aws

FlexMatch Guide du développeur

Amazon GameLift

Version

Amazon GameLift: FlexMatch Guide du développeur

Table of Contents

Qu'est-ce qu'Amazon GameLift FlexMatch ?

Amazon GameLift FlexMatch est un service de jumelage personnalisable pour les jeux multijoueurs. Vous pouvez ainsi créer un ensemble de règles personnalisées qui définissent à quoi ressemble un match multijoueur pour votre jeu, et qui déterminent comment évaluer et sélectionner les joueurs compatibles pour chaque match. FlexMatch Vous pouvez également affiner les principaux aspects de l'algorithme de matchmaking pour répondre aux besoins de votre jeu.

À utiliser FlexMatch comme service de jumelage autonome ou intégré à une solution d'hébergement de GameLift jeux Amazon. Par exemple, vous pouvez implémenter en FlexMatch tant que fonctionnalité autonome des jeux dotés d'une peer-to-peer architecture ou des jeux utilisant d'autres solutions de cloud computing. Vous pouvez également ajouter FlexMatch à votre hébergement EC2 GameLift géré par Amazon ou à votre hébergement sur site avec Amazon. GameLift Anywhere Ce guide fournit des informations détaillées sur la façon de créer un système de FlexMatch matchmaking pour votre scénario particulier.

FlexMatch vous donne la flexibilité de définir les priorités de matchmaking en fonction de vos exigences de jeu. Par exemple, vous pouvez effectuer les opérations suivantes :

- Trouvez le juste équilibre entre rapidité et qualité. Définissez des règles de match pour trouver rapidement des matchs suffisamment bons, ou demandez aux joueurs d'attendre un peu plus longtemps pour trouver le meilleur match possible pour une expérience de jeu optimale.
- Créez des matchs basés sur des joueurs ou des équipes bien assortis. Créez des matchs où tous les joueurs ont des caractéristiques similaires, telles que leurs compétences ou leur expérience. Ou organisez des matchs où les caractéristiques combinées de chaque équipe répondent à des critères communs.
- Hiérarchisez la façon dont la latence des joueurs entre en ligne de compte Voulez-vous fixer une limite stricte de latence pour tous les joueurs, ou des latences plus élevées sont-elles acceptables tant que tous les joueurs ont la même latence ?

> **Prêt à commencer à travailler avec FlexMatch ?**
>
> Pour obtenir step-by-step des conseils sur la mise en place et le fonctionnement de votre jeu FlexMatch, consultez les rubriques suivantes :
>
> - FlexMatchintégration avec l'GameLifthébergement Amazon

- GameLiftFlexMatchIntégration Amazon pour le matchmaking autonome

FlexMatch Caractéristiques principales

Les fonctionnalités suivantes sont disponibles dans tous les FlexMatch scénarios, que vous les utilisiez FlexMatch en tant que service autonome ou avec l'hébergement de GameLift jeux Amazon.

- Correspondance personnalisable entre joueurs. Concevez et construisez des entremetteurs adaptés à tous les modes de jeu que vous proposez à vos joueurs. Élaborez un ensemble de règles personnalisées pour évaluer les attributs clés des joueurs (tels que le niveau de compétence ou le rôle) et les données de latence géographique afin de créer des matchs de qualité pour votre jeu.
- Correspondance basée sur la latence. Fournissez des données de latence des joueurs et créez des règles de match qui obligent les joueurs à avoir des temps de réponse similaires lors d'un match. Cette fonctionnalité est utile lorsque les pools de matchmaking de vos joueurs s'étendent sur plusieurs régions géographiques.
- Support pour des matchs allant jusqu'à 200 joueurs. Créez des matchs réunissant jusqu'à 40 joueurs en utilisant des règles de match personnalisées pour votre jeu. Créez des matchs réunissant jusqu'à 200 joueurs à l'aide d'un processus de jumelage personnalisé simplifié pour réduire les temps d'attente des joueurs.
- Acceptation des joueurs. Demandez aux joueurs de s'inscrire à un match proposé avant de finaliser le match et de commencer une session de jeu. Utilisez cette fonctionnalité pour lancer votre flux de travail d'acceptation personnalisé et signaler les réponses des joueurs FlexMatch avant de créer une nouvelle session de jeu pour le match. Si tous les joueurs n'acceptent pas un match, le match proposé échoue et les joueurs qui l'ont accepté retournent automatiquement dans le pool de matchmaking.
- Soutien aux groupes de joueurs. Générez des matchs pour les groupes de joueurs qui souhaitent jouer ensemble dans la même équipe. FlexMatch À utiliser pour trouver des joueurs supplémentaires pour compléter le match selon les besoins.
- Règles de correspondance extensibles. Assouplissez progressivement les exigences de match après un certain temps sans trouver de correspondance réussie. L'extension des règles vous permet de décider où et quand assouplir les règles du match initial, afin que les joueurs puissent accéder aux parties jouables plus rapidement.

- Corrigez le remblai. Remplissez les emplacements vides d'une session de jeu existante avec de nouveaux joueurs bien adaptés. Personnalisez quand et comment recruter de nouveaux joueurs, et utilisez les mêmes règles de match personnalisées pour trouver d'autres joueurs.

FlexMatch avec l' GameLift hébergement Amazon

FlexMatch propose les fonctionnalités supplémentaires suivantes à utiliser avec les jeux que vous hébergez sur Amazon GameLift. Cela inclut les jeux dotés de serveurs de jeu personnalisés ou de serveurs en temps réel.

- Placement des sessions de jeu. Lorsqu'une correspondance est établie avec succès, demande FlexMatch automatiquement un nouveau placement de session de jeu à Amazon GameLift. Les données générées pendant le matchmaking, y compris les identifiants des joueurs et les affectations des équipes, sont fournies au serveur de jeu afin qu'il puisse utiliser ces informations pour démarrer la session de jeu pour le match. FlexMatch transmet ensuite les informations de connexion à la session de jeu afin que les clients du jeu puissent rejoindre le jeu. Pour minimiser la latence subie par les joueurs lors d'un match, le placement des sessions de jeu avec Amazon GameLift peut également utiliser les données de latence des joueurs régionales, si elles sont fournies.
- Remblayage automatique des allumettes. Lorsque cette fonctionnalité est activée, envoie FlexMatch automatiquement une demande de remplacement lorsqu'une nouvelle session de jeu commence avec des emplacements de joueur vides. Votre système de matchmaking lance le processus de placement des sessions de jeu avec un nombre minimum de joueurs, puis remplit rapidement les emplacements restants. Vous ne pouvez pas utiliser le remblayage automatique pour remplacer les joueurs qui abandonnent une session de jeu correspondante.

Si vous utilisez Amazon GameLift FleetIQ avec des jeux hébergés avec des ressources Amazon Elastic Compute Cloud (Amazon EC2), implémentez-les en tant que service autonome. FlexMatch

Tarification pour Amazon GameLift FlexMatch

Amazon GameLift facture les instances en fonction de la durée d'utilisation et de la bande passante en fonction de la quantité de données transférées. Si vous hébergez vos jeux sur des GameLift serveurs Amazon, FlexMatch leur utilisation est incluse dans les frais d'Amazon GameLift. Si vous hébergez vos jeux sur un autre serveur, FlexMatch l'utilisation est facturée séparément. Pour obtenir la liste complète des frais et des prix d'Amazon GameLift, consultez Amazon GameLift Pricing.

Pour plus d'informations sur le calcul du coût de l'hébergement de vos jeux ou du matchmaking avec Amazon GameLift, consultez Generating Amazon GameLift pricing estimates, qui décrit comment utiliser le AWS Pricing Calculator.

Comment GameLift FlexMatch fonctionne Amazon

Cette rubrique fournit une vue d'ensemble du GameLift FlexMatch service Amazon, notamment des principaux composants d'un FlexMatch système et de la manière dont ils interagissent.

Vous pouvez l'utiliser FlexMatch avec des jeux qui utilisent l'hébergement GameLift géré par Amazon ou avec des jeux qui utilisent une autre solution d'hébergement. Les jeux hébergés sur les GameLift serveurs Amazon, y compris les serveurs en temps réel, utilisent le GameLift service Amazon intégré pour localiser automatiquement les serveurs de jeu disponibles et démarrer des sessions de jeu pour les matchs. Les jeux utilisés FlexMatch en tant que service autonome, y compris Amazon GameLift FleetIQ, doivent se coordonner avec le système d'hébergement existant pour attribuer des ressources d'hébergement et démarrer des sessions de jeu pour les matchs.

Pour obtenir des instructions détaillées sur FlexMatch la configuration de vos jeux, consultezDémarrer avec FlexMatch.

Composants du matchmaking

Un système de FlexMatch matchmaking comprend certains ou tous les composants suivants.

GameLiftComposants Amazon

Ce sont GameLift des ressources Amazon qui contrôlent la manière dont le FlexMatch service effectue le matchmaking pour votre jeu. Ils sont créés et gérés à l'aide GameLift des outils Amazon, notamment la console et l'AWSinterface de ligne de commande, ou bien de manière programmatique à l'aide du AWS SDK pour Amazon. GameLift

- FlexMatchconfiguration de matchmaking (également appelée système de matchmaking) — Un système de matchmaking est un ensemble de valeurs de configuration qui personnalise le processus de matchmaking pour votre jeu. Un jeu peut avoir plusieurs systèmes de matchmaking, chacun étant configuré pour différents modes de jeu ou expériences selon les besoins. Lorsque votre jeu envoie une demande de matchmaking àFlexMatch, il spécifie quel système de matchmaking utiliser.
- FlexMatchensemble de règles de matchmaking — Un ensemble de règles contient toutes les informations nécessaires pour évaluer les joueurs en vue d'un match potentiel et les approuver ou

les rejeter. L'ensemble de règles définit la structure de l'équipe d'un match, déclare les attributs du joueur qui sont utilisés pour l'évaluation et fournit des règles qui décrivent les critères d'un match acceptable. Les règles peuvent s'appliquer à des joueurs individuels, à des équipes ou à l'ensemble du match. Par exemple, une règle peut exiger que tous les joueurs du match choisissent la même carte de jeu ou que toutes les équipes aient une moyenne de compétences similaire.

- File d'attente des sessions de GameLift jeu Amazon (uniquement pour les FlexMatch hébergements GameLift gérés par Amazon) : une file d'attente de sessions de jeu localise les ressources d'hébergement disponibles et démarre une nouvelle session de jeu pour le match. La configuration de la file d'attente détermine GameLift où Amazon recherche les ressources d'hébergement disponibles et comment sélectionner le meilleur hôte disponible correspondant à une correspondance.

Composants personnalisés

Les composants suivants incluent les fonctionnalités requises pour un FlexMatch système complet que vous devez implémenter en fonction de l'architecture de votre jeu.

- Interface joueur pour le matchmaking — Cette interface permet aux joueurs de rejoindre un match. Au minimum, il lance une demande de matchmaking via le composant du service de matchmaking client et fournit des données spécifiques aux joueurs, telles que le niveau de compétence et les données de latence, selon les besoins du processus de matchmaking.

  > **Note**
  >
  > Il est recommandé que la communication avec le FlexMatch service soit effectuée par un service principal, et non par un client de jeu.

- Service de jumelage client — Ce service répond aux demandes d'inscription des joueurs depuis l'interface du joueur, génère des demandes de matchmaking et les envoie au FlexMatch service. Pour les demandes en cours, il surveille les événements de matchmaking, suit l'état du matchmaking et prend les mesures nécessaires. Selon la façon dont vous gérez l'hébergement des sessions de jeu dans votre jeu, ce service peut renvoyer aux joueurs les informations de connexion à la session de jeu. Ce composant utilise le AWS SDK avec l'GameLiftAPI Amazon pour communiquer avec le FlexMatch service.
- Service de placement de matchs (uniquement FlexMatch en tant que service autonome) : ce composant fonctionne avec votre système d'hébergement de jeu existant pour localiser les

ressources d'hébergement disponibles et démarrer de nouvelles sessions de jeu pour les matchs. Le composant doit obtenir les résultats du matchmaking et extraire les informations nécessaires pour démarrer une nouvelle session de jeu, y compris les identifiants des joueurs, les attributs et les attributions des équipes pour tous les joueurs participant au match.

FlexMatchprocessus de jumelage

Cette rubrique décrit un scénario de matchmaking de base et les interactions entre les différents composants de votre jeu et le FlexMatch service.

Demande de matchmaking pour les joueurs

Un joueur utilisant votre client de jeu clique sur le bouton « Rejoindre le jeu ». Cette action amène le service de jumelage de votre client à envoyer une demande de jumelage àFlexMatch. La demande identifie le FlexMatch système de matchmaking à utiliser pour répondre à la demande. La demande inclut également les informations sur le joueur requises par votre système de matchmaking personnalisé, telles que le niveau de compétence, les préférences de jeu ou les données de latence géographique. Vous pouvez faire des demandes de matchmaking pour un joueur ou plusieurs joueurs.

Ajouter des demandes au pool de matchmaking

Lorsqu'il FlexMatch reçoit la demande de matchmaking, il génère un ticket de matchmaking et l'ajoute au pool de tickets du système de matchmaking. Le ticket reste dans le pool jusqu'à ce qu'il soit égalé ou qu'une limite de temps maximale soit atteinte. Votre service de jumelage client est régulièrement informé des événements de matchmaking, y compris des changements de statut des tickets.

Créez un match

Votre FlexMatch système de matchmaking exécute en permanence le processus suivant sur tous les tickets de son pool :

1. Le système de matchmaking trie le pool par âge du billet, puis commence à créer un match potentiel en commençant par le ticket le plus ancien.
2. Le système de matchmaking ajoute un deuxième ticket au match potentiel et évalue le résultat par rapport à vos règles de matchmaking personnalisées. Si le match potentiel passe l'évaluation, les joueurs du ticket sont affectés à une équipe.
3. Le système de matchmaking ajoute le ticket suivant dans l'ordre et répète le processus d'évaluation. Lorsque tous les emplacements des joueurs sont remplis, le match est prêt.

Le matchmaking pour les matchs importants (41 à 200 joueurs) utilise une version modifiée du processus décrit ci-dessus afin de pouvoir créer des matchs dans un délai raisonnable. Au lieu d'évaluer chaque ticket individuellement, le système de matchmaking divise un pool de tickets pré-triés en matchs potentiels, puis équilibre chaque match en fonction des caractéristiques du joueur que vous avez spécifiées. Par exemple, un système de matchmaking peut pré-trier les tickets en fonction d'emplacements similaires à faible latence, puis utiliser l'équilibrage après le match pour s'assurer que les équipes sont égales en termes de compétences des joueurs.

Signaler les résultats du matchmaking

Lorsqu'un match acceptable est trouvé, tous les tickets correspondants sont mis à jour et un événement de matchmaking réussi est généré pour chaque ticket correspondant.

- FlexMatchen tant que service autonome : votre jeu reçoit les résultats des matchs lors d'un événement de matchmaking réussi. Les données de résultats incluent une liste de tous les joueurs correspondants et de leurs attributions d'équipe. Si vos demandes de match contiennent des informations sur la latence des joueurs, les résultats suggèrent également un emplacement géographique optimal pour le match.
- FlexMatchavec une solution GameLift d'hébergement Amazon : les résultats des matchs sont automatiquement transmis à une GameLift file d'attente Amazon pour le placement des sessions de jeu. Le système de matchmaking détermine quelle file d'attente est utilisée pour le placement des sessions de jeu.

Démarrez une session de jeu pour le match

Une fois qu'un match proposé a été formé avec succès, une nouvelle session de jeu démarre. Vos serveurs de jeu doivent être en mesure d'utiliser les données des résultats du matchmaking, y compris les identifiants des joueurs et les attributions des équipes, lors de la configuration d'une session de jeu pour le match.

- FlexMatchen tant que service autonome : votre service de placement de matchs personnalisé obtient les résultats des matchs lors d'événements de matchmaking réussis et se connecte à votre système de placement de sessions de jeu existant pour localiser une ressource d'hébergement disponible pour le match. Une fois qu'une ressource d'hébergement est trouvée, le service de placement de matchs se coordonne avec votre système d'hébergement existant pour démarrer une nouvelle session de jeu et obtenir les informations de connexion.
- FlexMatchavec une solution GameLift d'hébergement Amazon : la file d'attente des sessions de jeu localise le meilleur serveur de jeu disponible pour le match. En fonction de la configuration de la file d'attente, elle essaie de placer la session de jeu avec les ressources les moins coûteuses et là où les joueurs bénéficieront d'une faible latence (si les données de latence

des joueurs sont fournies). Une fois la session de jeu lancée avec succès, le GameLift service Amazon invite le serveur de jeu à démarrer une nouvelle session de jeu, en transmettant les résultats du matchmaking et d'autres données de jeu facultatives.

Connectez les joueurs au match

Après le début d'une session de jeu, les joueurs s'y connectent, revendiquent leur affectation d'équipe et commencent à jouer.

- FlexMatchen tant que service autonome : votre jeu utilise le système de gestion de session de jeu existant pour fournir des informations de connexion aux joueurs.
- FlexMatchavec une solution GameLift d'hébergement Amazon : lorsque le placement d'une session de jeu est réussi, FlexMatch met à jour tous les tickets correspondants avec les informations de connexion à la session de jeu et l'identifiant de session du joueur.

FlexMatch pris en charge Régions AWS

Si vous utilisez une solution FlexMatch d' GameLift hébergement Amazon, vous pouvez organiser des sessions de jeu correspondantes partout où vous hébergez des jeux. Consultez la liste complète Régions AWS et les emplacements de l' GameLift hébergement Amazon.

Pour tous les FlexMatch utilisateurs, vous pouvez héberger FlexMatch des ressources, y compris des configurations de matchmaking et des ensembles de règles, dans les catégories suivantes prises en chargeRégions AWS. veuillez consulter Choisissez un lieu pour l'entremetteur.

Nom de l'Région AWS	Code région
US East (Virginie du Nord)	us-east-1
USA Ouest (Oregon)	us-west-2
Asie-Pacifique (Séoul)	AP-Northest-2
Asie-Pacifique (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Europe (Francfort)	eu-central-1
Europe (Irlande)	eu-west-1

Nom de l'Région AWS	Code région
Chine (régions de Pékin et de Ningxia)	

Configuration de FlexMatch

Amazon GameLift FlexMatch est un AWS service et vous devez disposer d'un AWS compte pour utiliser ce service. La création d'un compte AWS est gratuite. Pour plus d'informations sur ce que vous pouvez faire avec un AWS compte, consultez la section Mise en route AWS.

Si vous les utilisez FlexMatch avec d'autres GameLift solutions Amazon, consultez les rubriques suivantes :

- Configuration de l'accès à l'GameLifthébergement Amazon et aux serveurs en temps réel
- Configuration de l'accès pour l'hébergement sur Amazon EC2 avec Amazon GameLift FleetIQ

Pour configurer votre compte pour Amazon GameLift

1. Obtenez un compte. Ouvrez Amazon Web Services et choisissez Se connecter à la console. Suivez les instructions pour créer un nouveau compte ou pour vous connecter à un compte existant.
2. Configurez un groupe d'utilisateurs administratifs. Ouvrez la console de service AWS Identity and Access Management (IAM) et suivez les étapes pour créer ou mettre à jour des utilisateurs ou des groupes d'utilisateurs. IAM gère l'accès à vos AWS services et ressources. Toutes les personnes qui accèdent à vos FlexMatch ressources, à l'aide de la GameLift console Amazon ou en appelant GameLift les API Amazon, doivent bénéficier d'un accès explicite. Pour obtenir des instructions détaillées sur l'utilisation de la console (ou de l'AWSinterface de ligne de commande ou d'autres outils) pour configurer des groupes d'utilisateurs, voir Création d'utilisateurs IAM.
3. Attachez une politique d'autorisations à votre utilisateur ou à votre groupe d'utilisateurs.L'accès aux AWS services et aux ressources est géré en attachant une politique IAM à un utilisateur ou à un groupe d'utilisateurs. Les politiques d'autorisations définissent un ensemble de AWS services et d'actions auxquels un utilisateur doit avoir accès.

   Pour AmazonGameLift, vous devez créer une politique d'autorisations personnalisée et l'associer à chaque utilisateur ou groupe d'utilisateurs. Une stratégie est un document JSON. Utilisez l'exemple ci-dessous pour créer votre politique.

L'exemple suivant illustre une politique d'autorisations intégrée avec des autorisations administratives pour toutes les GameLift ressources et actions Amazon. Vous pouvez choisir de limiter l'accès en spécifiant uniquement FlexMatch des éléments spécifiques.

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

Démarrer avec FlexMatch

Utilisez les ressources de cette section pour vous aider à créer un système de matchmaking avec. FlexMatch

Rubriques

- GameLiftFlexMatchIntégration Amazon pour le matchmaking autonome
- FlexMatchintégration avec l'GameLifthébergement Amazon

GameLiftFlexMatchIntégration Amazon pour le matchmaking autonome

Cette rubrique décrit le processus d'intégration complet pour la mise en œuvre en FlexMatch tant que service de jumelage autonome. Utilisez ce processus si votre jeu multijoueur est hébergé à l'aide peer-to-peer de matériel local configuré sur mesure ou d'autres primitives de cloud computing. Ce processus est également destiné à être utilisé avec Amazon GameLift FleetIQ, qui est une solution d'optimisation de l'hébergement pour les jeux hébergés sur Amazon EC2. Si vous hébergez votre jeu via l'hébergement GameLift géré par Amazon (y compris les serveurs en temps réel), consultezFlexMatchintégration avec l'GameLifthébergement Amazon.

