valor predeterminado es `avg`.

Regla de colección

```
collection
```

Las reglas de colección comparan un grupo de valores de atributos de un jugador con los de otros jugadores del lote o con un valor de referencia. Una colección puede contener valores de atributos de varios jugadores, un atributo de un jugador como una lista de cadenas, o ambos. Por ejemplo, una regla de colección podría incluir los caracteres que elijan los jugadores de un equipo. En ese caso, la regla podría requerir que el equipo tenga al menos un carácter determinado.

Propiedades de regla de recopilación

- **`measurements`**: colección de valores del atributo del jugador que se va a comparar. Los valores de los atributos deben se listas de cadenas.
- **`referenceValue`**: valor (o colección de valores) que se utilizará para comparar las medidas de un posible emparejamiento.
- **`operation`**: el valor que determina cómo comparar una colección de medidas. Las operaciones válidas incluyen lo siguiente:
  - `intersection`: esta operación mide el número de valores que son iguales en las colecciones de todos los jugadores. Para ver un ejemplo de una regla que utiliza la operación de intersección, consulte Ejemplo: usa una clasificación explícita para encontrar las mejores coincidencias.
  - `contains`: esta operación mide el número de colecciones de atributos de los jugadores que contienen el valor de referencia especificado. Para ver un ejemplo de una regla que utiliza la operación de contenido, consulte Ejemplo: establece los requisitos y los límites de latencia a nivel de equipo.
  - `reference_intersection_count`: esta operación mide el número de elementos en una colección de atributos de los jugadores que emparejan los elementos en la colección de valores de referencia. Puede utilizar esta operación para comparar varios atributos de jugadores diferentes. Para ver un ejemplo de una regla que compara colecciones de atributos de varios jugadores, consulte Ejemplo: encuentra intersecciones entre los atributos de varios jugadores.
- **`minCount`**/**`maxCount`**: el valor de recuento mínimo o máximo para un emparejamiento correcto.
- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Para obtener este valor, puede utilizar `union` para combinar los atributos de jugador para todos los jugadores del grupo. También puede utilizar `intersection` para usar los atributos de los jugadores que el grupo tenga en común. El valor predeterminado es `union`.

Regla compuesta

```
compound
```

Las reglas compuestas utilizan enunciados lógicos para formar emparejamientos de 40 o menos jugadores. Puede utilizar varias reglas compuestas en un solo conjunto de reglas. Cuando se utilizan varias reglas compuestas, todas deben ser verdaderas para formar un emparejamiento.

No puede expandir una regla compuesta mediante reglas de expansión, pero puede ampliar las reglas subyacentes o de apoyo.

Propiedades de una regla compuesta

- **`statement`**: la lógica utilizada para combinar reglas individuales para formar la regla compuesta. Las reglas que especifique en esta propiedad deben haberse definido anteriormente en el conjunto de reglas. No puede utilizar reglas de `batchDistance` en una regla compuesta.

  Esta propiedad admite los siguientes operadores lógicos:

  - `and`: la expresión es verdadera si los dos argumentos proporcionados son verdaderos.
  - `or`: la expresión es verdadera si los dos argumentos proporcionados son verdaderos.
  - `not`: invierte el resultado del argumento de la expresión.
  - `xor`: la expresión es verdadera si solo uno de los argumentos es verdadero.

Example Ejemplo

El siguiente ejemplo empareja a jugadores de diferentes niveles de habilidad según el modo de juego que seleccionen.

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

Latency rule

```
latency
```

Las reglas de latencia miden la latencia de los jugadores por ubicación. Una regla de latencia ignora cualquier ubicación con una latencia superior a la máxima. Un jugador debe tener un valor de latencia inferior al máximo en al menos una ubicación para que la regla de latencia lo acepte. Puede utilizar este tipo de regla con coincidencias grandes especificando la propiedad `maxLatency`.

Propiedades de regla de latencia

- **`maxLatency`**: valor de latencia máximo aceptable para una ubicación. Si un ticket no tiene ubicaciones con una latencia inferior al máximo, no cumple la regla de latencia.
- **`maxDistance`**: valor máximo entre la latencia de cada billete y el valor de referencia de la distancia.
- **`distanceReference`**: valor de latencia con el que comparar la latencia de los tickets. Los tickets que se encuentren dentro de la distancia máxima del valor de referencia de la distancia dan como resultado un emparejamiento correcto. Las opciones válidas son los valores mínimo (`min`) y medio (`avg`) de latencia de los jugadores.
- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen los valores mínimo (`min`), máximo (`max`) y medio (`avg`) para los jugadores de un ticket. El valor predeterminado es `avg`.

> **Note**
>
> Una cola puede ubicar una sesión de juego en una región que no cumpla una regla de latencia. Para obtener más información sobre las políticas de latencia de las colas, consulte Creación de una política de latencia de jugadores.

Regla de ordenación absoluta

```
absoluteSort
```

Las reglas de ordenación absoluta clasifican un lote de tickets de emparejamiento en función de un atributo de jugador específico en comparación con el primer ticket que se añade al lote.

Propiedades de regla de ordenación absoluta

- **`sortDirection`**: orden en el que se ordenan los tickets de emparejamiento. Entre las opciones válidas se incluyen `ascending` y `descending`.
- **`sortAttribute`**: atributo del jugador por el que se ordenan los tickets.
- **`mapKey`**: opciones para ordenar el atributo del jugador si se trata de un mapa. Entre las opciones válidas se incluyen:
  - `minValue`: la clave con el valor más bajo es la primera.
  - `maxValue`: la clave con el valor más alto es la primera.

- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen el atributo de jugador mínimo (`min`), el atributo de jugador máximo (`max`) y la media (`avg`) de todos los atributos de jugador de los jugadores del grupo. El valor predeterminado es `avg`.

Example

Ejemplo

La siguiente regla de ejemplo ordena a los jugadores por nivel de habilidad y hace una media del nivel de habilidad de los grupos.

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

Regla de ordenación por distancia

```
distanceSort
```

Las reglas de ordenación por distancia clasifican un lote de tickets de emparejamiento en función de la distancia de un atributo de jugador específico con el primer ticket que se añade al lote.

Propiedades de regla de ordenación por distancia

- **`sortDirection`**: forma en que se ordenan los tickets de emparejamiento. Entre las opciones válidas se incluyen `ascending` y `descending`.
- **`sortAttribute`**: atributo del jugador por el que se ordenan los tickets.
- **`mapKey`**: opciones para ordenar el atributo del jugador si se trata de un mapa. Entre las opciones válidas se incluyen:
  - `minValue`: para el primer ticket que se añada al lote, busque la clave con el valor más bajo.
  - `maxValue`: para el primer ticket que se añada al lote, busque la clave con el valor más alto.

- **`partyAggregation`**— El valor que determina cómo se FlexMatch gestionan las entradas con varios jugadores (grupos). Las opciones válidas incluyen los valores mínimo (`min`), máximo (`max`) y medio (`avg`) para los jugadores de un ticket. El valor predeterminado es `avg`.

FlexMatch expresiones de propiedades

Las expresiones de propiedades se pueden utilizar para referirse a determinadas propiedades relacionadas con el emparejamiento. Le permiten utilizar los cálculos y la lógica al definir el valor de una propiedad. Las expresiones de propiedad suelen adoptar una de estas dos formas:

- Datos de los jugadores individuales.
- Recopilaciones calculadas de datos de jugadores individuales.