Avant de commencer l'intégration, vous devez disposer d'un AWS compte et configurer les autorisations d'accès au GameLift service Amazon. Pour plus de détails, consultez Configuration de FlexMatch. Toutes les tâches essentielles liées à la création et à la gestion des GameLift FlexMatch entremetteurs et des ensembles de règles Amazon peuvent être effectuées à l'aide de la GameLift console Amazon.

1. Créez un ensemble FlexMatch de règles de matchmaking. Votre ensemble de règles personnalisé fournit des instructions complètes sur la manière de créer une correspondance. Vous y définissez la structure et la taille de chaque équipe. Vous fournissez également un ensemble d'exigences auxquelles un match doit satisfaire pour être valide, qui permet FlexMatch d'inclure ou d'exclure des joueurs dans un match. Ces exigences peuvent s'appliquer à des joueurs individuels. Vous pouvez également personnaliser l'FlexMatchalgorithme dans l'ensemble de règles, par exemple pour créer des matchs de grande envergure réunissant jusqu'à 200 joueurs. Consultez ces rubriques :

   - Création d'un ensemble de FlexMatch règles

- FlexMatch exemples d'ensembles de règles

2. Configurez des notifications pour les événements de matchmaking. Utilisez les notifications pour suivre l'activité de FlexMatch matchmaking, y compris l'état des demandes de match en attente. Il s'agit du mécanisme utilisé pour fournir les résultats d'une proposition de correspondance. Les demandes de mise en relation étant asynchrones, vous devez disposer d'un moyen pour suivre le statut des demandes. L'utilisation des notifications est l'option préférée pour cela. Consultez ces rubriques :

   - Configurer les notifications FlexMatch d'événements
   - FlexMatchévénements de matchmaking

3. Configurez une configuration de FlexMatch matchmaking. Également appelé entremetteur, ce composant reçoit les demandes de matchmaking et les traite. Vous configurez un système de matchmaking en spécifiant un ensemble de règles, une cible de notification et un temps d'attente maximal. Vous pouvez également activer des fonctionnalités facultatives. Consultez ces rubriques :

   - Concevez un FlexMatch entremetteur
   - Créer une configuration de matchmaking

4. Créez un service de jumelage de clients. Créez ou développez un service client de jeu avec des fonctionnalités permettant de créer et d'envoyer des demandes de matchmaking àFlexMatch. Pour créer des demandes de matchmaking, ce composant doit disposer de mécanismes permettant d'obtenir les données sur les joueurs requises par l'ensemble de règles de matchmaking et, éventuellement, des informations de latence régionales. Il doit également disposer d'une méthode permettant de créer et d'attribuer des identifiants de ticket uniques pour chaque demande. Vous pouvez également choisir de créer un flux de travail d'acceptation des joueurs qui oblige les joueurs à accepter un match proposé. Ce service doit également surveiller les événements de matchmaking pour obtenir les résultats des matchs et lancer le placement des sessions de jeu pour que les matchs soient réussis. Consultez cette rubrique :

   - Ajouter FlexMatch à un client de jeu

5. Créez un service de placement de matchs. Créez un mécanisme qui fonctionne avec votre système d'hébergement de jeu existant pour localiser les ressources d'hébergement disponibles et démarrer de nouvelles sessions de jeu pour des matchs réussis. Ce composant doit pouvoir utiliser les informations relatives aux résultats des matchs pour obtenir un serveur de jeu disponible et démarrer une nouvelle session de jeu pour le match. Vous pouvez également mettre en œuvre un flux de travail pour effectuer des demandes de remplissage de matchs,

qui utilise le matchmaking pour pourvoir les emplacements vacants dans les sessions de jeu correspondantes déjà en cours.

FlexMatchintégration avec l'GameLifthébergement Amazon

FlexMatchest disponible avec l'GameLifthébergement Amazon géré pour les serveurs de jeux personnalisés et les serveurs en temps réel. Pour ajouter la mise en relation FlexMatch à votre jeu, veuillez exécuter les tâches suivantes.

- Définir un matchmaker. Un matchmaker reçoit les demandes de mise en relation des joueurs et les traite. Il regroupe les joueurs en fonction d'un ensemble de règles définies et, pour chaque mise en relation réussie, crée une nouvelle session de jeu et de joueur. Suivez ces étapes pour configurer un matchmaker :
  - Créer un ensemble de règles. Un ensemble de règles indique au matchmaker comment construire une mise en relation valide. Il spécifie la structure de l'équipe, ainsi que la manière d'évaluer les joueurs afin de les inclure dans une mise en relation. Consultez ces rubriques :
    - Création d'un ensemble de FlexMatch règles
    - FlexMatch exemples d'ensembles de règles
  - Créer une file d'attente de session de jeu. Une file d'attente localise la région la mieux adaptée à chaque mise en relation et crée une nouvelle session de jeu dans cette région. Vous pouvez utiliser une file d'attente existante ou en créer une nouvelle pour la mise en relation. Consultez cette rubrique :
    - Création d'une file d'attente
  - Configurer des notifications (facultatif). Les demandes de mise en relation étant asynchrones, vous devez disposer d'un moyen pour suivre le statut des demandes. Les notifications représentent la meilleure option. Consultez cette rubrique :
    - Configurer les notifications FlexMatch d'événements
  - Configurer un matchmaker. Une fois que vous disposez d'un ensemble de règles, d'une file d'attente et d'une cible pour les notifications, créez la configuration pour votre matchmaker. Consultez ces rubriques :
    - Concevez un FlexMatch entremetteur
    - Créer une configuration de matchmaking
- Intégrer FlexMatch à votre service de client de jeu. Ajoutez des fonctionnalités au service du client de jeu pour démarrer de nouvelles sessions de jeu avec la mise en relation. Les demandes

de mise en relation spécifient le matchmaker à utiliser et fournissent des données de joueur nécessaires pour la mise en relation. Consultez cette rubrique :

- Ajouter FlexMatch à un client de jeu

- Intégrer FlexMatch à votre serveur de jeux. Ajoutez des fonctionnalités à votre serveur de jeux pour démarrer des sessions créées par le biais de la mise en relation. Les demandes de ce type de session de jeu incluent des informations spécifiques à la partie, y compris les joueurs et les affectations d'équipe. Le serveur de jeux a besoin de pouvoir utiliser ces informations et y accéder lors de la construction d'une session de jeu pour la mise en relation. Consultez cette rubrique :
  - Ajouter FlexMatch à un serveur de jeu GameLift hébergé sur Amazon
- Configurer le remplissage FlexMatch (facultatif). Demandez des mises en relation supplémentaires de joueurs afin de remplir les emplacements ouverts dans les jeux existants. Vous pouvez activer le remblayage automatique pour qu'Amazon GameLift gère les demandes de remblayage. Vous pouvez également gérer manuellement le remplissage en ajoutant des fonctionnalités à votre client ou serveur de jeu afin de pouvoir initier des demandes de remplissage. Consultez cette rubrique :
  - Remplir les jeux existants avec FlexMatch

> **Note**
>
> FlexMatchLe backfill n'est actuellement pas disponible pour les jeux utilisant des serveurs en temps réel.

Créer un entremetteur Amazon GameLift FlexMatch

Un matchmaker FlexMatch effectue la tache de créer une mise en relation pour un jeu. Il gère le pool des demandes de mise en relation reçues, forme les équipes d'une mise en relation, traite et sélectionne les joueurs pour trouver les meilleurs groupes de joueurs possibles et lance le processus de placement et de démarrage d'une session de jeu pour la mise en relation. Cette rubrique décrit les principaux aspects d'un matchmaker et explique comment configurer un matchmaker personnalisé pour votre jeu.

Pour une description détaillée de la façon dont un matchmaker FlexMatch traite les demandes de mise en relation qu'il reçoit, consultez FlexMatchprocessus de jumelage.

Rubriques

Concevez un FlexMatch entremetteur

Cette rubrique fournit des conseils sur la façon de concevoir un système de matchmaking adapté à votre jeu.

Configurer un système de matchmaking de base

Au minimum, un entremetteur a besoin des éléments suivants :

- L'ensemble de règles détermine la taille et le périmètre des équipes pour une mise en relation et définit un ensemble de règles à utiliser lors de l'évaluation des joueurs d'une mise en relation. Chaque matchmaker est configuré pour utiliser un seul ensemble de règles. Consultez Création d'un ensemble de FlexMatch règles et FlexMatch exemples d'ensembles de règles.
- La cible de notification reçoit toutes les notifications d'événements de matchmaking. Vous devez configurer un sujet Amazon Simple Notification Service (SNS), puis ajouter l'ID du sujet au système de jumelage. Pour plus d'informations sur la configuration des notifications, consultez Configurer les notifications FlexMatch d'événements.

- Le délai d'attente détermine la durée pendant laquelle des demandes de mise en relation peuvent rester dans le pool de demandes et être évaluée pour des mises en relation potentielles. Une fois qu'une demande a dépassé le délai, elle a échoué à créer une mise en relation et elle supprimée du pool.
- Lorsque vous utilisez FlexMatch l'hébergement GameLift géré par Amazon, la file d'attente des sessions de jeu trouve les meilleures ressources disponibles pour héberger une session de jeu pour le match et démarre une nouvelle session de jeu. Chaque file d'attente est configurée avec une liste d'emplacements et de types de ressources (y compris les instances ponctuelles ou à la demande) qui déterminent où les sessions de jeu peuvent être placées. Pour plus d'informations sur les files d'attente, consultez la section Utilisation de files d'attente multisites.

Choisissez un lieu pour l'entremetteur

Décidez où vous souhaitez que l'activité de matchmaking ait lieu et créez votre configuration de matchmaking et votre ensemble de règles à cet endroit. Amazon GameLift gère des pools de tickets pour les demandes de match de votre jeu, où ils sont triés et évalués pour déterminer s'ils correspondent à des matchs viables. Après avoir fait une correspondance, Amazon GameLift envoie les détails du match pour le placement de la session de jeu. Vous pouvez exécuter les sessions de jeu correspondantes dans n'importe quel endroit compatible avec votre solution d'hébergement.

Consultez FlexMatch pris en charge Régions AWS les emplacements où vous pouvez créer des FlexMatch ressources.

Lorsque vous choisissez un match Région AWS pour votre système de matchmaking, réfléchissez à l'impact de l'emplacement sur les performances et à la manière dont il peut optimiser l'expérience de match pour les joueurs. Nous recommandons les bonnes pratiques suivantes :

- Placez un entremetteur dans un endroit proche de vos joueurs et de votre service client qui envoie des demandes de FlexMatch matchmaking. Cette approche réduit l'effet de latence sur votre flux de travail de demande de matchmaking et le rend plus efficace.
- Si votre jeu atteint un public mondial, pensez à créer des entremetteurs à plusieurs endroits et à acheminer les demandes de match vers le système de matchmaking le plus proche du joueur. En plus d'améliorer l'efficacité, cela entraîne la formation de pools de tickets avec des joueurs géographiquement proches les uns des autres, ce qui améliore la capacité du système de matchmaking à associer des joueurs en fonction des exigences de latence.

- Lorsque vous utilisez FlexMatch l'hébergement GameLift géré par Amazon, placez votre système de matchmaking et la file d'attente de session de jeu qu'il utilise au même endroit. Cela permet de réduire la latence des communication entre le matchmaker et la file d'attente.

Ajouter des éléments facultatifs

En plus de ces exigences minimales, vous pouvez configurer votre matchmaker avec les options supplémentaires suivantes. Si vous utilisez une solution FlexMatch d' GameLift hébergement Amazon, de nombreuses fonctionnalités sont intégrées. Si vous l'utilisez en FlexMatch tant que service de jumelage autonome, vous voudrez peut-être intégrer ces fonctionnalités à votre système.

Acceptation des joueurs

Vous pouvez configurer un système de matchmaking pour exiger que tous les joueurs sélectionnés pour un match acceptent de participer. Si votre système exige une acceptation, tous les joueurs doivent avoir la possibilité d'accepter ou de rejeter un match proposé. Une mise en relation doit recevoir les acceptations de tous les joueurs avant d'être effective. Si un joueur refuse ou n'accepte pas un match, le match proposé est annulé et les tickets sont traités comme suit. Les tickets pour lesquels tous les joueurs figurant sur le ticket ont accepté le match sont renvoyés au matchmaking pool pour un traitement ultérieur. Les tickets pour lesquels au moins un joueur a refusé le match ou n'a pas répondu sont considérés comme un échec et ne sont plus traités. L'acceptation des joueurs nécessite une limite de temps ; tous les joueurs doivent accepter un match proposé dans le délai imparti pour que le match puisse continuer.

Mode de remplissage

Utilisez FlexMatch le remblayage pour que vos sessions de jeu soient remplies de nouveaux joueurs parfaitement adaptés pendant toute la durée de la session de jeu. Lors du traitement des demandes de remplacement, FlexMatch utilise le même système de matchmaking que celui utilisé pour associer les joueurs d'origine. Vous pouvez personnaliser la façon dont les tickets de remblayage sont priorisés avec des tickets pour les nouveaux matchs, en plaçant les tickets de remblayage en première ou en fin de file. Cela signifie que, lorsque de nouveaux joueurs entrent dans le matchmaking pool, ils sont plus ou moins susceptibles d'être placés dans un jeu existant que dans un jeu nouvellement créé.

Le remplissage manuel est disponible, que votre jeu soit utilisé FlexMatch avec un GameLift hébergement Amazon géré ou avec d'autres solutions d'hébergement. Le remplissage manuel vous offre la flexibilité dont vous avez besoin pour déterminer quand déclencher une demande de

remplissage. Par exemple, vous souhaiterez peut-être ajouter de nouveaux joueurs uniquement pendant certaines phases de votre jeu ou uniquement lorsque certaines conditions sont réunies.

Le remplissage automatique n'est disponible que pour les jeux utilisant un GameLift hébergement Amazon géré. Lorsque cette fonctionnalité est activée, si une session de jeu commence avec des machines à sous ouvertes, Amazon GameLift commence à générer automatiquement des demandes de remplacement pour celle-ci. Cette fonctionnalité vous permet de configurer le matchmaking afin que les nouvelles parties commencent avec un nombre minimum de joueurs, puis soient rapidement remplies au fur et à mesure que de nouveaux joueurs entrent dans le pool de matchmaking. Vous pouvez désactiver le remblayage automatique à tout moment pendant la durée de vie de la session de jeu.

Propriétés du jeu

Pour les jeux utilisés FlexMatch avec l'hébergement GameLift géré par Amazon, vous pouvez fournir des informations supplémentaires à transmettre à un serveur de jeu chaque fois qu'une nouvelle session de jeu est demandée. Cela peut être un moyen utile de transmettre les configurations du mode jeu nécessaires au démarrage d'une session de jeu pour le type de matchs en cours de création. Toutes les sessions de jeu pour les matchs créées par un système de matchmaking reçoivent le même ensemble de propriétés de jeu. Vous pouvez modifier les informations sur les propriétés du jeu en créant différentes configurations de matchmaking.

Emplacements de joueur réservés

Vous pouvez définir certains emplacements de joueur comme étant réservés et remplis ultérieurement. Pour cela, vous devez configurer la propriété « additional player count » d'une configuration de mise en relation.

Données d'événement personnalisées

Utilisez cette propriété pour inclure un ensemble d'informations personnalisées dans tous les événements liés à l'activité de mise en relation et destinés au matchmaker. Cette fonction peut être utile pour le suivi d'une activité spécifique à votre jeu, y compris le suivi des performances de vos matchmakers.

Création d'un ensemble de FlexMatch règles

Chaque matchmaker FlexMatch doit disposer d'un ensemble de règles. L'ensemble de règles détermine les deux éléments clés d'une mise en relation : la structure et la taille de l'équipe, et le mode de regroupement des joueurs pour la meilleure expérience possible.

Par exemple, un ensemble de règles peut décrire une mise en relation comme suit : Créer une mise en relation avec deux équipes de 5 joueurs chacune, une équipe jouant le rôle du défenseur et l'autre celle de l'envahisseur. Une équipe peut être composée de joueurs novices et expérimentés, mais les compétences moyennes des deux équipes doivent se situer à moins de 10 points l'une de l'autre. Si aucune correspondance n'est trouvée après 30 secondes, assouplissez progressivement les exigences de compétence.

Les rubriques de cette section décrivent comment concevoir et créer un ensemble de règles de mise en relation. Lorsque vous créez un ensemble de règles, vous pouvez utiliser la GameLift console Amazon ou l'AWSinterface de ligne de commande.

Rubriques

Conception d'un ensemble de FlexMatch règles

Cette rubrique traite de la structure de base d'un ensemble de règles et explique comment créer un ensemble de règles pour les petits matchs jusqu'à 40 joueurs. Un ensemble de règles de matchmaking fait deux choses : définir la structure et la taille de l'équipe d'un match et indiquer au système de matchmaking comment choisir les joueurs pour former le meilleur match possible.

Mais votre ensemble de règles de matchmaking peut faire plus. Par exemple, vous pouvez :

- Optimisez l'algorithme de matchmaking pour votre jeu.
- Définissez des exigences de latence minimale pour les joueurs afin de protéger la qualité du jeu.
- Assouplissez progressivement les exigences de l'équipe et les règles des matchs au fil du temps afin que tous les joueurs actifs puissent trouver un match acceptable quand ils le souhaitent.
- Définissez la gestion des demandes de matchmaking de groupe à l'aide de l'agrégation de groupes.
- Organisez des matchs importants de 40 joueurs ou plus. Pour plus d'informations sur la création d'allumettes volumineuses, consultezConcevez un ensemble FlexMatch de règles pour les matchs importants.

Lorsque vous créez un ensemble de règles de matchmaking, prenez en compte les tâches facultatives et obligatoires suivantes :

- Décrire l'ensemble de règles (obligatoire)
- Personnalisez l'algorithme de correspondance
- Déclarer les attributs du joueur
- Définissez les équipes de match
- Définissez des règles pour l'appariement des joueurs
- Laissez les exigences s'assouplir au fil du temps

Vous pouvez créer votre ensemble de règles à l'aide de la GameLift console Amazon ou de l'`CreateMatchmakingRuleSet`opération.

Décrire l'ensemble de règles (obligatoire)

Fournissez les détails relatifs à l'ensemble de règles.

- nom (facultatif) : étiquette descriptive pour votre propre usage. Cette valeur n'est pas associée au nom de l'ensemble de règles que vous spécifiez lors de la création de l'ensemble de règles avec AmazonGameLift.
- ruleLanguageVersion— Version du langage d'expression de propriété utilisé pour créer des FlexMatch règles. La valeur doit être`1.0`.

Personnalisez l'algorithme de correspondance

FlexMatchoptimise l'algorithme par défaut de la plupart des jeux afin de permettre aux joueurs de participer à des matchs acceptables avec un minimum de temps d'attente. Vous pouvez personnaliser l'algorithme et ajuster le matchmaking pour votre jeu.

Voici l'algorithme de FlexMatch matchmaking par défaut :

1. FlexMatchplace tous les tickets de matchmaking ouverts et les tickets de remplissage dans un pool de tickets.
2. FlexMatchregroupe aléatoirement les tickets du pool en un ou plusieurs lots. Au fur et à mesure que le pool de tickets augmente, il FlexMatch forme des lots supplémentaires afin de maintenir une taille de lot optimale.
3. FlexMatchtrie les billets par âge, au sein de chaque lot.

4. FlexMatchcrée une correspondance en fonction du ticket le plus ancien de chaque lot.

Pour personnaliser l'algorithme de correspondance, ajoutez un `algorithm` composant au schéma de votre ensemble de règles. Voir FlexMatchschéma d'ensemble de règles pour les informations de référence complètes.

Utilisez les personnalisations facultatives suivantes pour influencer les différentes étapes de votre processus de matchmaking.

- Ajouter le tri préalable aux lots
- Formez des lots en fonction des attributs BatchDistance
- Prioriser les tickets de remblayage
- Privilégiez les anciens billets avec des extensions

Ajouter le tri préalable aux lots

Vous pouvez trier le pool de tickets avant de former des lots. Ce type de personnalisation est particulièrement efficace pour les jeux comportant de grands pools de tickets. Le tri préalable par lots peut aider à accélérer le processus de matchmaking et à améliorer l'uniformité des caractéristiques définies par les joueurs.

Définissez les méthodes de tri par lots à l'aide de la propriété de l'algorithme. `batchingPreference` Le paramètre par défaut est `random`.

Les options permettant de personnaliser le tri préalable au lot sont les suivantes :

- Triez selon les attributs du joueur. Fournissez une liste des attributs des joueurs pour trier à l'avance le pool de tickets.

  Pour trier par attributs de joueur, configurez `batchingPreference` et définissez votre liste d'attributs de joueur dans`sortByAttributes`. `sorted` Pour utiliser un attribut, déclarez-le d'abord dans le `playerAttributes` composant de l'ensemble de règles.

  Dans l'exemple suivant, FlexMatch trie le pool de tickets en fonction de la carte de jeu préférée des joueurs, puis en fonction de leurs compétences. Les lots qui en résultent sont plus susceptibles de contenir des joueurs aux compétences similaires qui souhaitent utiliser la même carte.

```
"algorithm": {
```

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

- Triez par latence. Créez des correspondances avec la latence la plus faible disponible ou créez rapidement des correspondances avec une latence acceptable. Cette personnalisation est utile pour les ensembles de règles constituant de grands matchs de plus de 40 joueurs.

  Définissez la propriété de l'algorithme `strategy` sur`balanced`. La stratégie équilibrée limite les types d'énoncés de règles disponibles. Pour plus d'informations, veuillez consulter Concevez un ensemble FlexMatch de règles pour les matchs importants.

  FlexMatchtrie les tickets en fonction des données de latence signalées par les joueurs de l'une des manières suivantes :

  - Emplacements à latence la plus faible. Le pool de tickets est pré-trié en fonction des emplacements où les joueurs signalent leurs valeurs de latence les plus faibles. FlexMatchregroupe ensuite les tickets avec une faible latence aux mêmes emplacements, créant ainsi une meilleure expérience de jeu. Cela réduit également le nombre de tickets dans chaque lot, de sorte que le matchmaking peut prendre plus de temps. Pour utiliser cette personnalisation, `batchingPreference` réglez-la sur`fastestRegion`, comme indiqué dans l'exemple suivant.

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

  - La latence acceptable correspond rapidement. Le pool de tickets est pré-trié en fonction des emplacements où les joueurs signalent une valeur de latence acceptable. Cela permet de former moins de lots contenant plus de tickets. Avec un plus grand nombre de tickets par lot, il est plus rapide de trouver des matchs acceptables. Pour utiliser cette personnalisation, définissez la propriété `batchingPreference` sur `largestPopulation`, comme indiqué dans l'exemple suivant.

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

> **Note**
>
> La valeur par défaut de la stratégie équilibrée est`largestPopulation`.

Prioriser les tickets de remblayage

Si votre jeu implémente le remplissage automatique ou le remplissage manuel, vous pouvez personnaliser la façon dont les tickets de matchmaking sont FlexMatch traités en fonction du type de demande. Le type de demande peut être une nouvelle correspondance ou une demande de remplissage. Par défaut, FlexMatch traite les deux types de demandes de la même manière.

La priorisation du remblayage a un impact sur la façon dont les tickets FlexMatch sont gérés une fois qu'ils ont été groupés. La priorisation des remplissages nécessite des ensembles de règles permettant d'utiliser une stratégie de recherche exhaustive.

FlexMatchne correspond pas à plusieurs tickets de remblayage.

Pour modifier la priorité des tickets de remplissage, définissez la propriété. `backfillPriority`

- Faites correspondre les tickets de remblayage en premier. Cette option essaie de faire correspondre les tickets de remplacement avant de créer de nouvelles correspondances. Cela signifie que les nouveaux joueurs ont plus de chances de rejoindre une partie existante.

  Il est préférable de l'utiliser si votre jeu utilise le remblayage automatique. Le remblayage automatique est souvent utilisé dans les jeux où les sessions de jeu sont courtes et où le taux de rotation des joueurs est élevé. Le remplissage automatique permet à ces jeux de former un minimum de parties viables et de les lancer tout en FlexMatch recherchant plus de joueurs pour occuper les places disponibles.

  Définissez `backfillPriority` sur `high`.

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

- Faites correspondre les tickets de remblayage en dernier. Cette option ignore les tickets de remplissage jusqu'à ce qu'elle évalue tous les autres tickets. Cela signifie qu'il FlexMatch réintègre

les joueurs entrants dans des jeux existants lorsqu'il n'est pas possible de les intégrer à de nouveaux jeux.

Cette option est utile lorsque vous souhaitez utiliser le remblayage comme option de dernière chance pour faire participer des joueurs à une partie, par exemple lorsqu'il n'y a pas assez de joueurs pour former un nouveau match.

Définissez `backfillPriority` sur `low`.

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

Privilégiez les anciens billets avec des extensions

Les règles d'extension assouplissent les critères de match lorsque les matchs sont difficiles à terminer. Amazon GameLift applique des règles d'extension lorsque les billets d'un match partiellement terminé atteignent un certain âge. Les horodatages de création des tickets déterminent à quel moment Amazon GameLift applique les règles ; par défaut, FlexMatch suit l'horodatage du dernier ticket correspondant.

Pour modifier l'FlexMatchapplication des règles d'extension, définissez la propriété `expansionAgeSelection` comme suit :

- Développez en fonction des derniers billets. Cette option applique les règles d'extension en fonction du dernier ticket ajouté au match potentiel. Chaque fois qu'un nouveau ticket FlexMatch correspond à un nouveau ticket, l'horloge est remise à zéro. Avec cette option, les correspondances obtenues ont tendance à être de meilleure qualité mais prennent plus de temps à être mises en correspondance ; les demandes de correspondance peuvent expirer avant d'être terminées si elles mettent trop de temps à correspondre. Réglez `expansionAgeSelection` sur`newest`. `newest`est la valeur par défaut.
- Développez en fonction des billets les plus anciens. Cette option applique les règles d'extension en fonction du ticket le plus ancien du match potentiel. Cette option permet d'FlexMatchappliquer les extensions plus rapidement, ce qui réduit les temps d'attente pour les joueurs les plus proches, mais diminue la qualité des parties pour tous les joueurs. Définissez `expansionAgeSelection` sur `oldest`.

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

Déclarer les attributs du joueur

Dans cette section, listez les attributs individuels des joueurs à inclure dans les demandes de matchmaking. Vous pouvez déclarer les attributs d'un joueur dans un ensemble de règles pour deux raisons :

- Lorsque l'ensemble de règles contient des règles qui dépendent des attributs du joueur.
- Lorsque vous souhaitez transmettre un attribut de joueur à la session de jeu via la demande de match. Par exemple, vous souhaiterez peut-être transmettre les choix de personnages des joueurs à la session de jeu avant que chaque joueur ne se connecte.

Lorsque vous déclarez un attribut de joueur, incluez les informations suivantes :

- nom (obligatoire) — Cette valeur doit être unique pour l'ensemble de règles.
- type (obligatoire) — Type de données de la valeur de l'attribut. Les types de données valides sont number, string ou string map.
- par défaut (facultatif) — Entrez une valeur par défaut à utiliser si une demande de matchmaking ne fournit pas de valeur d'attribut. Si aucune valeur par défaut n'est déclarée et qu'une demande ne contient pas de valeur, FlexMatch impossible de répondre à la demande.

Définissez les équipes de match

Décrivez la structure et la taille des équipes pour une partie. Chaque partie doit être associée à au moins une équipe. Le nombre total d'équipes n'est pas limité. Vos équipes peuvent avoir le même nombre de joueurs ou être asymétriques. Par exemple, vous pouvez définir une équipe d'une seule personne pour jouer le monstre face à une équipe de 10 joueurs comme chasseurs.

FlexMatch traite les demandes de mise en relation selon que la partie implique un nombre de joueurs faible ou élevé, comme défini dans l'ensemble de règles. Les matchs potentiels réunissant jusqu'à 40 joueurs sont des petits matchs, tandis que les matchs avec plus de 40 joueurs sont des matchs importants. Pour déterminer la taille d'une partie potentielle dans un ensemble de règles, ajoutez les paramètres maxPlayer pour toutes les équipes définies dans cet ensemble de règles.

- nom (obligatoire) — Attribuez un nom unique à chaque équipe. Vous utilisez ce nom dans les règles et les extensions, ainsi que dans les FlexMatch références pour les données de matchmaking d'une session de jeu.
- MaxPlayers (obligatoire) — Spécifiez le nombre maximum de joueurs à affecter à l'équipe.
- MinPlayers (obligatoire) — Spécifiez le nombre minimum de joueurs à affecter à l'équipe.
- quantité (facultatif) — Spécifiez le nombre d'équipes à former avec cette définition. Lorsqu'un match est FlexMatch créé, il donne à ces équipes le nom fourni avec un numéro ajouté. Par exemple `Red-Team1Red-Team2`, et`Red-Team3`.

FlexMatchessaie de remplir les équipes jusqu'à la taille maximale mais crée des équipes avec moins de joueurs. Si vous souhaitez que toutes les équipes de la partie soient de taille égale, vous pouvez créer une règle à cette fin. Consultez la FlexMatch exemples d'ensembles de règles rubrique pour un exemple de `EqualTeamSizes` règle.

Définissez des règles pour l'appariement des joueurs

Créez un ensemble d'énoncés de règles qui évaluent l'admissibilité des joueurs à un match. Les règles peuvent définir des exigences qui s'appliquent aux joueurs individuels, à des équipes ou à une partie entière. Lorsqu'Amazon GameLift traite une demande de match, elle part du joueur le plus âgé du groupe de joueurs disponibles et crée un match autour de ce joueur. Pour obtenir une aide détaillée sur la création de FlexMatch règles, consultezFlexMatchtypes de règles.

- name (obligatoire) : nom significatif identifiant de manière unique la règle au sein d'un ensemble de règles. Les noms de règle sont également référencés dans les journaux des événements et les métriques qui suivent l'activité associée à cette règle.
- description (facultatif) : utilisez cet élément pour joindre une description sous forme de texte libre.
- type (obligatoire) — L'élément type identifie l'opération à utiliser lors du traitement de la règle. Chaque type de règle nécessite un ensemble de propriétés supplémentaires. Pour consulter la liste des types de règles et des propriétés valides, voir FlexMatchlangage des règles.
- Propriété du type de règle (peut être obligatoire) : selon le type de règle défini, il se peut que vous deviez définir certaines propriétés de règle. Pour en savoir plus sur les propriétés et sur l'utilisation du langage d'expression des propriétés FlexMatch, voir FlexMatchlangage des règles.

Laissez les exigences s'assouplir au fil du temps

Les extensions vous permettent d'assouplir les critères des règles au fil du temps lorsque FlexMatch vous ne trouvez pas de correspondance. Cette fonctionnalité FlexMatch garantit la disponibilité d'un meilleur produit lorsqu'il n'est pas possible d'obtenir une correspondance parfaite. En assouplissant vos règles grâce à une extension, vous augmentez progressivement le nombre de joueurs qui correspondent à un match acceptable.

Les extensions commencent lorsque l'âge du ticket le plus récent d'un match incomplet correspond au délai d'attente d'une extension. Lorsque FlexMatch vous ajoutez un nouveau ticket au match, le temps d'attente de l'extension peut être réinitialisé. Vous pouvez personnaliser le début des extensions dans la `algorithm` section de l'ensemble de règles.

Voici un exemple d'extension qui augmente progressivement le niveau de compétence minimum requis pour le match. L'ensemble de règles utilise une règle de distance, nommée SkillDeltapour exiger que tous les joueurs participant à un match se situent à moins de 5 niveaux de compétence les uns des autres. Si aucun nouveau match n'est créé pendant quinze secondes, cette extension recherche une différence de niveau de compétence de 10, puis dix secondes plus tard, une différence de 20.

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

Avec les systèmes de matchmaking qui ont activé le remplissage automatique, n'assouplissez pas trop rapidement vos exigences en matière de nombre de joueurs. Quelques secondes peuvent être nécessaire pour que la nouvelle session de jeu démarre et commence le remplissage automatique. Une meilleure approche consiste à démarrer votre extension une fois que le remblayage automatique a tendance à démarrer pour vos jeux. Le calendrier des extensions varie en fonction de la composition de votre équipe. Faites donc des tests pour trouver la meilleure stratégie d'extension pour votre jeu.

Concevez un ensemble FlexMatch de règles pour les matchs importants

Si votre ensemble de règles crée des matchs pouvant accueillir de 41 à 200 joueurs, vous devez apporter quelques modifications à la configuration de votre ensemble de règles. Ces ajustements optimisent l'algorithme de match afin qu'il puisse créer des matchs de grande envergure viables tout en réduisant les temps d'attente des joueurs. Par conséquent, les ensembles de règles de correspondance volumineux remplacent les règles personnalisées fastidieuses par des solutions standard optimisées pour les priorités de matchmaking courantes.

Voici comment déterminer si vous devez optimiser votre ensemble de règles pour les correspondances importantes :

1. Pour chaque équipe définie dans votre ensemble de règles, obtenez la valeur de MaxPlayer,
2. Additionnez toutes les valeurs de MaxPlayer. Si le total est supérieur à 40, vous disposez d'un ensemble de règles de correspondance volumineux.

Pour optimiser votre ensemble de règles pour les matchs importants, effectuez les ajustements décrits ci-dessous. Consultez le schéma d'un ensemble de règles de correspondance volumineux dans Schéma d'ensemble de règles pour les matchs importants et des exemples d'ensembles de règles dansExemple 7 : créer une grande correspondance.

Personnalisez l'algorithme de correspondance pour les correspondances importantes

Ajoutez un composant d'algorithme à l'ensemble de règles, s'il n'en existe pas déjà un. Définissez les propriétés suivantes.

- `strategy`(obligatoire) — Définissez la `strategy` propriété sur « équilibré ». Ce paramètre déclenche FlexMatch des vérifications supplémentaires après le match pour trouver l'équilibre optimal de l'équipe en fonction d'un attribut de joueur spécifié, qui est défini dans la `balancedAttribute` propriété. La stratégie équilibrée remplace le besoin de règles personnalisées pour créer des équipes équilibrées.
- `balancedAttribute`(obligatoire) — Identifiez un attribut de joueur à utiliser pour équilibrer les équipes lors d'un match. Cet attribut doit avoir un type de données numérique (double ou entier). Par exemple, si vous choisissez de mettre l'accent sur les compétences des joueurs, FlexMatch essayez de répartir les joueurs de manière à ce que toutes les équipes aient des niveaux de compétence agrégés aussi égaux que possible. L'attribut d'équilibrage doit être déclaré dans les attributs du joueur de l'ensemble de règles.

- `batchingPreference`(facultatif) — Choisissez dans quelle mesure vous souhaitez mettre l'accent sur la formation de matchs avec la plus faible latence possible pour vos joueurs. Ce paramètre affecte la façon dont les tickets de match sont triés avant de créer des matchs. Les options incluent :
  - La plus grande population. FlexMatchautorise les matchs en utilisant tous les tickets du pool présentant des valeurs de latence acceptables à au moins un endroit en commun. Par conséquent, le pool de billets potentiel a tendance à être important, ce qui permet de remplir les matchs plus rapidement. Les joueurs peuvent être placés dans des jeux avec une latence acceptable, mais pas toujours optimale. Si la `batchingPreference` propriété n'est pas définie, il s'agit du comportement par défaut lorsqu'elle `strategy` est définie sur « équilibré ».
  - Emplacement le plus rapide. FlexMatchtrie à l'avance tous les tickets du pool en fonction de l'endroit où ils indiquent les valeurs de latence les plus faibles. Par conséquent, les matchs ont tendance à se former avec des joueurs qui signalent une faible latence aux mêmes endroits. Dans le même temps, le nombre de tickets potentiels pour chaque match est réduit, ce qui peut augmenter le temps nécessaire pour remplir un match. De plus, étant donné que la priorité est accordée à la latence, les joueurs peuvent varier plus largement en ce qui concerne l'attribut d'équilibrage lors des matchs.

L'exemple suivant configure l'algorithme de match pour qu'il se comporte comme suit : (1) Triez à l'avance le pool de tickets pour regrouper les tickets par emplacement où ils présentent des valeurs de latence acceptables ; (2) Formez des lots de tickets triés pour les faire correspondre ; (3) Créez des matchs avec des tickets par lots et équilibrez les équipes pour égaliser les compétences moyennes des joueurs.

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

Déclarer les attributs du joueur

Assurez-vous de déclarer l'attribut du joueur qui est utilisé comme attribut d'équilibrage dans l'algorithme de l'ensemble de règles. Cet attribut doit être inclus pour chaque joueur dans une demande de matchmaking. Vous pouvez fournir une valeur par défaut pour l'attribut du joueur, mais l'équilibrage des attributs fonctionne mieux lorsque des valeurs spécifiques au joueur sont fournies.

Définissez les équipes

Le processus de définition de la taille et de la structure des équipes est le même que pour les parties de petite envergure, mais la façon dont FlexMatch remplit les équipes est différente. Cela affecte l'apparence des correspondances lorsqu'elles ne sont que partiellement remplies. Vous souhaiterez peut-être ajuster la taille minimale de votre équipe en conséquence.

FlexMatch utilise les règles suivantes lors de l'affectation d'un joueur à une équipe. Premièrement, il recherche les équipes qui ne comptent pas encore le nombre minimal de joueurs requis. Deuxièmement, parmi ces équipes, il identifie celle qui a le moins de joueurs.

Pour les parties définissant des équipes de taille égale, les joueurs sont ajoutés de manière séquentielle pour chaque équipe jusqu'à ce qu'elles soient pleines. Par conséquent, les équipes participant à un match ont toujours un nombre presque égal de joueurs, même lorsque le match n'est pas complet. Actuellement, il n'est pas possible d'imposer des équipes de taille égale dans les partie à grande échelle. Pour les parties dont les tailles d'équipes sont asymétriques, le processus est un peu plus complexe. Dans ce scénario, les joueurs sont initialement affectés aux équipes les plus importantes ayant le plus de places libres. Au fur et à mesure que le nombre de places libres est réparti de manière plus uniforme entre toutes les équipes, les joueurs sont répartis dans les équipes les plus petites.

Supposons, par exemple, que vous ayez un ensemble de règles avec trois équipes. Les équipes rouge et bleue sont toutes deux réglées sur `maxPlayers` =10, `minPlayers` =5. L'équipe verte est réglée sur `maxPlayers` =3, `minPlayers` =2. Voici la séquence de remplissage :

1. Aucune équipe n'y est parvenue`minPlayers`. Les équipes rouge et bleue ont 10 emplacements ouverts, tandis que l'équipe verte en compte 3. Les 10 premiers joueurs sont affectés aux équipes rouge et bleue, à hauteur de 5 joeurs par équipe. Les deux équipes ont désormais atteint le but`minPlayers`.
2. L'équipe verte n'est pas encore arrivée`minPlayers`. Les 2 prochains joueurs sont attribués à l'équipe verte. L'équipe des Verts est maintenant arrivée`minPlayers`.
3. Toutes les équipes étant présentes`minPlayers`, des joueurs supplémentaires sont désormais affectés en fonction du nombre de places disponibles. Les équipes rouge et bleue ont chacune 5 places libres, tandis que l'équipe verte en a une. Les 8 joueurs suivants sont affectés (4 chacun) aux équipes rouge et bleue. Toutes les équipes ont désormais 1 place libre.
4. Les emplacements de 3 joueurs restants sont attribués (1 par équipe) sans ordre particulier.

Établissez des règles pour les grands matchs

Le matchmaking pour les matchs importants repose principalement sur la stratégie d'équilibrage et les optimisations du traitement par lots de latence. la plupart des règles personnalisées ne sont pas disponibles. Vous pouvez toutefois intégrer les types de règles suivants :

- Règle qui fixe une limite stricte à la latence des joueurs. Utilisez le type de `latency` règle avec la propriété`maxLatency`. Voir la Règle de latence référence. Voici un exemple qui définit une latence maximum de 200 millisecondes pour les joeurs :

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

- Règle consistant à regrouper les joueurs en fonction de leur proximité dans un attribut de joueur spécifié. Cela est différent de la définition d'un attribut d'équilibrage dans le cadre de l'algorithme de grande correspondance, qui vise à créer des équipes équilibrées. Cette règle regroupe les tickets de matchmaking en fonction de la similitude des valeurs des attributs spécifiés, telles que le niveau de débutant ou d'expert, ce qui a tendance à conduire à des matchs entre joueurs étroitement liés à l'attribut spécifié. Utilisez le type de `batchDistance` règle, identifiez un attribut numérique et spécifiez la plage la plus large à autoriser. Voir la Règle de distance par lots référence. Voici un exemple qui demande que les joueurs d'un match se situent à moins d'un niveau de compétence l'un de l'autre :

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

Assouplir les exigences relatives aux matchs

Comme pour les parties de petite envergure, vous pouvez utiliser des extensions pour assouplir progressivement les exigences de mise en relation lorsqu'aucune partie ne correspond. Pour les matchs de grande envergure, vous avez la possibilité d'assouplir les règles de latence ou le nombre de joueurs de l'équipe.

Si vous utilisez le remblayage automatique pour les matchs importants, évitez de relâcher le nombre de joueurs de votre équipe trop rapidement. FlexMatchcommence à générer des demandes de remplissage uniquement après le début d'une session de jeu, ce qui peut ne pas se produire plusieurs secondes après la création d'un match. Parallèlement, FlexMatch permet de créer plusieurs sessions de jeu partiellement remplies, notamment lorsque les règles liées au nombre de joueurs sont abaissées. Dans ce cas, vous pouvez vous retrouver avec plus de sessions de jeu que nécessaire et une trop faible répartition des joueurs entre eux. Une bonne pratique consiste ici à attribuer à la première étape de l'extension du nombre de joueurs un délai d'attente suffisamment long pour que la session de jeu puisse démarrer. Dans la mesure où les demandes de remplissage sont prioritaires dans les parties à grande échelle, les joueurs entrants sont placés dans les jeux existants avant le démarrage de nouveaux jeux. Vous devrez peut-être effectuer plusieurs tests afin de déterminer le temps d'attente idéal pour votre jeu.