Expresiones comunes de propiedades de emparejamiento

Una expresión de propiedad identifica un valor específico para un jugador, equipo o emparejamiento. Las siguientes expresiones parciales ilustran cómo identificar equipos y jugadores:

Objetivo	Input	Significado	Output
Para identificar un equipo específico en un emparejamiento:	`teams[red]`	El equipo rojo	Equipo
Para identificar un conjunto de equipos específicos en un emparejamiento:	`teams[red,blue]`	El equipo rojo y el equipo azul	List<Team>
Para identificar todos los equipos en un emparejamiento:	`teams[*]`	Todos los equipos	List<Team>
Para identificar jugadores en un equipo específico:	`team[red] .players`	Jugadores en el equipo rojo	List<Player>

Objetivo	Input	Significado	Output
Para identificar jugadores en un conjunto de equipos específicos en un emparejamiento, realice el siguiente procedimiento:	`team[red, blue].players`	Jugadores en el emparejamiento, agrupados por equipo	List<List<Player>>
Para identificar jugadores en un emparejamiento:	`team[*].players`	Jugadores en el emparejamiento, agrupados por equipo	List<List<Player>>

Ejemplos de expresiones de propiedades

En la tabla siguiente se muestran algunas expresiones de propiedades que se basan en los ejemplos anteriores:

Expression	Significado	Tipo resultante
`teams[red].players[playerId]`	El jugador IDs de todos los jugadores del equipo rojo	List<string>
`teams[red].players.attributes[skill]`	Los atributos de "habilidad" de todos los jugadores del equipo rojo	List<number>
`teams[red,blue].players.attributes[skill]`	Los atributos de «habilidad» de todos los jugadores del equipo rojo y del equipo azul agrupados por equipo.	List<List<number>>
`teams[*].players.attributes[skill]`	Los atributos de "habilidad" de todos los jugadores del emparejamiento, agrupados por equipo	List<List<number>>

Agregaciones de expresiones de propiedades

Pueden utilizarse expresiones de propiedades para agregar datos del equipo utilizando las siguientes funciones o combinaciones de funciones:

Agregación	Input	Significado	Output
`min`	List<number>	Obtenga el mínimo de todos los números en la lista.	número
`max`	List<number>	Obtenga el máximo de todos los números en la lista.	número
`avg`	List<number>	Obtenga el promedio de todos los números en la lista.	número
`median`	List<number>	Obtenga la mediana de todos los números en la lista.	número
`sum`	List<number>	Obtenga la suma de todos los números en la lista.	número
`count`	List<?>	Obtenga el número de elementos en la lista.	número
`stddev`	List<number>	Obtenga la desviación estándar de todos los números en la lista.	número
`flatten`	List<List<?>>	Convierta una colección de listas anidadas en una sola lista que contenga todos los elementos.	List<?>

Agregación	Input	Significado	Output
`set_intersection`	List<List<string>>	Obtenga una lista de cadenas que se encuentran en todas las listas de cadenas de una colección.	List<string>
All above	List<List<?>>	Todas las operaciones de una lista anidada actúan en cada lista secundaria de manera individual para generar una lista de resultados.	List<?>

La tabla siguiente ilustra algunas expresiones de propiedades válidas que utilizan funciones de agregación:

Expression	Significado	Tipo resultante
flatten(teams[*].players.attributes[skill])	Los atributos de "habilidad" de todos los jugadores del emparejamiento (no agrupados)	List<number>
avg(teams[red].players.attributes[skill])	Las habilidades promedio de todos los jugadores del equipo rojo	número
avg(teams[*].players.attributes[skill])	La habilidad promedio de cada equipo en el emparejamiento	List<number>

Expression	Significado	Tipo resultante
avg(flatten(teams[*].players.attributes[skill]))	El nivel de habilidad promedio de todos los jugadores en el emparejamiento. Esta expresión obtiene una lista aplanada de las habilidad es del jugador y, a continuación, hace un promedio.	número
count(teams[red].players)	El número de jugadores del equipo rojo	número
count (teams[*].players)	El número de jugadores de cada equipo en el emparejamiento	List<number>
max(avg(teams[*].players.attributes[skill]))	El nivel de habilidad más alto del equipo en el emparejamiento	número

FlexMatch eventos de emparejamiento

Amazon GameLift Servers FlexMatch Emite eventos para cada ticket de emparejamiento a medida que se procesa. Puede publicar estos eventos en un tema de Amazon SNS, como se describe en Configuración FlexMatch notificaciones de eventos. Estos eventos también se transmiten a Amazon CloudWatch Events prácticamente en tiempo real y con el máximo esfuerzo.