Voici un exemple qui réduit progressivement le nombre de joueurs de l'équipe jaune, avec un temps d'attente initial plus long. Gardez à l'esprit que les délais d'attente des extensions dans les ensembles de règles sont des valeurs absolues, et non composées. Par conséquent, la première extension se produit à cinq secondes, tandis que la seconde extension se produit cinq secondes plus tard, à savoir au bout de dix secondes.

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

Créez des ensembles de règles de matchmaking

Avant de créer un ensemble de règles de matchmaking pour votre partenaire Amazon GameLift FlexMatch , nous vous recommandons de vérifier la syntaxe du jeu de règles. Une fois que vous avez créé un ensemble de règles à l'aide de la GameLift console Amazon ou du AWS Command Line Interface (AWS CLI), vous ne pouvez pas le modifier.

Notez qu'il existe un quota de service pour le nombre maximum d'ensembles de règles que vous pouvez avoir dans une AWS région. Il est donc conseillé de supprimer les ensembles de règles non utilisés.

Voir aussi

- Conception d'un ensemble de FlexMatch règles
- FlexMatch exemples d'ensembles de règles
- FlexMatchlangage des règles

Console

Création d'un ensemble de règles

1. Ouvrez la GameLift console Amazon à l'adresse https://console.aws.amazon.com/gamelift/.
2. Passez à la AWS région dans laquelle vous souhaitez créer votre ensemble de règles. Définissez des ensembles de règles dans la même région que la configuration de matchmaking qui les utilise.
3. Dans le volet de navigation, choisissez FlexMatch, Ensembles de règles de matchmaking.
4. Sur la page Ensembles de règles de matchmaking, choisissez Créer un ensemble de règles.
5. Sur la page Créer un ensemble de règles de matchmaking, procédez comme suit :

   a. Dans Paramètres de l'ensemble de règles, dans Nom, entrez un nom descriptif unique que vous pouvez utiliser pour l'identifier dans une liste ou dans des tableaux d'événements et de statistiques.

   b. Pour Ensemble de règles, entrez votre ensemble de règles en JSON. Pour plus d'informations sur la conception d'un ensemble de règles, consultezConception d'un ensemble de FlexMatch règles. Vous pouvez également utiliser l'un des exemples d'ensembles de règles deFlexMatch exemples d'ensembles de règles.

   c. Choisissez Valider pour vérifier que la syntaxe de votre ensemble de règles est correcte. Vous ne pouvez pas modifier les ensembles de règles une fois qu'ils ont été créés. Il est donc conseillé de les valider au préalable.

   d. (Facultatif) Sous Balises, ajoutez des balises pour vous aider à gérer et à suivre vos AWS ressources.

6. Choisissez Créer. Si la création est réussie, vous pouvez utiliser l'ensemble de règles avec un entremetteur.

AWS CLI

Création d'un ensemble de règles

Ouvrez une fenêtre de ligne de commande et utilisez la commande create-matchmaking-rule-set.

Cet exemple de commande crée un ensemble de règles de matchmaking simple qui permet de configurer une seule équipe. Assurez-vous de créer l'ensemble de règles dans la même AWS région que les configurations de matchmaking qui l'utilisent.

```
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 demande de création aboutit, Amazon GameLift renvoie un MatchmakingRuleSetobjet contenant les paramètres que vous avez spécifiés. Un entremetteur peut désormais utiliser le nouvel ensemble de règles.

Console

Supprimer un jeu de règles

1. Ouvrez la GameLift console Amazon à l'adresse https://console.aws.amazon.com/gamelift/.
2. Basculez vers la région dans laquelle vous avez créé l'ensemble de règles.
3. Dans le volet de navigation, choisissez FlexMatch, Ensembles de règles de matchmaking.
4. Sur la page Ensembles de règles de matchmaking, sélectionnez l'ensemble de règles que vous souhaitez supprimer, puis choisissez Supprimer.
5. Dans la boîte de dialogue Supprimer l'ensemble de règles, choisissez Supprimer pour confirmer la suppression.

> **Note**
>
> Si une configuration de matchmaking utilise l'ensemble de règles, Amazon GameLift affiche un message d'erreur (Impossible de supprimer l'ensemble de règles). Si cela se produit, modifiez la configuration de matchmaking pour utiliser un ensemble de règles différent, puis réessayez. Pour savoir quelles configurations de matchmaking

utilisent un ensemble de règles, choisissez le nom d'un ensemble de règles pour afficher sa page de détails.

AWS CLI

Supprimer un ensemble de règles

Ouvrez une fenêtre de ligne de commande et utilisez la commande delete-matchmaking-rule-setpour supprimer un ensemble de règles de matchmaking.

Si une configuration de matchmaking utilise l'ensemble de règles, Amazon GameLift renvoie un message d'erreur. Si cela se produit, modifiez la configuration de matchmaking pour utiliser un ensemble de règles différent, puis réessayez. Pour obtenir une liste des configurations de matchmaking utilisant un ensemble de règles, utilisez la commande describe-matchmaking-configurationset spécifiez le nom de l'ensemble de règles.

Cet exemple de commande vérifie l'utilisation de l'ensemble de règles de matchmaking, puis le supprime.

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

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

FlexMatch exemples d'ensembles de règles

FlexMatch les ensembles de règles peuvent couvrir une variété de scénarios de matchmaking. Les exemples suivants sont conformes à la structure FlexMatch de configuration et au langage d'expression des propriétés. Copiez entièrement ces ensembles de règles ou choisissez des éléments en fonction de vos besoins.

Pour plus d'informations sur l'utilisation FlexMatch des règles et des ensembles de règles, consultez les rubriques suivantes :

- Création d'un ensemble de FlexMatch règles
- Conception d'un ensemble de FlexMatch règles

- FlexMatchschéma d'ensemble de règles
- FlexMatchlangage des règles

> **Note**
>
> Lorsque vous évaluez une demande de mise en relation incluant plusieurs joueurs, tous les joueurs de la demande doivent satisfaire aux exigences de mise en relation.

Exemple 1 : créer deux équipes avec des joueurs égaux

Cet exemple illustre la configuration de deux équipes ayant un nombre identique de joueurs mis en relation avec les instructions suivantes.

- Créez deux équipes de joueurs.
  - Inclure entre quatre et huit joueurs dans chaque équipe.
  - Au final, les équipes doivent avoir le même nombre de joueurs.
- Inclure un niveau de compétence pour les joueurs (si aucune valeur n'est fournie, la valeur par défaut est 10).
- Choisissez les joueurs en fonction de leur niveau de compétences, qui doit être similaire à celui d'autres joueurs. Assurez-vous que le niveau moyen de compétence des deux équipes diffère de 10 points maximum.
- Si la mise en relation n'est pas effectuée rapidement, assouplissez les exigences du niveau de compétences pour que la mise en relation soit réalisée dans un délai raisonnable.
  - Après 5 secondes, étendez la recherche pour autoriser un écart moyen de compétences des joueurs de 50 points maximum.
  - Après 15 secondes, étendez la recherche pour autoriser un écart moyen de compétences des joueurs de 100 points maximum.

Remarques sur l'utilisation de cet ensemble de règles :

- Cet exemple permet de disposer d'équipes composées de 4 à 8 joueurs (même si elles doivent être de même taille). Pour les équipes offrant plusieurs tailles valides, le matchmaker fait de son mieux pour mettre en relation le nombre maximal de joueurs autorisés.

- La règle `FairTeamSkill` garantit que les équipes sont mises en relation de façon uniforme en fonction de la compétence des joueurs. Pour évaluer cette règle pour chaque nouveau joueur potentiel, FlexMatch ajoute provisoirement le joueur à une équipe et calcule les moyennes. Si la règle échoue, le joueur potentiel n'est pas ajouté à la mise en relation.
- Étant donné que les deux équipes ont des structures identiques, vous pouvez choisir de créer une seule définition d'équipe et de définir la quantité de l'équipe sur « 2 ». Dans ce scénario, si vous avez nommé l'équipe « extra-terrestres », vos équipes recevront les noms « extra-terrestres_1" et « extra-terrestres_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
        }]
    }]
}
```

Exemple 2 : créer des équipes inégales (chasseurs contre monstres)

Cet exemple décrit un mode de jeu dans lequel un groupe de joueurs chasse un seul monstre. Les joueurs choisissent un rôle de chasseur ou un rôle de monstre. Les chasseurs spécifient le niveau minimal de compétence du monstre qu'ils souhaitent affronter. La taille minimale de l'équipe de chasseurs peut être assouplie au fil du temps pour réaliser la mise en relation. Ce scénario établit les instructions suivantes :

- Créez une équipe comportant exactement cinq chasseurs.
- Créez une équipe distincte comportant un seul monstre.
- Définissez les attributs suivants pour les joueurs :
  - Un niveau de compétence pour les joueurs (si aucune valeur n'est fournie, la valeur par défaut est 10).
  - Un niveau de compétence souhaité pour le monstre (si aucune valeur n'est fournie, la valeur par défaut est 10).
  - Si le joueur souhaite être le monstre (si cette valeur n'est pas renseignée, la valeur par défaut est 0 ou false).
- Choisissez le joueur qui sera le monstre en fonction des critères suivants :
  - Le joueur doit demander le rôle de monstre.

    - Le niveau de compétence du joueur doit correspondre ou être supérieur au niveau de compétence le plus élevé souhaité par les joueurs qui ont déjà intégré l'équipe de chasseurs.
- Choisissez les joueurs de l'équipe de chasseurs en fonction des critères suivants :
    - Les joueurs qui demandent le rôle de monstre ne peuvent pas être intégrés à l'équipe de chasseurs.
    - Si le rôle de monstre est déjà attribué, le joueur doit vouloir un niveau de compétence pour le monstre qui soit inférieur à la compétence du monstre proposé.
- Si la mise en relation n'est pas effectuée rapidement, assouplissez les critères de taille minimale de l'équipe des chasseurs de la manière suivante :
    - Après 30 secondes, autorisez le démarrage du jeu avec seulement quatre joueurs dans l'équipe des chasseurs.
    - Après 60 secondes, autorisez le démarrage du jeu avec seulement trois joueurs dans l'équipe des chasseurs.

Remarques sur l'utilisation de cet ensemble de règles :

- En utilisant deux équipes distinctes pour les chasseurs et le monstre, vous pouvez évaluer les membres en fonction de différents ensembles de critères.

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

Exemple 3 : définir les exigences et les limites de latence au niveau de l'équipe

Cet exemple illustre la configuration d’équipes de joueurs et l'application d'un ensemble de règles à chacune d’elles plutôt qu'à chaque joueur. Il utilise une seule définition pour créer trois équipes de taille égale. Il établit également une latence maximale pour tous les joueurs. Les valeurs maximales de latence peuvent être assouplies au fil du temps pour réaliser la mise en relation. Cet exemple présente les instructions suivantes :

- Créez trois équipes de joueurs.
  - Inclure entre trois et cinq joueurs dans chaque équipe.
  - Au final, les équipes doivent avoir le même nombre de joueurs ou un joueur d'écart.
- Définissez les attributs suivants pour les joueurs :
  - Un niveau de compétence pour les joueurs (si aucune valeur n'est fournie, la valeur par défaut est 10).
  - Un rôle pour chaque joueur (si aucune valeur n'est fournie, la valeur par défaut est « paysan »).
- Choisissez les joueurs en fonction de leur niveau de compétences, qui doit être similaire à celui d'autres joueurs de la relation.
  - Assurez-vous que le niveau moyen de compétence de chaque équipe diffère de 10 points maximum.
- Limitez le nombre de personnages « médecin » dans chaque équipe :
  - Le nombre maximum de médecins d'une relation complète est 5.
- Seuls les joueurs dont la latence est de 50 millisecondes ou moins doivent être pris en compte.
- Si la mise en relation n'est pas effectuée rapidement, assouplissez le critère de latence pour les joueurs de la manière suivante :
  - Après 10 secondes, autoriser les valeurs de latence pour les joueurs jusqu'à 100 ms.
  - Après 20 secondes, autoriser les valeurs de latence pour les joueurs jusqu'à 150 ms.

Remarques sur l'utilisation de cet ensemble de règles :

- L'ensemble de règles garantit que les équipes sont mises en relation de façon uniforme en fonction de la compétence des joueurs. Pour évaluer la `FairTeamSkill` règle, ajoute FlexMatch provisoirement le joueur potentiel à une équipe et calcule les compétences moyennes des joueurs de l'équipe. Ensuite, il le compare au niveau de compétence moyen des joueurs dans les deux équipes. Si la règle échoue, le joueur potentiel n'est pas ajouté à la mise en relation.

- Pour respecter les exigences au niveau de l'équipe et de la mise en relation (nombre total de médecins) un ensemble de règles est utilisé. Ce type de règle vérifie la liste des attributs de personnages pour tous les joueurs par rapport au nombre maximum. Utilisez `flatten` pour créer une liste pour tous les joueurs dans toutes les équipes.
- Lors de l'évaluation basée sur la latence, notez les éléments suivants :
  - Les données de latence sont fournies dans la demande de mise en relation dans le carte de l'objet Player (joueur). Il ne s'agit pas d'un attribut de joueur, il n'a donc pas besoin d'être répertorié en tant que tel.
  - Le matchmaker évalue la latence par région. Toute région dont la latence est plus élevée que la latence maximale est ignorée. Pour être accepté pour une mise en relation, un joueur doit avoir au moins une région avec une latence inférieure au maximum.
  - Si une demande de mise en relation omet les données de latence pour un ou plusieurs joueurs, la demande est rejetée pour toutes les relations.

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

Exemple 4 : utiliser un tri explicite pour trouver les meilleures correspondances

Cet exemple montre la configuration d'une mise en relation simple avec deux équipes de trois joueurs. Il montre comment utiliser les règles de tri explicite pour aider à trouver les meilleures mises en relation possibles aussi rapidement que possible. Ces règles trient tous les tickets de matchmaking actifs pour créer les meilleurs matchs en fonction de certaines exigences clés. Cet exemple est implémenté avec les instructions suivantes :

- Créez deux équipes de joueurs.
- Inclure exactement trois joueurs dans chaque équipe.
- Définissez les attributs suivants pour les joueurs :
  - Niveau d'expérience (si aucune valeur n'est fournie, la valeur par défaut est 50).
  - Modes de jeu préférés (peut contenir plusieurs valeurs) (si aucune valeur n'est fournie, les valeurs par défaut sont « coop » et « deathmatch »).
  - Relations de jeu préférées, indiquant le nom de la mise en relation et la pondération des préférences (si aucune valeur n'est fournie, la valeur par défaut est `"defaultMap"` avec une pondération de 100).
- Configurez le tri préalable :
  - Triez les joueurs en fonction de leurs préférences pour la même relation de jeu en tant que joueur d'ancrage. Les joueurs peuvent avoir plusieurs relations de jeu préférées, c'est pourquoi cet exemple utilise une valeur de préférence.
  - Triez les joueurs en fonction du degré de correspondance de leur niveau d'expérience avec celui du joueur d'ancrage. Ce tri permet de rapprocher le mieux possible les niveaux d'expérience de tous les joueurs de toutes les équipes.
- Tous les joueurs dans toutes les équipes doivent avoir sélectionné au moins un mode de jeu en commun.
- Tous les joueurs dans toutes les équipes doivent avoir sélectionné au moins une relation de jeu en commun.

Remarques sur l'utilisation de cet ensemble de règles :

- Le tri de la relation de jeu utilise un tri absolu qui compare la valeur d'attribut mapPreference. Comme il s’agit de la première valeur dans l'ensemble de règles, ce tri est effectuée en premier.
- Le tri utilise un paramètre de distance pour comparer le niveau de compétence d'un joueur potentiel par rapport à celui du joueur d'ancrage.

- Les tris sont exécutés dans l'ordre dans lequel ils sont répertoriés dans un ensemble de règles. Dans ce scénario, les joueurs sont triés par préférence de relation de jeu, puis par niveau d'expérience.

```
{
    "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
    }]
}
```

Exemple 5 : rechercher des intersections entre plusieurs attributs de joueur

Cet exemple illustre comment utiliser une règle de collection pour rechercher des intersections dans deux attributs de joueur ou plus. Lorsque vous utilisez des collections, vous pouvez utiliser l'opération `intersection` pour un attribut unique, et l'opération `reference_intersection_count` pour plusieurs attributs.

Pour illustrer cette approche, cet exemple évalue les joueurs d'une mise en relation en fonction de leurs préférences de personnages. L'exemple de jeu est un style free-for-all « » dans lequel tous les joueurs d'un match sont des adversaires. Chaque joueur est invité à (1) choisir un personnage pour lui-même, et (2) à choisir les personnages contre lesquels il veut jouer. Nous avons besoin d'une règle qui garantit que chaque joueur d'une mise en relation utilise un personnage figurant sur la liste des adversaires préférés de tous les autres joueurs.

L'exemple d'ensemble de règles décrit une mise en relation avec les caractéristiques suivantes :

- Structure de l'équipe : une équipe de cinq joueurs
- Attributs du joueur :
  - myCharacter : personnage choisi par le joueur.
  - preferredOpponents : liste des personnages contre lesquels le joueur veut jouer.
- Règles de mise en relation : une mise en relation potentielle est acceptable si chaque personnage en cours d'utilisation figure sur la liste des adversaires préférés de chaque joueur.

Pour implémenter la règle de mise en relation, cet exemple utilise une règle de collection avec les valeurs de propriété suivantes :

- Opération — Utilise une `reference_intersection_count` opération pour évaluer la manière dont chaque liste de chaînes de la valeur de mesure intersecte la liste de chaînes de la valeur de référence.
- Mesure — Utilise l'expression de `flatten` propriété pour créer une liste de listes de chaînes, chaque liste de chaînes contenant la valeur d'attribut MyCharacter d'un joueur.
- Valeur de référence — Utilise l'expression de *`set_intersection`* propriété pour créer une liste de chaînes contenant toutes les valeurs d'attribut PreferredAdversaires communes à tous les joueurs du match.
- Restrictions : `minCount` est définie sur 1 pour garantir que le personnage choisi par chaque joueur (une liste de chaînes dans la mesure) correspond à au moins un des adversaires préférés communs à tous les joueurs. (une chaîne dans la valeur de référence).
- Expansion — Si une correspondance n'est pas remplie dans les 15 secondes, assouplissez l'exigence minimale d'intersection.

Le processus pour cette règle se présente comme suit :

1. Un joueur est ajouté à la mise en relation potentielle. La valeur de référence (une liste de chaînes) est recalculée afin d'inclure les intersections avec la liste des adversaires préférés du nouveau joueur. La valeur de mesure (une liste de listes de chaînes) est recalculée pour ajouter le caractère choisi par le nouveau joueur en tant que nouvelle liste de chaînes.
2. Amazon GameLift vérifie que chaque liste de chaînes figurant dans la valeur de mesure (les caractères choisis par les joueurs) croise au moins une chaîne dans la valeur de référence (les adversaires préférés des joueurs). Comme dans cet exemple, chaque liste de chaînes de la mesure ne contient qu'une seule valeur, l'intersection est 0 ou 1.
3. Si aucune liste de chaînes de la mesure n'interagit avec la liste de chaînes de la valeur de référence, la règle échoue et le nouveau joueur est supprimé de la mise en relation potentielle.
4. Si une mise en relation n'est pas remplie dans un délai de 15 secondes, abandonnez l'exigence de mise en relation de l'adversaire afin de remplir les emplacements restants du joueur dans la mise en relation.

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

Exemple 6 : comparer les attributs de tous les joueurs

Cet exemple illustre la façon de comparer les attributs d'un joueur à un groupe de joueurs.

L'exemple d'ensemble de règles décrit une mise en relation avec les caractéristiques suivantes :

- Structure des équipes : deux équipes à un seul joueur
- Attributs du joueur :
  - gameMode : type de jeu choisi par le joueur (si aucune valeur n'est fournie, la valeur par défaut est « turn-based »).
  - gameMap : monde du jeu choisi par le joueur (si aucune valeur n'est fournie, la valeur par défaut est 1).
  - character : personnage choisi par le joueur (l'absence de valeur par défaut signifie que les joueurs doivent indiquer un personnage).
- Règles de mise en relation : les joueurs mis en relation doivent répondre aux exigences suivantes :
  - Les joueurs doivent choisir le même mode de jeu.
  - Les joueurs doivent choisir la même relation de jeu.
  - Les joueurs doivent choisir des personnages différents.

Remarques sur l'utilisation de cet ensemble de règles :

- Pour implémenter la règle de mise en relation, cet exemple utilise des règles de comparaison permettant de vérifier les valeurs d'attribut de tous les joueurs. Pour le mode et la relation de jeu, la règle vérifie que les valeurs sont identiques. Pour le personnage, la règle vérifie que les valeurs sont différentes.

- Cet exemple utilise une seule définition de joueur avec une propriété de quantité pour créer les deux équipes de joueurs. Les équipes se voient attribuer les noms suivants : « joueur_1 » et « joueur_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])"]
    }]
}
```

Exemple 7 : créer une grande correspondance

Cet exemple illustre comment configurer un ensemble de règles pour une partie pouvant impliquer plus de 40 joueurs. Lorsqu'un ensemble de règles décrit les équipes avec une valeur maxPlayer totale supérieure à 40, on parle de partie à grande échelle. Pour en savoir plus, voir Concevez un ensemble FlexMatch de règles pour les matchs importants.

Cet exemple d'ensemble de règles crée une partie avec les instructions suivantes :

- Créer une équipe pouvant compter jusqu'à 200 joueurs, avec un minimum de 175 joueurs.
- Critères d'équilibrage : sélectionner les joueurs en fonction d'un niveau de compétence similaire. Tous les joueurs doivent indiquer leur niveau de compétence pour pouvoir être mis en relation.
- Préférence de traitement par lots : regrouper les joueurs en fonction de critères d'équilibrage similaires lors de la création des parties.
- Règles de latence : définir une latence maximum acceptable de 150 millisecondes pour les joueurs.
- Si la mise en relation n'est pas effectuée rapidement, assouplissez les exigences pour que la partie puisse se remplir dans un délai raisonnable.
  - Après 10 secondes, accepter une équipe avec 150 joueurs.
  - Après 12 secondes, augmenter la latence maximum acceptable en la faisant passer à 200 millisecondes.
  - Après 15 secondes, accepter une équipe avec 100 joueurs.

Remarques sur l'utilisation de cet ensemble de règles :

- Étant donné que l'algorithme utilise la préférence de traitement par lots largestPopulation, les joueurs sont d'abord triés en fonction des critères d'équilibrage. Par conséquent, les parties ont tendance à accueillir plus de joueurs aux compétences similaires. Tous les joueurs répondent aux exigences de latence acceptables, mais n'ont pas forcément la meilleure latence possible pour leur emplacement.

- La stratégie d'algorithme utilisée dans cet ensemble de règles, « largest population », est le paramètre par défaut. Pour utiliser la valeur par défaut, vous pouvez choisir d'omettre le paramètre.
- Si vous avez activé le remplissage des parties, patientez un peu avant d'assouplir les exigences relatives au nombre de joueurs. Dans le cas contraire, vous risquez de vous retrouver avec un trop grand nombre de sessions de jeu partiellement remplies. Pour en savoir plus, voir Assouplir les exigences relatives aux matchs.

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

Exemple 8 : créer un grand match entre plusieurs équipes

Cet exemple illustre comment configurer un ensemble de règles pour une partie entre plusieurs équipes pouvant impliquer plus de 40 joueurs. Il indique comment créer plusieurs équipes identiques avec une seule définition et présente comment les équipes de taille asymétrique sont remplies au cours de la création d'une partie.

Cet exemple d'ensemble de règles crée une partie avec les instructions suivantes :

- Créer dix équipes « chasseur » identiques avec jusqu'à 15ˆjoueurs, et une équipe « monstre » avec exactement 5 joueurs.
- Critères d'équilibrage : sélectionner les joueurs en fonction du nombre de monstres tués. Si les joueurs ne rapportent pas leur nombre de monstres tués, utiliser la valeur par défaut 5.
- Préférence de traitement par lots : regrouper les joueurs en fonction des régions où la latence est la plus rapide possible pour les joueurs.
- Règle de latence : définir une latence maximum acceptable de 200 millisecondes pour les joueurs.
- Si la mise en relation n'est pas effectuée rapidement, assouplissez les exigences pour que la partie puisse se remplir dans un délai raisonnable.
  - Après 15 secondes, accepter les équipes avec 10 joueurs.
  - Après 20 secondes, accepter les équipes avec 8 joueurs.

Remarques sur l'utilisation de cet ensemble de règles :

- Cet ensemble de règles définit les équipes pouvant contenir jusqu'à 155 joueurs, ce qui en fait un match important. (10 x 15 chasseurs+5 monstres = 155)
- Étant donné que l'algorithme utilise la préférence de traitement par lots fastestRegion, les joueurs ont tendance à être placés dans les régions où ils signalent une latence plus rapide et non dans celles où ils signalent une latence élevée (mais acceptable). Parallèlement, les parties sont susceptibles d'avoir moins de joueurs, et les critères d'équilibrage (nombre de monstres tués) peuvent varier plus largement.

- Lorsqu'une extension est spécifiée pour une définition d'équipe (quantité > 1), elle s'applique à toutes les équipes créées avec cette définition. Par conséquent, en assouplissant le paramètre lié au nombre minimum de joueurs d'une équipe de chasseurs, les dix équipes de chasseurs sont affectées de manière égale.
- Étant donné que cet ensemble de règles est optimisé pour réduire la latence des joueurs, la règle de latence exclut tous les joueurs qui n'ont pas d'options de connexion acceptables. Il n'est pas nécessaire d'assouplir cette exigence.
- Voici comment FlexMatch remplir les correspondances pour cet ensemble de règles avant que les extensions n'entrent en vigueur :
  - Aucune équipe n'a encore atteint le nombre minimum de joueurs requis. Les équipes de chasseurs ont 15 emplacements ouverts, tandis que l'équipe Monstre compte 5 emplacements ouverts.
    - Les 100 premiers joueurs sont affectés aux dix équipes de chasseurs (10 joueurs pour chaque équipe).
    - Les 22 joueurs suivants seront affectés de manière séquentielle (2 joueurs pour chaque équipe) aux équipes de chasseurs et à l'équipe Monstre.
  - Les équipes de chasseurs ont chacune atteint le nombre minimum requis de 12 joueurs. L'équipe Monstre a 2 joueurs et n'a pas atteint le nombre minimum de joueurs requis.
    - Les trois joueurs suivants sont donc attribués cette équipe.
  - Toutes les équipes ont maintenant le nombre minimum de joueurs requis. Les équipes de chasseurs ont chacune trois emplacements ouverts. L'équipe Monstre est au complet.
    - Les 30 derniers joueurs sont affectés de manière séquentielle aux équipes de chasseurs, en s'assurant qu'elles ont toutes presque la même taille (à un joueur près).
- Si vous avez activé le remplissage des parties créées avec cet ensemble de règles, patientez un peu avant d'assouplir les exigences relatives au nombre de joueurs. Dans le cas contraire, vous risquez de vous retrouver avec un trop grand nombre de sessions de jeu partiellement remplies. Pour en savoir plus, voir Assouplir les exigences relatives aux matchs.

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

Exemple 9 : créer un grand match avec des joueurs aux attributs similaires

Cet exemple montre comment configurer un ensemble de règles pour les matchs où deux équipes utilisent`batchDistance`. Dans l'exemple :

- La `SimilarLeague` règle garantit que tous les joueurs d'un match ont `league` moins de 2 % des autres joueurs.

- La `SimilarSkill` règle garantit que tous les joueurs d'un match ont `skill` moins de 10 % des autres joueurs. Si un joueur attend 10 secondes, la distance passe à 20. Si un joueur attend depuis 20 secondes, la distance passe à 40.
- La `SameMap` règle garantit que tous les joueurs d'un match en ont fait la même demande`map`.
- La `SameMode` règle garantit que tous les joueurs d'un match en ont fait la même demande`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
        }]
    }]
}
```

Exemple 10 : Utiliser une règle composée pour créer un match avec des joueurs ayant des attributs similaires ou des sélections similaires

Cet exemple montre comment configurer un ensemble de règles pour les matchs où deux équipes utilisentcompound. Dans l'exemple :

- La `SimilarLeagueDistance` règle garantit que tous les joueurs d'un match ont `league` moins de 2 % des autres joueurs.
- La `SimilarSkillDistance` règle garantit que tous les joueurs d'un match ont `skill` moins de 10 % des autres joueurs. Si un joueur attend 10 secondes, la distance passe à 20. Si un joueur attend depuis 20 secondes, la distance passe à 40.
- La `SameMapComparison` règle garantit que tous les joueurs d'un match en ont fait la même demande`map`.
- La `SameModeComparison` règle garantit que tous les joueurs d'un match en ont fait la même demande`mode`.

- La `CompoundRuleMatchmaker` règle garantit une correspondance si au moins l'une des conditions suivantes est vraie :
  - Les joueurs participant à un match ont demandé la même chose `map` et la même chose`mode`.
  - Les joueurs d'un match ont des `league` attributs `skill` et des attributs 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
        }]
    }]
}
```

Exemple 11 : Création d'une règle utilisant la liste de blocage d'un joueur

Cet exemple illustre un ensemble de règles qui permet aux joueurs d'éviter d'être jumelés à certains autres joueurs. Les joueurs peuvent créer une liste de blocage, que le système de matchmaking évalue lors de la sélection des joueurs pour un match. Pour plus d'informations sur l'ajout d'une fonctionnalité de liste de blocage ou de liste à éviter, consultez AWSle blog sur les jeux.

Cet exemple présente les instructions suivantes :

- Créez deux équipes de cinq joueurs exactement.

- Transmettez la liste de blocage d'un joueur, qui est une liste d'identifiants de joueurs (jusqu'à 100).
- Comparez tous les joueurs à la liste de blocage de chaque joueur et rejetez un match proposé si des identifiants de joueurs bloqués sont trouvés.

Remarques sur l'utilisation de cet ensemble de règles :

- Lors de l'évaluation d'un nouveau joueur à ajouter à un match proposé (ou pour remplacer une place dans un match existant), le joueur peut être rejeté pour l'une des raisons suivantes :
  - Si le nouveau joueur figure sur la liste des joueurs déjà sélectionnés pour le match.
  - Si des joueurs déjà sélectionnés pour le match figurent sur la liste de blocage du nouveau joueur.
- Comme indiqué, cet ensemble de règles empêche de faire correspondre un joueur à un joueur figurant sur sa liste de blocage. Vous pouvez remplacer cette exigence par une préférence (également appelée liste « à éviter ») en ajoutant une extension des règles et en augmentant la `maxCount` valeur.

```
{
    "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
    }]
}
```

Créer une configuration de matchmaking

Pour configurer un système de GameLift FlexMatch matchmaking Amazon afin de traiter les demandes de matchmaking, créez une configuration de matchmaking. Utilisez la GameLift console Amazon ou le AWS Command Line Interface (AWS CLI). Pour plus d'informations sur la création d'un système de matchmaking, voirConcevez un FlexMatch entremetteur.

Créez un entremetteur pour l'hébergement Amazon GameLift

Avant de créer une configuration de matchmaking, créez un ensemble de règles et une file d'attente de sessions de GameLift jeu Amazon à utiliser avec le système de matchmaking.

Console

1. Dans la GameLiftconsole Amazon, dans le volet de navigation, choisissez Matchmaking configurations.
2. Passez à la AWS région dans laquelle vous souhaitez créer votre entremetteur.
3. Sur la page des configurations du matchmaking, choisissez Créer une configuration de matchmaking.
4. Sur la page Définir les détails de configuration, sous Détails de configuration du matchmaking, procédez comme suit :

    a. Dans Nom, entrez le nom d'un système de matchmaking qui peut vous aider à l'identifier dans une liste et dans des statistiques. Le nom du matchmaker doit être unique au sein de la région. Les demandes de matchmaking identifient le matchmaker à utiliser par son nom et sa région.

    b. (Facultatif) Dans Description, ajoutez une description pour aider à identifier le système de matchmaking.

    c. Pour Ensemble de règles, choisissez un ensemble de règles dans la liste à utiliser avec le système de matchmaking. La liste contient tous les ensembles de règles que vous avez créés dans la région actuelle.

d. Pour FlexMatchle mode, choisissez Géré pour l'hébergement GameLift géré par Amazon. Ce mode invite FlexMatch à transférer les matchs réussis vers la file d'attente de session de jeu spécifiée.

e. Pour AWSRégion, choisissez la région dans laquelle vous avez configuré la file d'attente des sessions de jeu que vous souhaitez utiliser avec le système de matchmaking.

f. Dans Queue, choisissez la file d'attente des sessions de jeu que vous souhaitez utiliser avec le système de matchmaking.

5. Choisissez Suivant.

6. Sur la page Configurer les paramètres, sous Paramètres du matchmaking, procédez comme suit :

   a. Pour le délai d'expiration des demandes, définissez le délai maximum, en secondes, pendant lequel le système de matchmaking doit terminer une correspondance pour chaque demande. FlexMatchannule les demandes de matchmaking qui dépassent ce délai.

   b. Pour le mode de remblayage, choisissez un mode de gestion des remplissages de matchs.

      - Pour activer la fonction de remplissage automatique, choisissez Automatique.
      - Pour créer votre propre gestion des demandes de remplissage ou pour ne pas utiliser la fonction de remplissage, choisissez Manuel.

   c. (Facultatif) Pour le nombre de joueurs supplémentaires, définissez le nombre de places réservées aux joueurs pendant un match. FlexMatchpourra remplir ces machines à sous avec des joueurs à l'avenir.

   d. (Facultatif) Sous Options d'acceptation des matchs, pour Acceptation requise, si vous souhaitez demander à chaque joueur participant à un match proposé d'accepter activement de participer au match, sélectionnez Obligatoire. Si vous sélectionnez cette option, alors pour le délai d'acceptation, définissez la durée, en secondes, pendant laquelle vous souhaitez que le système de matchmaking attende l'acceptation des joueurs avant d'annuler le match.

7. (Facultatif) Sous Paramètres de notification d'événements, procédez comme suit :

   a. (Facultatif) Pour la rubrique SNS, choisissez une rubrique Amazon Simple Notification Service (Amazon SNS) pour recevoir les notifications des événements de matchmaking. Si vous n'avez pas encore créé de rubrique SNS, vous pouvez la choisir ultérieurement

en modifiant la configuration du matchmaking. Pour plus d'informations, veuillez consulter Configurer les notifications FlexMatch d'événements.

b. (Facultatif) Pour les données d'événement personnalisées, entrez les données personnalisées que vous souhaitez associer à ce système de matchmaking dans la messagerie des événements. FlexMatchinclut ces données dans chaque événement associé au système de matchmaking.

8. (Facultatif) Développez les données de jeu supplémentaires, puis procédez comme suit :

   a. (Facultatif) Pour les données de session de jeu, saisissez toutes les informations supplémentaires relatives au jeu que vous souhaitez transmettre FlexMatch aux nouvelles sessions de jeu démarrées par des matchs réalisés à l'aide de cette configuration de matchmaking.

   b. (Facultatif) Pour les propriétés du jeu, ajoutez des propriétés de paire clé-valeur qui contiennent des informations sur une nouvelle session de jeu.

9. (Facultatif) Sous Balises, ajoutez des balises pour vous aider à gérer et à suivre vos AWS ressources.

10. Choisissez Suivant.

11. Sur la page Révision et création, passez en revue vos choix, puis choisissez Créer. Une fois la création réussie, le système de matchmaking est prêt à accepter les demandes de matchmaking.

AWS CLI

Pour créer une configuration de mise en relation avec l'AWS CLI, ouvrez une fenêtre de ligne de commande et utilisez la commande create-matchmaking-configuration pour définir un nouveau matchmaker.

Cet exemple de commande crée une nouvelle configuration de matchmaking qui nécessite l'acceptation du joueur et active le remplissage automatique. Il réserve également des emplacements pour deux joueurs FlexMatch pour ajouter des joueurs plus tard, et fournit certaines données de session de jeu.

```
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 demande de création d'une configuration de matchmaking aboutit, Amazon GameLift renvoie un MatchmakingConfigurationobjet avec les paramètres que vous avez demandés pour le système de matchmaking. Le nouveau système de matchmaking est prêt à accepter les demandes de matchmaking.

Créez un système de matchmaking pour une application autonome FlexMatch

Avant de créer une configuration de matchmaking, créez un ensemble de règles à utiliser avec le système de matchmaking.

Console

1. Ouvrez la GameLift console Amazon à l'adresse https://console.aws.amazon.com/gamelift/home.
2. Passez à la AWS région dans laquelle vous souhaitez créer votre entremetteur. Pour une liste des régions qui prennent en charge les configurations de FlexMatch matchmaking, voirChoisissez un lieu pour l'entremetteur.
3. Dans le volet de navigation FlexMatch, choisissez Configurations du matchmaking.
4. Sur la page des configurations du matchmaking, choisissez Créer une configuration de matchmaking.
5. Sur la page Définir les détails de configuration, sous Détails de configuration du matchmaking, procédez comme suit :

    a. Dans Nom, entrez le nom d'un système de matchmaking qui peut vous aider à l'identifier dans une liste et dans des statistiques. Le nom du matchmaker doit être unique au sein

de la région. Les demandes de matchmaking identifient le matchmaker à utiliser par son nom et sa région.

b. (Facultatif) Dans Description, ajoutez une description pour aider à identifier le système de matchmaking.

c. Pour Ensemble de règles, choisissez un ensemble de règles dans la liste à utiliser avec le système de matchmaking. La liste contient tous les ensembles de règles que vous avez créés dans la région actuelle.

d. Pour le FlexMatchmode, choisissez Autonome. Cela indique que vous disposez d'un mécanisme personnalisé pour démarrer de nouvelles sessions de jeu sur une solution d'hébergement externe à AmazonGameLift.

6. Choisissez Suivant.

7. Sur la page Configurer les paramètres, sous Paramètres du matchmaking, procédez comme suit :

a. Pour le délai d'expiration des demandes, définissez le délai maximum, en secondes, pendant lequel le système de matchmaking doit terminer une correspondance pour chaque demande. Les demandes de matchmaking qui dépassent ce délai sont rejetées.

b. (Facultatif) Sous Options d'acceptation des matchs, pour Acceptation requise, si vous souhaitez demander à chaque joueur participant à un match proposé d'accepter activement de participer au match, sélectionnez Obligatoire. Si vous sélectionnez cette option, alors pour le délai d'acceptation, définissez la durée, en secondes, pendant laquelle vous souhaitez que le système de matchmaking attende l'acceptation des joueurs avant d'annuler le match.

8. (Facultatif) Sous Paramètres de notification d'événements, procédez comme suit :

a. (Facultatif) Pour la rubrique SNS, choisissez une rubrique Amazon SNS pour recevoir les notifications des événements de matchmaking. Si vous n'avez pas encore créé de rubrique SNS, vous pouvez la choisir ultérieurement en modifiant la configuration du matchmaking. Pour plus d'informations, veuillez consulter Configurer les notifications FlexMatch d'événements.

b. (Facultatif) Pour les données d'événement personnalisées, entrez les données personnalisées que vous souhaitez associer à ce système de matchmaking dans la messagerie des événements. FlexMatchinclut ces données dans chaque événement associé au système de matchmaking.

9. (Facultatif) Sous Balises, ajoutez des balises pour vous aider à gérer et à suivre vos AWS ressources.
10. Choisissez Suivant.
11. Sur la page Révision et création, passez en revue vos choix, puis choisissez Créer. Une fois la création réussie, le système de matchmaking est prêt à accepter les demandes de matchmaking.

AWS CLI

Pour créer une configuration de mise en relation avec l'AWS CLI, ouvrez une fenêtre de ligne de commande et utilisez la commande create-matchmaking-configuration pour définir un nouveau matchmaker.

Cet exemple de commande crée une nouvelle configuration de matchmaking pour un système de matchmaking autonome qui nécessite l'acceptation du joueur.

```
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 demande de création d'une configuration de matchmaking aboutit, Amazon GameLift renvoie un MatchmakingConfigurationobjet avec les paramètres que vous avez demandés pour le système de matchmaking. Le nouveau système de matchmaking est prêt à accepter les demandes de matchmaking.

Modifier une configuration de matchmaking

Pour modifier une configuration de matchmaking, choisissez Configurations de matchmaking dans la barre de navigation et choisissez la configuration que vous souhaitez modifier. Vous pouvez mettre à jour n'importe quel champ d'une configuration existante à l'exception de son nom.

Lors de la mise à jour d'un ensemble de règles de configuration, un nouvel ensemble de règles peut être incompatible s'il existe des tickets de matchmaking actifs pour les raisons suivantes :

- Noms d'équipes ou nombre d'équipes nouveaux ou différents
- Attributs des nouveaux joueurs
- Modifications apportées aux types d'attributs de joueurs existants

Pour apporter l'une de ces modifications à votre ensemble de règles, créez une nouvelle configuration de matchmaking avec l'ensemble de règles mis à jour.

Configurer les notifications FlexMatch d'événements

Vous pouvez utiliser les notifications d'événements pour suivre l'état des demandes de matchmaking individuelles. Tous les jeux en production ou en pré-production avec une activité de matchmaking importante doivent utiliser des notifications d'événements.

Il existe deux options pour configurer les notifications d'événement.

- Demandez à votre système de matchmaking de publier des notifications d'événements dans une rubrique Amazon Simple Notification Service (Amazon SNS).
- Utilisez les EventBridge événements Amazon publiés automatiquement et sa suite d'outils pour gérer les événements.

Pour obtenir la liste des FlexMatch événements émis par GameLift Amazon, consultezFlexMatchévénements de matchmaking.

Configurez EventBridge des événements

Amazon publie GameLift automatiquement tous les événements de matchmaking sur AmazonEventBridge. AvecEventBridge, vous pouvez configurer des règles pour que les événements de matchmaking soient acheminés vers des cibles pour traitement. Par exemple, vous pouvez définir une règle pour acheminer l'événement « PotentialMatchCreated » vers une AWS Lambda fonction qui gère les acceptations des joueurs. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ?

> **Note**
>
> Lorsque vous configurez vos matchmakers, laissez le champ cible de notification vide ou faites référence à une rubrique SNS si vous souhaitez utiliser les deux EventBridge et Amazon SNS.

Configurer une rubrique Amazon SNS

Vous pouvez demander à Amazon de GameLift publier tous les événements générés par un FlexMatch système de matchmaking sur une rubrique Amazon SNS.

Pour créer une rubrique SNS pour les notifications d'GameLiftévénements Amazon

1. Ouvrez la console Amazon SNS.
2. Dans le panneau de navigation, sélectionnez Topics (Rubriques).
3. Sur la page Rubriques, choisissez Créer une rubrique.
4. Créez une rubrique dans la console . Pour plus d'informations, consultez la section Pour créer une rubrique AWS Management Console à l'aide du manuel du développeur Amazon Simple Notification Service.
5. Sur la page Détails de votre sujet, choisissez Modifier.
6. (Facultatif) Sur la page d'édition de votre sujet, développez la politique d'accès, puis ajoutez la syntaxe en gras de la déclaration de politique AWS Identity and Access Management (IAM) suivante à la fin de votre politique existante. (La politique complète est présentée ici pour plus de clarté.) Veillez à utiliser les informations Amazon Resource Name (ARN) pour votre propre rubrique SNS et pour la configuration du GameLift matchmaking Amazon.