En este tema se describe la estructura de FlexMatch eventos y proporciona un ejemplo para cada tipo de evento. Para obtener más información sobre el estado de las entradas de matchmaking, consulta MatchmakingTicketla Amazon GameLift Servers Referencia de API.

Temas

MatchmakingSearching

Se ha insertado el ticket en el emparejamiento. Esto incluye nuevas solicitudes y solicitudes que formaron parte de un emparejamiento propuesto que generó error.

Recurso: ConfigurationArn

Detalle: tipo, entradas estimatedWaitMillis, gameSessionInfo

Ejemplo

```
{
  "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

Se ha creado un posible emparejamiento. Este evento se emite para todos los emparejamientos nuevos posibles, independientemente de que sea necesaria la aceptación.

Recurso: ConfigurationArn

Detalles: tipo, entradas, tiempo de espera de aceptación, aceptación requerida,,, MatchID ruleEvaluationMetrics gameSessionInfo

Ejemplo

```
{
  "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

Los jugadores han aceptado un posible emparejamiento. Este evento contiene el estado de aceptación actual de cada jugador en el emparejamiento. Si faltan datos, significa que no se ha solicitado el nombre de ese jugador. AcceptMatch

Recurso: ConfigurationArn

Detalles: tipo, entradas, matchID, gameSessionInfo

Ejemplo

```
{
  "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

La aceptación del emparejamiento está completa debido a la aceptación de los jugadores, el rechazo de los jugadores o el agotamiento del tiempo de aceptación.

Recurso: ConfigurationArn

Detalle: tipo, entradas, aceptación, matchID, gameSessionInfo

Ejemplo

```
{
  "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

Se ha completado correctamente el emparejamiento y se ha creado una sesión de juego.

Recurso: ConfigurationArn

Detalle: tipo, entradas, matchID, gameSessionInfo

Ejemplo

```
{
  "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

Error del ticket de emparejamiento debido a que se ha agotado el tiempo de espera.

Recurso: ConfigurationArn

Detalle: tipo, entradas, mensaje ruleEvaluationMetrics, matchID, gameSessionInfo

Ejemplo

```
{
  "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

Se ha cancelado el ticket de emparejamiento.

Recurso: ConfigurationArn

Detalle: tipo, entradas, mensaje ruleEvaluationMetrics, matchID, gameSessionInfo

Ejemplo

```
{
```

```
  "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

El ticket de emparejamiento ha encontrado un error. Esto puede deberse a que no es posible acceder a la cola de la sesión de juego o a un error interno.

Recurso: ConfigurationArn

Detalle: tipo, entradas, mensaje ruleEvaluationMetrics, matchID, gameSessionInfo

Ejemplo

```
{
  "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"
  }
}
```

Amazon GameLift Serversnotas de publicación y versiones del SDK

Las notas de la Amazon GameLift Servers versión proporcionan detalles sobre Amazon GameLift Servers las nuevas funciones, actualizaciones y correcciones relacionadas con el servicio, incluidas FlexMatch las funciones. También puedes encontrar el historial de Amazon GameLift Servers versiones de todos los complementos SDKs y complementos.

- Amazon GameLift ServersVersiones del SDK
- Amazon GameLift Serversnotas de lanzamiento

Amazon GameLift Servers recursos para desarrolladores

Para ver todos Amazon GameLift Servers la documentación y los recursos para desarrolladores, consulte la Amazon GameLift Servers página de inicio de la documentación.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.