```
{
  "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. Choisissez Save Changes (Enregistrer les modifications).

Configuration d'une rubrique SNS avec chiffrement côté serveur

Vous pouvez utiliser le chiffrement côté serveur (SSE) pour stocker des données sensibles dans des rubriques cryptées. Le chiffrement SSE protège le contenu des messages présents dans les rubriques Amazon SNS à l'aide de clés gérées dans AWS Key Management Service (AWS KMS).

Pour plus d'informations sur le chiffrement côté serveur avec Amazon SNS, consultez la section Chiffrement au repos dans le guide du développeur Amazon Simple Notification Service.

Pour configurer une rubrique SNS avec chiffrement côté serveur, consultez les rubriques suivantes :

- Création d'une clé dans le Guide du AWS Key Management Service développeur
- Activation de SSE pour une rubrique dans le guide du développeur Amazon Simple Notification Service

Lorsque vous créez votre clé KMS, utilisez la politique de clé KMS suivante :

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

Configurer un abonnement à une rubrique pour appeler une fonction Lambda

Vous pouvez invoquer une fonction Lambda à l'aide des notifications d'événements publiées dans votre rubrique Amazon SNS. Lorsque vous configurez le système de matchmaking, veillez à définir la cible de notification sur l'ARN de votre sujet SNS.

Le AWS CloudFormation modèle suivant configure un abonnement à une rubrique SNS nommée `MyFlexMatchEventTopic` pour appeler une fonction Lambda nommée. `FlexMatchEventHandlerLambdaFunction` Le modèle crée une politique d'autorisations IAM qui permet GameLift à Amazon d'écrire dans la rubrique SNS. Le modèle ajoute ensuite des autorisations permettant à la rubrique SNS d'invoquer la fonction 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
```

Préparer des jeux pour FlexMatch

Utilisez Amazon GameLift FlexMatch pour ajouter une fonctionnalité de jumelage de joueurs à vos jeux. FlexMatchest disponible avec les GameLift solutions Amazon gérées pour les serveurs de jeux personnalisés et les serveurs en temps réel.

FlexMatch associe le service de mise en relation avec un moteur de règles personnalisables. Vous pouvez ainsi concevoir la façon de mettre en relation des joueurs en fonction d'attributs et de modes de jeu adaptés à votre jeu, et vous appuyer sur FlexMatch pour gérer les détails pratiques de la formation de groupes de joueurs et de leur placement dans des jeux. Voir plus de détails sur la mise en relation personnalisée dans FlexMatch exemples d'ensembles de règles.

FlexMatch s'appuie sur la fonction des files d'attente. Une fois qu'une mise en relation est créée, FlexMatch en communique les détails à la file d'attente de votre choix. La file d'attente recherche les ressources d'hébergement disponibles sur vos GameLift flottes Amazon et lance une nouvelle session de jeu pour le match.

Les rubriques de cette section illustrent comment ajouter la prise en charge de la mise en relation à des serveurs et clients de jeux. Pour créer un matchmaker pour votre jeu, consultez Créer un entremetteur Amazon GameLift FlexMatch. Pour plus d'informations sur la manière dont FlexMatch fonctionne, consultez Comment GameLift FlexMatch fonctionne Amazon.

Ajouter FlexMatch à un client de jeu

Cette rubrique explique comment ajouter la prise en charge du FlexMatch matchmaking à vos services de jeu côté client. Le processus est essentiellement le même, que vous l'utilisiez FlexMatch avec l'hébergement GameLift géré par Amazon ou avec une autre solution d'hébergement. Pour en savoir plus sur FlexMatch et sur la marche à suivre pour configurer un matchmaker personnalisé pour vos jeux, consultez les rubriques suivantes :

- FlexMatchintégration avec l'GameLifthébergement Amazon
- Comment GameLift FlexMatch fonctionne Amazon
- Créer un entremetteur Amazon GameLift FlexMatch
- FlexMatch exemples d'ensembles de règles

Pour activer le FlexMatch matchmaking dans votre jeu, ajoutez les fonctionnalités suivantes :

- Préparez-vous à demander le matchmaking pour un ou plusieurs joueurs (obligatoire).
- Suivez l'état des demandes de matchmaking (obligatoire).
- Demander l'acceptation d'un joueur pour un match proposé (facultatif).
- Une fois qu'une session de jeu est créée pour le nouveau match, récupérez les informations de connexion du joueur et rejoignez la partie.

Préparez-vous à demander le matchmaking pour les joueurs

Nous recommandons vivement à votre client de jeu de faire des demandes de matchmaking via un service de jeu côté client. En utilisant une source de confiance, vous pouvez plus facilement vous protéger contre les tentatives de piratage et les données factices des joueurs. Si votre jeu utilise un service d'annuaire des sessions, il s'agit d'une bonne option pour traiter les demandes de mise en relation.

Pour préparer le service client, effectuez les tâches suivantes :

- Ajoutez l'GameLiftAPI Amazon. Votre service client utilise les fonctionnalités de l'GameLiftAPI Amazon, qui fait partie du AWS SDK. Consultez les GameLiftkits SDK Amazon pour les services clients pour en savoir plus sur le AWS SDK et télécharger la dernière version. Ajoutez ce kit SDK au projet de service de votre client de jeu.
- Configurez un système de tickets de matchmaking. Toutes les demandes de mise en relation doivent avoir un ID de ticket unique. Vous devez disposer d'un mécanisme permettant de générer des ID uniques et de les attribuer aux nouvelles demandes de mise en relation. Un ID de ticket peut utiliser utiliser n'importe quel format de chaîne, jusqu'à un maximum de 128 caractères.
- Procurez-vous les informations relatives au matchmaker. Obtenez le nom de la configuration de mise en relation que vous prévoyez d'utiliser. Vous avez également besoin de la liste des attributs de joueur requis du matchmaker, qui sont définis dans l'ensemble de règles du matchmaker.
- Obtenir les données des joueurs. Configurez un moyen d'obtenir les données pertinentes pour chaque joueur. Cela inclut l'ID de joueur, les valeurs d'attribut de joueur et les données de latence mises à jour pour chaque région où le joueur est susceptible d'être impliqué dans un jeu.
- Activez le remplissage des parties (facultatif). Déterminez la façon dont vous souhaitez remplir les jeux existants. Si le remplissage manuel est configuré pour vos matchmakers, il peut être utile d'ajouter à votre jeu la prise en charge du remplissage. Si le mode de remplissage est automatique, il se peut que vous deviez inclure un moyen de le désactiver pour les sessions de jeu individuelles. Pour en savoir plus sur la gestion du remplissage des parties, voire Remplir les jeux existants avec FlexMatch.

Demande de matchmaking pour les joueurs

Ajoutez du code à votre service client pour créer et gérer les demandes de mise en relation à un matchmaker FlexMatch. Le processus de demande de FlexMatch matchmaking est identique pour les jeux utilisés FlexMatch avec l'GameLiftmanagedhébergement Amazon et pour les jeux utilisés FlexMatch comme solution autonome.

Création d'une demande de mise en relation :

- Appelez l'GameLiftAPI Amazon StartMatchmaking. Chaque demande doit contenir les informations suivantes.

  Matchmaker

  Le nom de la configuration de matchmaking à utiliser pour la demande. FlexMatchplace chaque demande dans le pool pour le matchmaker spécifié, et la demande est traitée en fonction de la façon dont le matchmaker est configuré. Il s'agit notamment d'appliquer une limite de temps, que ce soit pour demander l'acceptation du joueur pour une partie, la file d'attente à utiliser lorsque vous placez une session de jeu qui en résulte, etc. Pour en savoir plus sur les matchmakers et les ensembles de règles, voir Concevez un FlexMatch entremetteur.

  ID ticket

  ID de ticket unique attribué à la demande. Tout élément lié à la demande, y compris les événements et les notifications, utilise l'ID de ticket comme référence.

  Données du joueur

  Liste des joueurs pour lesquels vous voulez créer une mise en relation. Si un des joueurs de la demande ne répond pas aux exigences de mise en relation (selon les règles de mise en relation et les minimums de latence), la demande de mise en relation n'aboutit jamais. Vous pouvez inclure jusqu'à dix joueurs dans une demande de mise en relation. Lorsqu'une demande contient plusieurs joueurs, FlexMatch tente de créer une mise en relation unique et d'affecter tous les joueurs à la même équipe (sélectionnée de façon aléatoire). Si une demande contient trop de joueurs pour constituer une équipe de mise en relation, la demande échoue. Par exemple, si vous avez configuré votre matchmaker pour créer des rencontres 2 contre 2 (deux équipes de deux joueurs), vous ne pouvez pas envoyer de demande de mise en relation contenant plus de deux joueurs.

> **Note**
>
> Un joueur (identifié par son ID de joueur) ne peut être inclus que dans une seule demande de mise en relation à la fois. Si vous créez une autre demande pour un joueur, tous les tickets de mise en relation actifs ayant le même ID de joueur sont automatiquement annulés.

Pour chaque joueur répertorié, incluez les données suivantes :

- ID de joueur — Chaque joueur doit avoir un identifiant de joueur unique, que vous générez. Consultez la section Générer des identifiants de joueurs.
- Attributs du joueur : si le système de matchmaking utilisé demande des attributs de joueur, la demande doit fournir ces attributs pour chaque joueur. Les attributs de joueur nécessaires sont définis dans l'ensemble de règles du matchmaker, qui spécifie également le type de données des attributs. Un attribut de joueur est facultatif uniquement lorsque l'ensemble de règles spécifie une valeur par défaut pour cet attribut. Si la demande de mise en relation ne fournit pas les attributs de joueur nécessaires pour tous les joueurs, elle n'aboutit pas. Pour en savoir plus sur les ensembles de règles du matchmaker et les attributs de joueur, consultez Création d'un ensemble de FlexMatch règles et FlexMatch exemples d'ensembles de règles.
- Latences des joueurs : si le système de matchmaking utilisé dispose d'une règle de latence pour les joueurs, la demande doit indiquer la latence pour chaque joueur. Les données de latence des joueurs correspondent à une liste d'une ou de plusieurs valeurs par joueur. Elles représente la latence que subit le joueur pour chaque région dans la file d'attente du matchmaker. Si aucune valeur de latence des joueurs n'est incluse dans la demande, le joueur ne peut pas être mis en relation, et la demande échoue.

Récupération des informations relatives à la demande de mise en relation :

- Une fois qu'une demande de correspondance est envoyée, vous pouvez consulter les détails de la demande en appelant DescribeMatchmakingavec le numéro de ticket associé à la demande. Cet appel renvoie les informations relatives à la demande, y compris le statut actuel. Une fois qu'une demande se termine avec succès, le ticket contient également les informations nécessaires à la connexion d'un joueur à la partie.

Annulation d'une demande de mise en relation :

- Vous pouvez annuler une demande de matchmaking à tout moment en appelant StopMatchmakingavec le numéro de ticket de la demande.

Suivez les événements de matchmaking

Configurez des notifications pour suivre les événements qu'Amazon GameLift émet pour les processus de matchmaking. Vous pouvez configurer les notifications directement, en créant une rubrique SNS ou en utilisant AmazonEventBridge. Pour plus d'informations sur la configuration des notifications, consultez Configurer les notifications FlexMatch d'événements. Une fois que vous avez configuré les notifications, vous devez ajouter un écouteur au niveau de votre service client pour détecter les événements et y répondre si nécessaire.

C'est également une bonne idée de sauvegarder les notifications en interrogeant périodiquement les mises à jour de statut lorsqu'une période importante passe sans notification. Pour minimiser l'impact sur les performances de mise en relation, assurez-vous d'interroger seulement après avoir attendu au moins 30 secondes après l'envoi du ticket de mise en relation ou après la dernière notification reçue.

Récupérez un ticket de demande de matchmaking, y compris son statut actuel, en appelant DescribeMatchmakingavec l'identifiant du ticket de la demande. Nous recommandons une interrogation toutes les 10 secondes au plus. Cette approche est destinée uniquement aux scénarios de développement à faible volume.

> **Note**
>
> Vous devez configurer votre jeu avec des notifications d'événements avant d'avoir un volume élevé d'utilisation de la mise en relation, par exemple avec les tests de charge de pré-production. Tous les jeux en version publique doivent utiliser des notifications quel que soit le volume. L'approche de sondage continu n'est appropriée que pour les jeux en développement avec une faible utilisation de la mise en relation.

Demander l'acceptation du joueur

Si vous utilisez un matchmaker pour lequel l'acceptation des joueurs est activée, ajoutez le code à votre service client pour gérer ce processus d'acceptation. Le processus de gestion des acceptations

des joueurs est identique pour les jeux utilisant FlexMatch l'hébergement GameLift géré par Amazon et pour les jeux utilisés FlexMatch comme solution autonome.

Demande d'acceptation du joueur pour une mise en relation proposée :

1. Identifier quand une partie proposée nécessite l'acceptation du joueur. Surveillez le ticket de mise en relation pour détecter lorsque l'état passe à `REQUIRES_ACCEPTANCE`. La modification de ce statut déclenche l'FlexMatchévénement`MatchmakingRequiresAcceptance`.
2. Obtenir les acceptations de tous les joueurs. Créez un mécanisme pour présenter les détails de la partie proposée à chaque joueur associé au ticket de mise en relation. Les joueurs doivent être en mesure d'indiquer qu'ils acceptent ou refusent la partie proposée. Vous pouvez récupérer les détails du match en appelant DescribeMatchmaking. Les joueurs ont un temps limité pour répondre avant que le matchmaker supprime la partie proposée.
3. Signaler les réponses des joueurs à FlexMatch. Signalez les réponses des joueurs en appelant AcceptMatchavec « Accepter » ou « Refuser ». Tous les joueurs d'une demande doivent accepter la mise en relation pour que celle-ci se poursuive.
4. Gérer les tickets avec les acceptations ayant échoué. Une demande échoue lorsqu'un joueur refuse la partie proposée ou ne parvient pas à répondre dans les limites imparties. Les billets des joueurs qui ont accepté le match sont automatiquement renvoyés à la billetterie. Les tickets pour les joueurs qui n'ont pas accepté le match passent au statut ÉCHEC et ne sont plus traités. Pour les tickets avec plusieurs joueurs, si l'un des joueurs figurant sur le ticket n'a pas accepté le match, le ticket sera annulé dans son intégralité.

Se connecter à un match

Ajoutez du code à votre service client pour gérer une correspondance correctement formée (statut `COMPLETED` ou événement`MatchmakingSucceeded`). Ce processus inclut la notification des joueurs concernés et l'envoi des informations de connexion à leurs clients de jeu.

Pour les jeux qui utilisent l'hébergement GameLift géré par Amazon, lorsqu'une demande de matchmaking est traitée avec succès, les informations de connexion à la session de jeu sont ajoutées au ticket de matchmaking. Récupérez un ticket de matchmaking terminé en appelant DescribeMatchmaking. Les informations de connexion incluent l'adresse IP et le port de la session de jeu, ainsi qu'un ID de session pour chaque joueur. Pour en savoir plus, voir GameSessionConnectionInfo. Votre client de jeu peut utiliser ces informations pour se connecter directement à la session de jeu pendant le match. La demande de connexion doit inclure un identifiant de session du joueur et un identifiant de joueur. Ces données associent le joueur

connecté aux données de match de la session de jeu, y compris les attributions des équipes (voir GameSession).

Pour les jeux qui utilisent d'autres solutions d'hébergement, notamment Amazon GameLift FleetIQ, vous devez intégrer un mécanisme permettant aux joueurs du match de se connecter à la session de jeu appropriée.

Exemples de demandes de matchmaking

Les extraits de code suivants créent des demandes de matchmaking pour plusieurs entremetteurs différents. Comme décrit, une demande doit fournir les attributs de joueur qui sont requis par le matchmaker utilisé, tel que défini dans l'ensemble de règles de ce dernier. L'attribut fourni doit utiliser le même type de données, le même numéro (N) ou la même chaîne (S) que dans l'ensemble de règles.

```
# 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)
```

Ajouter FlexMatch à un serveur de jeu GameLift hébergé sur Amazon

Cette rubrique explique comment ajouter la prise en charge du FlexMatch matchmaking aux serveurs de jeu personnalisés qui utilisent l'hébergement GameLift géré Amazon. Pour en savoir plus sur l'ajout de FlexMatch à vos jeux, consultez les rubriques suivantes :

- Comment GameLift FlexMatch fonctionne Amazon
- FlexMatchintégration avec l'GameLifthébergement Amazon

Les informations de cette rubrique supposent que vous avez intégré avec succès le SDK Amazon GameLift Server à votre projet de serveur de jeu, comme décrit dans Ajouter Amazon GameLift à votre serveur de jeu. Une fois cette tâche terminée, vous disposez de la plupart des mécanismes dont vous avez besoin. Les sections de cette rubrique couvrent les tâches restantes nécessaires au traitement des jeux configurés avec FlexMatch.

Configurez votre serveur de jeu pour le matchmaking

Pour configurer votre serveur de jeu au traitement des jeux correspondants, réalisez les étapes suivantes.

1. Lancer des sessions de jeu créées avec la mise en relation. Pour demander une nouvelle session de jeu, Amazon GameLift envoie une `onStartGameSession()` demande à votre serveur de jeu avec un objet de session de jeu (voir GameSession). Pour démarrer la session de jeu demandée, votre serveur de jeu utilise les informations de session de jeu, notamment les données de jeu personnalisées. Pour plus de détails, voir Démarrer une session de jeu.

   Pour les jeux mis en relation, l'objet de session de jeu contient également un ensemble de données du matchmaker. Ces dernières comprennent les informations nécessaires à votre serveur de jeux afin de configurer une nouvelle session de jeu pour la partie. Cela inclut la structure de l'équipe, les affectations d'équipe et certains attributs de joueur qui peuvent être utiles pour votre jeu. Par exemple, votre jeu peut débloquer certaines fonctionnalités ou certains niveaux en fonction du niveau de compétence des joueurs, ou peut choisir une carte en fonction des préférences des joueurs. Pour en savoir plus, voir Travaillez avec les données du matchmaker.
2. Gérer les connexions des joueurs. Lors de la connexion à un jeu correspondant, un client de jeu fait référence à un identifiant de joueur et à un identifiant de session de joueur (voir Valider un nouveau joueur). Le serveur de jeux utilise l'ID de joueur pour associer le joueur entrant aux informations contenues dans les données du matchmaker. Les données du matchmaker identifient l'affectation d'équipe de chaque joueur et peuvent fournir d'autres informations nécessaires à la bonne représentation du joueur dans le jeu.
3. Signaler que des joueurs quittent le jeu. Assurez-vous que votre serveur de jeu appelle l'API du serveur `RemovePlayerSession()` pour signaler l'abandon de joueurs (voir Signaler la fin de la session d'un joueur). Cette étape est importante si vous utilisez le remplissage FlexMatch pour remplir les emplacements vides dans des jeux existants. Elle est essentielle si votre jeu initie les demandes de remplissage via un service de jeu côté client. Pour plus d'informations

sur l'implémentation du remplissage FlexMatch, consultez Remplir les jeux existants avec FlexMatch.

4. Demander de nouveaux joueurs pour des sessions de jeu correspondant existantes (facultatif). Déterminez la façon dont vous souhaitez remplir les jeux existants. Si le remplissage manuel est configuré pour vos matchmakers, il peut être utile d'ajouter à votre jeu la prise en charge du remplissage. Si le mode de remplissage est automatique, il se peut que vous deviez inclure un moyen de le désactiver pour les sessions de jeu individuelles. Par exemple, vous pouvez choisir d'arrêter le remplissage d'une session une fois qu'un certain stade est atteint dans le jeu. Pour en savoir plus sur la gestion du remplissage des parties, voire Remplir les jeux existants avec FlexMatch.

Travaillez avec les données du matchmaker

Votre serveur de jeu doit être capable de reconnaître et d'utiliser les informations de jeu contenues dans un GameSessionobjet. Le GameLift service Amazon transmet ces objets à votre serveur de jeu chaque fois qu'une session de jeu est lancée ou mise à jour. Les informations de session de jeu principales comprennent l'ID et le nom de la session de jeu, le nombre maximum de joueurs, les informations de connexion et les données de jeu personnalisées (si fournies).

Pour les sessions de jeu qui sont créées à l'aide de FlexMatch, l'objet `GameSession` contient également un ensemble de données du matchmaker. En plus d'un ID de partie unique, il identifie le matchmaker utilisé pour créer la rencontre et décrit les équipes, les affectations de l'équipe et les joueurs. Il comprend les attributs de joueur depuis la demande de mise en relation d'origine (consultez l'objet Player (joueur)). Il ne comprend pas la latence du joueur. Si vous avez besoin des données de latence des joueurs actuels, par exemple dans le cadre d'un remplissage de rencontre, nous vous recommandons d'actualiser les données.

> **Note**
>
> Les données Matchmaker spécifient l'ARN complet de la configuration de matchmaking, qui identifie le nom de la configuration, le AWS compte et la région. Lorsque vous demandez un remplissage de partie depuis un client ou un service de jeux, seul le nom de la configuration est nécessaire. Vous pouvez extraire le nom de la configuration en analysant la chaîne qui suit « :matchmakingconfiguration/ ». Dans l'exemple illustré, le nom de la configuration du matchmaking est « MyMatchmakerConfig ».

Le code JSON suivant illustre un ensemble de données de matchmaker typique. Cet exemple décrit un jeu à deux joueurs, les joueurs étant mis en relation en fonction de leurs scores de compétence et du niveau atteint le plus élevé. Le matchmaker base également la mise en relation sur le personnage et garantie que les joueurs mis en relation ont au moins une préférence de carte en commun. Dans ce scénario, le serveur de jeux doit être en mesure de déterminer quelle carte est la préférée et de l'utiliser dans la session de jeu.

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

Remplir les jeux existants avec FlexMatch

Le remplissage utilise vos mécanismes FlexMatch pour trouver de nouveaux joueurs pour de sessions de jeu mises en correspondance existantes. Bien que vous puissiez toujours ajouter des joueurs à n'importe quel jeu (voir Rejoindre un joueur à une session de jeu), le remplissage des matchs garantit que les nouveaux joueurs répondent aux mêmes critères de match que les joueurs actuels. En outre, le remplissage des parties attribue les nouveaux joueurs à des équipes, gère

l'acceptation des joueurs et envoie des informations mises à jour sur les parties au serveur de jeux. Découvrez le renvoi de correspondance dans FlexMatchprocessus de jumelage.

> **Note**
>
> FlexMatchLe backfill n'est actuellement pas disponible pour les jeux utilisant des serveurs en temps réel.

Il existe deux types de mécanismes de remplissage :

- Pour remplir les sessions de jeu qui commencent avec un nombre de joueurs inférieur au nombre maximum autorisé, activez le remplissage automatique.
- Pour remplacer les joueurs qui abandonnent une session de jeu en cours, ajoutez des fonctionnalités à votre serveur de jeu afin d'envoyer des demandes de remplissage.

Activer le remblayage automatique

Grâce au remplissage automatique des matchs, Amazon déclenche GameLift automatiquement une demande de remplissage chaque fois qu'une session de jeu commence avec un ou plusieurs emplacements de joueurs vides. Cette fonction permet de commencer les parties dès que le nombre minimum de joueurs mis en correspondance est trouvé et de remplir les emplacements restants plus tard, lorsque des joueurs supplémentaires sont mis en correspondance. Vous pouvez choisir d'arrêter le remplissage automatique à tout moment.

Prenons l'exemple d'un jeu pouvant accepter six à dix joueurs. FlexMatchlocalise initialement six joueurs, forme le match et lance une nouvelle session de jeu. Avec le remplissage automatique, la nouvelle session de jeu peut immédiatement demander quatre joueurs supplémentaires. Selon le style de jeu, nous souhaiterons peut-être autoriser de nouveaux joueurs à rejoindre le jeu à tout moment pendant la session de jeu. Nous pouvons également vouloir arrêter le remplissage automatique après la phase de configuration initiale et avant le début du jeu.

Pour ajouter le remplissage automatique à un jeu, effectuez les mises à jour suivantes.

1. Activez le remplissage automatique. Le remplissage automatique est géré dans la configuration de la mise en relation. Lorsqu'il est activé, il est utilisé avec toutes les sessions de jeu correspondantes créées avec ce système de matchmaking. Amazon GameLift commence à

générer des demandes de remplissage pour une session de jeu qui n'est pas complète dès que la session de jeu démarre sur un serveur de jeu.

Pour activer le remplissage automatique, ouvrez la configuration d'une mise en relation et définissez le mode de remplissage sur « AUTOMATIQUE ». Pour plus d'informations, consultez Créer une configuration de matchmaking.

2. Activez la priorisation du remblayage. Personnalisez votre processus de matchmaking pour prioriser le remplissage des demandes de remplissage avant de créer de nouvelles correspondances. Dans votre ensemble de règles de matchmaking, ajoutez un composant d'algorithme et définissez la priorité du remblayage sur « élevée ». Pour en savoir plus, consultez Personnalisez l'algorithme de correspondance.
3. Mettez à jour la session de jeu avec de nouvelles données de matchmaking. Amazon GameLift met à jour votre serveur de jeu avec les informations correspondantes à l'aide de la fonction de rappel du SDK du serveur `onUpdateGameSession` (voir Initialiser le processus du serveur). Ajoutez du code au serveur de jeux pour traiter les mises à jour des objets de session de jeu suite à l'action de remplissage. Pour en savoir plus, voir Mettre à jour les données des matchs sur le serveur de jeu.
4. Désactivez le remplissage automatique pour une session de jeu. Vous pouvez choisir d'arrêter le remplissage automatique à tout moment au cours d'une session de jeu individuelle. Pour arrêter le remplissage automatique, ajoutez du code à votre client de jeu ou à votre serveur de jeu afin d'effectuer un appel StopMatchmakingà l'GameLiftAPI Amazon. Cet appel nécessite un ID de ticket. Utilisez l'ID de ticket de la dernière demande de remplissage. Vous pouvez obtenir ces informations à partir des données de mise en relation de la session de jeu, lesquelles sont mises à jour comme décrit à l'étape précédente.

Envoyer des demandes de remblayage (depuis un serveur de jeu)

Vous pouvez initier des requêtes de renvoi de correspondance directement depuis le processus de serveur de jeu qui héberge la session de jeu. Le processus du serveur contient le plus up-to-date d'informations sur les joueurs actuellement connectés au jeu et sur l'état des emplacements vides.

Il suppose que vous avez déjà créé les composants FlexMatch nécessaires et ajouté avec succès les processus de mise en relation au serveur de jeux et à un service de jeu côté client. Pour plus d'informations sur la configuration de FlexMatch, consultez FlexMatchintégration avec l'GameLifthébergement Amazon.

Pour activer le remplissage des parties pour votre jeu, vous devez ajouter les fonctionnalités suivantes :

- Envoyer les demandes de remplissage à un matchmaker et suivre l'état de ces demandes.
- Mettre à jour les informations de correspondance pour la session de jeu. Consultez Mettre à jour les données des matchs sur le serveur de jeu.

Comme pour les autres fonctionnalités du serveur, un serveur de jeu utilise le SDK Amazon GameLift Server. Ce kit SDK est disponible en C++ et C#.

Pour créer des requêtes de renvoi de correspondance depuis votre serveur de jeux, exécutez les tâches suivantes.

1. Déclenchez une requête de renvoi de correspondance. En général, vous pouvez si vous le souhaitez initier une requête de renvoi chaque fois qu'un jeu correspondant comporte un ou plusieurs emplacements de joueur vides. Vous pouvez si vous le souhaitez lier des requêtes de renvoi à des circonstances spécifiques, comme des rôles de personnages critiques ou l'équilibrage des équipes. Vous pouvez aussi limiter l'activité de renvoi en fonction de l'ancienneté d'une session de jeu.
2. Créez une requête de renvoi. Ajoutez le code pour créer et envoyer les requêtes de renvoi de correspondance à un matchmaker FlexMatch. Les requêtes de renvoi sont gérées à l'aide des API suivantes du serveur :

   - StartMatchBackfill()
   - StopMatchBackfill()

   Pour créer une requête de renvoi, appelez `StartMatchBackfill` avec les informations suivantes. Pour annuler une requête de renvoi, appelez `StopMatchBackfill` avec l'identifiant de ticket de requête de renvoi.

   - ID de ticket — Fournissez un identifiant de ticket de matchmaking (ou optez pour qu'ils soient générés automatiquement). Vous pouvez utiliser le même mécanisme pour attribuer des identifiants de ticket à la fois aux requêtes de correspondance et de renvoi. Les tickets de correspondance et de renvoi sont traités de la même manière.
   - Matchmaker — Identifiez le système de matchmaking à utiliser pour la demande de remblayage. En général, vous devrez utiliser le matchmaker qui a été utilisé pour créer la correspondance d'origine. Cette requête nécessite un ARN de configuration

de correspondance. Ces informations sont stockées dans l'objet de session de jeu (GameSession), qui a été fourni au processus serveur par Amazon GameLift lors de l'activation de la session de jeu. L'ARN de configuration de correspondance est inclus dans la propriété `MatchmakerData`.

- ARN de session de jeu : identifie la session de jeu en cours de remblayage. Vous pouvez obtenir l'ARN de la session de jeu en appelant l'API du serveur GetGameSessionId(). Pendant le processus de correspondance, les tickets des nouvelles requêtes n'ont pas d'identifiant de session de jeu, tandis que les tickets des requêtes de renvoi en ont un. La présence d'un identifiant de session de jeu est un moyen de faire la différence entre les tickets des nouvelles correspondances et les tickets des renvois.
- Données du joueur : incluez les informations sur les joueurs (joueur) pour tous les joueurs actuels de la session de jeu que vous complétez. Ces information permettent au matchmaker de localiser les meilleures correspondances de joueur possibles pour les joueurs actuellement dans la session de jeu. Vous devez inclure l'appartenance à l'équipe pour chaque joueur. Ne spécifiez pas d'équipe si vous n'utilisez pas le remblayage. Si votre serveur de jeux indique de manière précise l'état de connexion des joueurs, vous devez pouvoir acquérir ces données comme suit :

  1. Le processus serveur hébergeant la session de jeu doit contenir le plus up-to-date d'informations sur les joueurs actuellement connectés à la session de jeu.
  2. Pour obtenir les identifiants des joueurs, les attributs et les attributions des équipes, extrayez les données des joueurs à partir de l'objet (GameSession) de la session de jeu, de la `MatchmakerData` propriété (voirTravaillez avec les données du matchmaker). Les données du matchmaker incluent tous les joueurs qui ont été mis en correspondance avec la session de jeu, vous devez donc extraire les données de joueur des joueurs actuellement connectés uniquement.
  3. Pour la latence de joueur, si le matchmaker appelle les données de latence, collectez les nouvelles valeurs de latence de tous les joueurs actuels et incluez-les dans chaque objet `Player`. Si les données de latence sont omises et si le matchmaker utilise une règle de latence, la requête ne sera pas correctement mise en correspondance. Les requêtes de renvoi nécessitent les données de latence uniquement pour la région dans laquelle se trouve actuellement la session de jeu. Vous pouvez obtenir la région d'une session de jeu dans la propriété `GameSessionId` de l'objet `GameSession` ; cette valeur est un ARN, lequel inclut la région.

3. Suivez l'état d'une demande de remblayage. Amazon GameLift informe votre serveur de jeu de l'état des demandes de remplissage à l'aide de la fonction de rappel du SDK du serveur

`onUpdateGameSession` (voir Initialiser le processus du serveur). Ajoutez du code pour gérer les messages d'état, ainsi que les objets de session de jeu mis à jour suite à des demandes de remplissage réussies, à l'adresse. Mettre à jour les données des matchs sur le serveur de jeu

Un matchmaker peut uniquement traiter une seul requête de renvoi de correspondance depuis une session de jeu à la fois. Si vous devez annuler une demande, appelez StopMatchBackfill(). Si vous devez modifier une requête, appelez `StopMatchBackfill` et soumettez une requête mise à jour.

Envoyer des demandes de remblayage (depuis un service client)

Comme alternative à l'envoi de requêtes de renvoi à partir d'un serveur de jeux, vous pouvez les envoyer à partir d'un service de jeu côté client. Pour utiliser cette option, le service côté client doit avoir accès aux données actuelles sur l'activité de session de jeu et des connexions joueur. Si votre jeu utilise un service d'annuaire de session, cela pourrait être un bon choix.

Il suppose que vous avez déjà créé les composants FlexMatch nécessaires et ajouté avec succès les processus de mise en relation au serveur de jeux et à un service de jeu côté client. Pour plus d'informations sur la configuration de FlexMatch, consultez FlexMatchintégration avec l'GameLifthébergement Amazon.

Pour activer le remplissage des parties pour votre jeu, vous devez ajouter les fonctionnalités suivantes :

- Envoyer les demandes de remplissage à un matchmaker et suivre l'état de ces demandes.
- Mettre à jour les informations de correspondance pour la session de jeu. Consultez Mettre à jour les données des matchs sur le serveur de jeu.

Comme pour les autres fonctionnalités du client, un service de jeu côté client utilise le AWS SDK avec l'API Amazon. GameLift Ce kit SDK est disponible en C++, C # et plusieurs autres langages. Pour une description générale des API clientes, consultez la référence des API de GameLift service Amazon, qui décrit l'API de service de bas niveau pour les actions GameLift liées à Amazon et inclut des liens vers des guides de référence spécifiques à chaque langue.

Pour configurer un service de jeu côté client pour le renvoi de jeux correspondants, exécutez les tâches suivantes.

1. Déclenchez une requête pour le renvoi. En général, un jeu initie une requête de renvoi chaque fois qu'un jeu correspondant comporte un ou plusieurs emplacements de joueur vides. Vous pouvez si vous le souhaitez lier des requêtes de renvoi à des circonstances spécifiques, comme des rôles de personnages critiques ou l'équilibrage des équipes. Vous pouvez aussi limiter le renvoi en fonction de l'ancienneté d'une session de jeu. Quel que soit l'élément que vous utilisez pour un déclencheur, vous devrez au minimum connaître les informations suivantes. Vous pouvez obtenir ces informations à partir de l'objet de session de jeu (GameSession) en appelant DescribeGameSessionsavec un identifiant de session de jeu.

   - Nombre d'emplacements de joueur actuellement vides. Cette valeur peut être calculée à partir de la limite de joueur maximale d'une session de jeu et du nombre de joueurs en cours. Le nombre de joueurs actuel est mis à jour chaque fois que votre serveur de jeu contacte le GameLift service Amazon pour valider la connexion d'un nouveau joueur ou pour signaler un joueur abandonné.

   - Stratégie de création. Ce paramètre indique si la session de jeu accepte actuellement de nouveaux joueurs.

   L'objet de session de jeu peut potentiellement contenir d'autres informations utiles, comme l'heure de début de la session de jeu, les propriétés de jeu personnalisées et les données de matchmaker.

2. Créez une requête de renvoi. Ajoutez le code pour créer et envoyer les requêtes de renvoi de correspondance à un matchmaker FlexMatch. Les requêtes de renvoi sont gérées à l'aide des API suivantes du client :

   - StartMatchBackfill
   - StopMatchmaking

   Pour créer une requête de renvoi, appelez `StartMatchBackfill` avec les informations suivantes. Une requête de renvoi est similaire à une requête de correspondance (consultez Demande de matchmaking pour les joueurs), mais elle identifie également la session de jeu existante. Pour annuler une requête de renvoi, appelez `StopMatchmaking` avec l'identifiant de ticket de requête de renvoi.

   - ID de ticket — Fournissez un identifiant de ticket de matchmaking (ou optez pour qu'ils soient générés automatiquement). Vous pouvez utiliser le même mécanisme pour attribuer des

identifiants de ticket à la fois aux requêtes de correspondance et de renvoi. Les tickets de correspondance et de renvoi sont traités de la même manière.

- Matchmaker — Identifiez le nom de la configuration de matchmaking à utiliser. En général, vous devrez utiliser le matchmaker pour le renvoi qui a été utilisé pour créer la correspondance d'origine. Ces informations se trouvent dans un objet de session de jeu (GameSession), une `MatchmakerData` propriété, sous l'ARN de configuration du matchmaking. La valeur de nom est la chaîne qui suit « matchmakingconfiguration / ». (Par exemple, dans la valeur d'ARN « arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4 », le nom de la configuration de correspondance est « MM-4v4 ».)
- ARN de session de jeu — Spécifiez la session de jeu à remplir. Utilisez la propriété `GameSessionId` de l'objet de session de jeu ; cet identifiant utilise la valeur d'ARN dont vous avez besoin. Les tickets de matchmaking (MatchmakingTicket) pour les demandes de remblayage portent l'identifiant de session de jeu pendant leur traitement ; les tickets pour les nouvelles demandes de matchmaking ne reçoivent pas d'identifiant de session de jeu tant que le match n'est pas joué ; la présence d'un identifiant de session de jeu est un moyen de faire la différence entre les tickets pour les nouveaux matchs et les tickets pour les remplissages.
- Données du joueur : incluez les informations sur les joueurs (joueur) pour tous les joueurs actuels de la session de jeu que vous complétez. Ces information permettent au matchmaker de localiser les meilleures correspondances de joueur possibles pour les joueurs actuellement dans la session de jeu. Vous devez inclure l'appartenance à l'équipe pour chaque joueur. Ne spécifiez pas d'équipe si vous n'utilisez pas le remblayage. Si votre serveur de jeux indique de manière précise l'état de connexion des joueurs, vous devez pouvoir acquérir ces données comme suit :

  1. Appelez DescribePlayerSessions() avec l'identifiant de la session de jeu pour découvrir tous les joueurs actuellement connectés à la session de jeu. Chaque session de joueur inclut un identifiant de joueur. Vous pouvez ajouter un filtre de statut pour récupérer les sessions de joueur actives uniquement.
  2. Extraire les données du joueur à partir de l'objet (GameSession) de la session de jeu, de la `MatchmakerData` propriété (voirTravaillez avec les données du matchmaker. Utilisez les identifiants de joueur acquis à l'étape précédente pour l'obtention des données des joueurs actuellement connectés uniquement. Dans la mesure où les données de matchmaker ne sont pas mises à jour lorsque les joueurs abandonnent, vous devez extraire les données des joueurs actuels uniquement.
  3. Pour la latence de joueur, si le matchmaker appelle les données de latence, collectez les nouvelles valeurs de latence de tous les joueurs actuels et incluez-les dans l'objet `Player`.

Si les données de latence sont omises et si le matchmaker utilise une règle de latence, la requête ne sera pas correctement mise en correspondance. Les requêtes de renvoi nécessitent les données de latence uniquement pour la région dans laquelle se trouve actuellement la session de jeu. Vous pouvez obtenir la région d'une session de jeu dans la propriété `GameSessionId` de l'objet `GameSession` ; cette valeur est un ARN, lequel inclut la région.

3. Suivez l'état de la requête de renvoi. Ajoutez du code pour écouter les mises à jour de statut du ticket de correspondance. Vous pouvez utiliser le mécanisme défini pour suivre les tickets des nouvelles requêtes de correspondance (consultez Suivez les événements de matchmaking) à l'aide de la notification d'événement (préférée) ou l'interrogation. Même s'il n'est pas nécessaire de déclencher l'activité d'acceptation de joueur à l'aide de requêtes de renvoi, et si les informations de joueur sont mises à jour sur le serveur de jeux, vous devez toujours surveiller le statut du ticket pour gérer les erreurs de requêtes et les nouvelles soumissions de requêtes.

   Un matchmaker peut uniquement traiter une seul requête de renvoi de correspondance depuis une session de jeu à la fois. Si vous devez annuler une requête, appelez StopMatchmaking. Si vous devez modifier une requête, appelez `StopMatchmaking` et soumettez une requête mise à jour.

   Dès qu'une requête de renvoi aboutit, votre serveur de jeux reçoit un objet `GameSession` mis à jour et gère les tâches nécessaires pour associer de nouveaux joueurs à la session de jeu. Pour plus d'informations, consultez Mettre à jour les données des matchs sur le serveur de jeu.

Mettre à jour les données des matchs sur le serveur de jeu

Quelle que soit la manière dont vous lancez les demandes de remplissage de matchs dans votre jeu, votre serveur de jeu doit être en mesure de gérer les mises à jour de session de jeu qu'Amazon GameLift fournit à la suite des demandes de remplissage de matchs.

Lorsqu'Amazon traite une GameLift demande de match backfill, avec succès ou non, il appelle votre serveur de jeu à l'aide de la fonction de rappel. `onUpdateGameSession` Cet appel comporte trois paramètres d'entrée : un identifiant de ticket de match backfill, un message de statut et un GameSession objet contenant le plus de données de up-to-date matchmaking, y compris les informations sur les joueurs. Vous devez ajouter le code suivant à votre serveur de jeux dans le cadre de votre intégration au serveur de jeux :

1. Implémentez la fonction onUpdateGameSession. Cette fonction doit pouvoir gérer les messages de statut suivants (updateReason) :
   - MATCHMAKING_DATA_UPDATED — Les nouveaux joueurs ont été associés avec succès à la session de jeu. L'objet GameSession contient les données de matchmaker mises à jour; y compris les données de joueur sur les joueurs existants et les joueurs nouvellement mis en correspondance.
   - BACKFILL_FAILED — La tentative de remplissage du match a échoué en raison d'une erreur interne. L'objet GameSession est inchangé.
   - BACKFILL_TIMED_OUT — Le système de matchmaking n'a pas trouvé de correspondance de remplissage dans le délai imparti. L'objet GameSession est inchangé.
   - BACKFILL_CANCELLED — La demande de remplissage correspondant a été annulée par un appel à StopMatchmaking (client) ou StopMatchBackfill (serveur). L'objet GameSession est inchangé.
2. Pour que les correspondances de renvoi aboutissent, utilisez les données matchmaker mises à jour pour gérer les nouveaux joueurs lorsqu'ils se connectent à la session de jeu. Au minimum, vous devrez utiliser les affectations d'équipe pour le ou les nouveaux joueurs, ainsi que d'autres attributs de joueur qui sont requis pour que le joueur puisse démarrer dans le jeu.
3. Dans l'appel de votre serveur de jeu à l'action ProcessReady() du SDK du serveur, ajoutez le nom de la méthode de onUpdateGameSession rappel en tant que paramètre de processus.

GameLiftFlexMatchRéférence Amazon

Cette section contient de la documentation de référence pour le matchmaking avec Amazon GameLiftFlexMatch.

Rubriques

- Référence d'GameLiftFlexMatchAPI Amazon (AWSSDK)
- FlexMatchlangage des règles
- FlexMatchévénements de matchmaking

Référence d'GameLiftFlexMatchAPI Amazon (AWSSDK)

Cette rubrique fournit une liste basée sur les tâches des opérations d'API pour Amazon. GameLift FlexMatch L'API du GameLift FlexMatch service Amazon est intégrée au AWS SDK de l'espace de `aws.gamelift` nommage. Téléchargez le AWS SDK ou consultez la documentation de référence de GameLift l'API Amazon.

Amazon GameLift FlexMatch fournit des services de jumelage à utiliser avec des jeux hébergés par des solutions GameLift d'hébergement Amazon (y compris l'hébergement géré pour des serveurs de jeu personnalisés ou des serveurs en temps réel, et l'hébergement sur Amazon EC2 avec Amazon GameLift FleetIQ), ainsi qu'avec d'autres systèmes d'hébergement tels que des systèmes sur site ou des primitives de peer-to-peer calcul dans le cloud. Consultez le guide du GameLift développeur Amazon pour plus d'informations sur les autres options GameLift d'hébergement Amazon.

Mettre en place des règles et des processus de matchmaking

Appelez ces opérations pour créer un FlexMatch système de matchmaking, configurer le processus de matchmaking pour votre jeu et définir un ensemble de règles personnalisées pour créer des matchs et des équipes.

Configuration de la mise en relation

- CreateMatchmakingConfiguration— Créez une configuration de matchmaking avec des instructions pour évaluer les groupes de joueurs et constituer des équipes de joueurs. Lorsque vous utilisez Amazon GameLift pour l'hébergement, précisez également comment créer une nouvelle session de jeu pour le match.

- DescribeMatchmakingConfigurations— Récupère les configurations de matchmaking définies pour une GameLift région Amazon.
- UpdateMatchmakingConfiguration— Modifiez les paramètres de configuration du matchmaking. file d'attente.
- DeleteMatchmakingConfiguration— Supprime une configuration de matchmaking de la région.

Ensemble de règles de mise en relation

- CreateMatchmakingRuleSet— Créez un ensemble de règles à utiliser lors de la recherche de matchs entre joueurs.
- DescribeMatchmakingRuleSets— Récupérez les ensembles de règles de matchmaking définis dans une GameLift région Amazon.
- ValidateMatchmakingRuleSet— Vérifiez la syntaxe d'un ensemble de règles de matchmaking.
- DeleteMatchmakingRuleSet— Supprime un ensemble de règles de matchmaking de la région.

Demandez un match pour un ou plusieurs joueurs

Appelez ces opérations depuis votre service client de jeu pour gérer les demandes de mise en relation des joueurs.

- StartMatchmaking— Demandez le matchmaking pour un joueur ou un groupe qui souhaite participer au même match.
- DescribeMatchmaking— Obtenez des détails sur une demande de matchmaking, y compris son statut.
- AcceptMatch— Pour un match nécessitant l'acceptation d'un joueur, informez Amazon GameLift lorsqu'un joueur accepte un match proposé.
- StopMatchmaking— Annulez une demande de matchmaking.
- StartMatchBackfill- Demandez des matchs entre joueurs supplémentaires pour remplir les cases vides d'une session de jeu existante.

Langages de programmation disponibles

Le AWS SDK compatible avec Amazon GameLift est disponible dans les langues suivantes. Pour plus d'informations sur la prise en charge des environnements de développement, consultez la documentation de chaque langue.

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

FlexMatchlangage des règles

Les rubriques de référence de cette section décrivent la syntaxe et la sémantique utilisées pour créer des règles de matchmaking à utiliser avec Amazon. GameLift FlexMatch Pour une aide détaillée sur la rédaction de règles et d'ensembles de règles de matchmaking, voirCréation d'un ensemble de FlexMatch règles.

Rubriques

- FlexMatchschéma d'ensemble de règles
- FlexMatchdéfinitions des propriétés d'un ensemble de règles
- FlexMatchtypes de règles
- FlexMatchexpressions de propriété

FlexMatchschéma d'ensemble de règles

Les ensembles de règles FlexMatch utilisent le schéma d'ensemble de règles standard pour les parties de petite envergure et à grande échelle. Pour une description détaillée de chaque section, reportez-vous à la sectionFlexMatchdéfinitions des propriétés d'un ensemble de règles.

Schéma d'ensemble de règles pour les petits matchs

Le schéma suivant décrit toutes les propriétés possibles et les valeurs autorisées pour un ensemble de règles utilisé pour créer des matchs réunissant jusqu'à 40 joueurs.

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

```
}
```

Schéma d'ensemble de règles pour les matchs importants

Le schéma suivant décrit toutes les propriétés possibles et les valeurs autorisées pour un ensemble de règles utilisé pour créer des matchs réunissant plus de 40 joueurs. Si le total des `maxPlayers` valeurs pour toutes les équipes de l'ensemble de règles est supérieur à 40, FlexMatch les demandes de matchs utilisant cet ensemble de règles sont traitées conformément aux directives relatives aux matchs importants.

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

FlexMatchdéfinitions des propriétés d'un ensemble de règles

Cette section définit chaque propriété du schéma de l'ensemble de règles. Pour obtenir de l'aide supplémentaire sur la création d'un ensemble de règles, consultezCréation d'un ensemble de FlexMatch règles.

`name`

Une étiquette descriptive pour l'ensemble de règles. Cette valeur n'est pas associée au nom attribué à la GameLift MatchmakingRuleSetressource Amazon. Cette valeur est incluse dans les données de matchmaking décrivant une correspondance terminée, mais elle n'est utilisée par aucun GameLift processus Amazon.

Valeurs autorisées : chaîne

Obligatoire ? Non

`ruleLanguageVersion`

Version du langage d'expression de FlexMatch propriété utilisé.

Valeurs autorisées : « 1.0 »

Obligatoire ? Oui

`playerAttributes`

Une collection de données sur les joueurs qui est incluse dans les demandes de matchmaking et utilisée dans le processus de matchmaking. Vous pouvez également déclarer des attributs ici pour que les données du joueur soient incluses dans les données de matchmaking transmises aux serveurs de jeu, même si les données ne sont pas utilisées dans le processus de matchmaking.

Obligatoire ? Non

`name`

Un nom unique pour l'attribut du joueur à utiliser par le système de matchmaking. Ce nom doit correspondre au nom de l'attribut du joueur référencé dans les demandes de matchmaking.

Valeurs autorisées : chaîne

Obligatoire ? Oui

`type`

Le type de données de la valeur de l'attribut du joueur.

Valeurs autorisées : « string », « number », « string_list », « string_number_map »

Obligatoire ? Oui

`default`

Une valeur par défaut à utiliser lorsqu'une demande de matchmaking n'en fournit pas pour un joueur.

Valeurs autorisées : n'importe quelle valeur autorisée pour l'attribut du joueur.

Obligatoire ? Non

`algorithm`

Paramètres de configuration facultatifs pour personnaliser le processus de matchmaking.

Obligatoire ? Non

`strategy`

La méthode à utiliser pour créer des allumettes. Si cette propriété n'est pas définie, le comportement par défaut est « ExhaustiveSearch ».

Valeurs autorisées :

- « Recherche exhaustive » — Méthode de correspondance standard. FlexMatchforme un match autour du ticket le plus ancien d'un lot en évaluant les autres tickets du pool sur la base d'un ensemble de règles de match personnalisées. Cette stratégie est utilisée

pour les matchs de 40 joueurs ou moins. Lorsque vous utilisez cette stratégie, vous `batchingPreference` devez la régler sur « aléatoire » ou « trié ».

- « équilibrée » : méthode optimisée pour former rapidement de grandes allumettes. Cette stratégie n'est utilisée que pour les matchs de 41 à 200 joueurs. Il organise les matchs en triant à l'avance le pool de tickets, en créant des matchs potentiels et en affectant les joueurs à des équipes, puis en équilibrant chaque équipe lors d'un match à l'aide d'un attribut de joueur spécifié. Par exemple, cette stratégie peut être utilisée pour égaliser les niveaux de compétence moyens de toutes les équipes lors d'un match. Lorsque vous utilisez cette stratégie, elle `balancedAttribute` doit être définie et `batchingPreference` doit être définie sur « LargestPopulation » ou « FastestRegion ». La plupart des types de règles personnalisés ne sont pas reconnus avec cette stratégie.

Obligatoire ? Oui

`batchingPreference`

La méthode de tri préalable à utiliser avant de regrouper les tickets pour la construction de matchs. Le tri préalable du pool de billets entraîne le regroupement des billets en fonction d'une caractéristique spécifique, ce qui tend à améliorer l'uniformité entre les joueurs lors des matchs finaux.

Valeurs autorisées :

- « random » — Valable uniquement avec `strategy` = « ExhaustiveSearch ». Aucun tri préalable n'est effectué ; les billets du pool sont regroupés de manière aléatoire. Il s'agit du comportement par défaut pour une stratégie de recherche exhaustive.
- « sorted » — Valable uniquement avec `strategy` = « ExhaustiveSearch ». Le pool de tickets est pré-trié en fonction des attributs des joueurs répertoriés dans`sortbyAttributes`.
- « LargestPopulation » — Valable uniquement avec `strategy` = « balanced ». Le pool de tickets est pré-trié par région dans laquelle les joueurs signalent des niveaux de latence acceptables. Il s'agit du comportement par défaut pour une stratégie équilibrée.
- « FastestRegion » — Valable uniquement avec `strategy` = « balanced ». Le pool de tickets est pré-trié par région dans laquelle les joueurs signalent leur plus faible niveau de latence. Les parties qui en résultent prennent plus de temps à se terminer, mais la latence pour tous les joueurs a tendance à être faible.

Obligatoire ? Oui

`balancedAttribute`

Le nom d'un attribut de joueur à utiliser lors de la construction de matchs importants avec une stratégie équilibrée.

Valeurs autorisées : Tout attribut déclaré `playerAttributes` avec `type` = « nombre ».

Obligatoire ? Oui, si `strategy` = « équilibré ».

`sortByAttributes`

Liste des attributs des joueurs à utiliser lors du tri préalable du pool de tickets avant le traitement par lots. Cette propriété n'est utilisée que lors du tri préalable avec la stratégie de recherche exhaustive. L'ordre de la liste d'attributs détermine l'ordre de tri. FlexMatchutilise une convention de tri standard pour les valeurs alphanumériques et numériques.

Valeurs autorisées : n'importe quel attribut déclaré dans`playerAttributes`.

Obligatoire ? Oui, si `batchingPreference` = « trié ».

`backfillPriority`

La méthode de priorisation pour faire correspondre les tickets de remblayage. Cette propriété détermine à quel moment FlexMatch les tickets de remblayage sont traités par lot. Il n'est utilisé que lors du tri préalable avec la stratégie de recherche exhaustive. Si cette propriété n'est pas définie, le comportement par défaut est « normal ».

Valeurs autorisées :

- « normal » — Le type de demande d'un ticket (remplissage ou nouveau match) n'est pas pris en compte lors de la création de matchs.
- « élevé » : un lot de tickets est trié par type de demande (puis par âge) et FlexMatch tente d'abord de faire correspondre les tickets de remplissage.
- « faible » : un lot de tickets est trié par type de demande (puis par âge) et FlexMatch tente d'abord de faire correspondre les tickets non renvoyés.

Obligatoire ? Non

`expansionAgeSelection`

Méthode de calcul du temps d'attente pour l'extension d'une règle de correspondance. Les extensions sont utilisées pour assouplir les exigences des matchs si un match n'est pas

terminé après un certain laps de temps. Le temps d'attente est calculé en fonction de l'âge des billets qui sont déjà partiellement remplis pour le match. Si cette propriété n'est pas définie, le comportement par défaut est « le plus récent ».

Valeurs autorisées :

- « le plus récent » : le temps d'attente pour l'extension est calculé sur la base du ticket dont l'horodatage de création est le plus récent lors d'un match partiellement terminé. Les extensions ont tendance à être déclenchées plus lentement, car un nouveau ticket peut relancer le temps d'attente.
- « le plus ancien » : le temps d'attente pour l'extension est calculé sur la base du ticket dont l'horodatage de création est le plus ancien du match. Les extensions ont tendance à être déclenchées plus rapidement.

Obligatoire ? Non

`teams`

La configuration des équipes lors d'un match. Indiquez un nom d'équipe et une fourchette de taille pour chaque équipe. Un ensemble de règles doit définir au moins une équipe.

`name`

Un nom unique pour l'équipe. Les noms des équipes peuvent être mentionnés dans les règles et les extensions. En cas de match réussi, les joueurs sont assignés par nom d'équipe dans les données de matchmaking.

Valeurs autorisées : chaîne

Obligatoire ? Oui

`maxPlayers`

Le nombre maximum de joueurs pouvant être affectés à l'équipe.

Valeurs autorisées : Nombre

Obligatoire ? Oui

`minPlayers`

Le nombre minimum de joueurs qui doivent être affectés à l'équipe avant le match est viable.

Valeurs autorisées : Nombre

Obligatoire ? Oui

`quantity`

Le nombre d'équipes de ce type à créer lors d'un match. Les équipes dont les quantités sont supérieures à 1 sont désignées par un chiffre ajouté (« Red_1 », « Red_2 », etc.). Si cette propriété n'est pas définie, la valeur par défaut est « 1 ».

Valeurs autorisées : Nombre

Obligatoire ? Non

`rules`

Un ensemble d'énoncés de règles qui définissent comment évaluer les joueurs lors d'un match.

Obligatoire ? Non

`name`

Un nom unique pour la règle. Toutes les règles d'un ensemble de règles doivent porter des noms uniques. Les noms des règles sont référencés dans les journaux d'événements et les mesures qui permettent de suivre l'activité liée à la règle.

Valeurs autorisées : chaîne

Obligatoire ? Oui

`description`

Description textuelle de la règle. Ces informations peuvent être utilisées pour identifier l'objectif d'une règle. Il n'est pas utilisé dans le processus de matchmaking.

Valeurs autorisées : chaîne

Obligatoire ? Non

`type`

Type d'énoncé de règle. Chaque type de règle possède des propriétés supplémentaires qui doivent être définies. Pour plus de détails sur la structure et l'utilisation de chaque type de règle, consultezFlexMatchtypes de règles.

Valeurs autorisées :

- « AbsoluteSort » : trie à l'aide d'une méthode de tri explicite qui classe les tickets d'un lot en fonction de la comparaison entre un attribut de joueur spécifié et le ticket le plus ancien du lot.
- « collection » : évalue les valeurs d'une collection, par exemple un attribut de joueur qui est une collection ou un ensemble de valeurs pour plusieurs joueurs.
- « comparaison » — Compare deux valeurs.
- « composé » — Définit une règle de matchmaking composée en utilisant une combinaison logique d'autres règles de l'ensemble de règles. Pris en charge uniquement pour les matchs de 40 joueurs ou moins.
- « distance » — Mesure la distance entre les valeurs numériques.
- « BatchDistance » : mesure la différence entre une valeur d'attribut et l'utilise pour regrouper les demandes de correspondance.
- « DistanceSort » : trie à l'aide d'une méthode de tri explicite qui classe les tickets d'un lot en fonction de la comparaison entre un attribut de joueur spécifié doté d'une valeur numérique et le ticket le plus ancien du lot.
- « latence » — Évalue les données de latence régionales signalées pour une demande de matchmaking.

Obligatoire ? Oui

`expansions`

Règles visant à assouplir les exigences relatives aux matchs au fil du temps lorsqu'un match ne peut pas être terminé. Configurez les extensions sous la forme d'une série d'étapes qui s'appliquent progressivement afin de faciliter la recherche de correspondances. Par défaut, FlexMatch calcule le temps d'attente en fonction de l'âge du dernier ticket ajouté à un match. Vous pouvez modifier la façon dont les temps d'attente d'expansion sont calculés à l'aide de la propriété de l'algorithme`expansionAgeSelection`.

Les temps d'attente pour l'extension sont des valeurs absolues, de sorte que chaque étape doit avoir un temps d'attente plus long que l'étape précédente. Par exemple, pour planifier une série d'extensions graduelles, vous pouvez utiliser des temps d'attente de 30 secondes, 40 secondes et 50 secondes. Les temps d'attente ne peuvent pas dépasser le temps maximum autorisé pour une demande de match, qui est défini dans la configuration du matchmaking.

Obligatoire ? Non

`target`

L'élément de l'ensemble de règles à assouplir. Vous pouvez assouplir les propriétés relatives à la taille de l'équipe ou à n'importe quelle propriété de déclaration de règle. La syntaxe est « <component name>[<rule/team name>]. <property name>». Par exemple, pour modifier la taille minimale des équipes :`teams[Red, Yellow].minPlayers`. Pour modifier la compétence minimale requise dans un énoncé de règle de comparaison nommé « MinSkill » :`rules[minSkill].referenceValue`.

Obligatoire ? Oui

`steps`

`waitTimeSeconds`

Durée, en secondes, à attendre avant d'appliquer la nouvelle valeur à l'élément de l'ensemble de règles cible.

Obligatoire ? Oui

`value`

La nouvelle valeur de l'élément d'ensemble de règles cible.

FlexMatchtypes de règles

Règle de distance par lots

```
batchDistance
```

Les règles de distance par lots mesurent la différence entre deux valeurs attributaires. Vous pouvez utiliser le type de règle de distance par lots pour les grandes et les petites correspondances. Il existe deux types de règles de distance par lots :

- Comparez les valeurs d'attributs numériques. Par exemple, une règle de distance par lots de ce type peut exiger que tous les joueurs participant à un match se situent à moins de deux niveaux de compétence l'un de l'autre. Pour ce type, définissez une distance maximale entre tous `batchAttribute` les tickets.
- Comparez les valeurs des attributs de chaîne. Par exemple, une règle de distance par lots de ce type peut exiger que tous les joueurs participant à un match demandent le même mode de jeu. Pour ce type, définissez une `batchAttribute` valeur à FlexMatch utiliser pour former des lots.

Propriétés des règles de distance par lots

- **`batchAttribute`**— La valeur de l'attribut du joueur utilisée pour former des lots.
- **`maxDistance`**— La valeur de distance maximale pour un match réussi. Utilisé pour comparer des attributs numériques.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent les valeurs minimale (`min`), maximale (`max`) et moyenne (`avg`) pour les joueurs d'un ticket. La valeur par défaut est `avg`.

Example

Exemples

```
{
  "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"
}
```

Règle de comparaison

```
comparison
```

Les règles de comparaison comparent la valeur d'un attribut du joueur à une autre valeur. Il existe deux types de règles de comparaison :

- Comparer à la valeur de référence. Par exemple, une règle de comparaison de ce type peut exiger que les joueurs correspondants possèdent un certain niveau de compétence ou plus. Pour ce type, spécifiez un attribut de joueur, une valeur de référence et une opération de comparaison.

- Comparez les joueurs. Par exemple, une règle de comparaison de ce type peut exiger que tous les joueurs du match utilisent des personnages différents. Pour ce type, spécifiez un attribut de joueur et l'opération de comparaison égale (=) ou non égale (!=). Ne spécifiez pas de valeur de référence.

> **Note**
>
> Les règles de distance par lots sont plus efficaces pour comparer les attributs des joueurs. Pour réduire la latence du matchmaking, utilisez une règle de distance par lots lorsque cela est possible.

Propriétés des règles de comparaison

- **`measurements`**— La valeur de l'attribut du joueur à comparer.
- **`referenceValue`**— La valeur à laquelle comparer la mesure pour une correspondance potentielle.
- **`operation`**— La valeur qui détermine comment comparer la mesure à la valeur de référence. Les opérations valides incluent : <<=,=,!=,>,>=.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent les valeurs minimale (`min`), maximale (`max`) et moyenne (`avg`) pour les joueurs d'un ticket. La valeur par défaut est `avg`.

Règle de distance

```
distance
```

Les règles de distance mesurent la différence entre deux valeurs numériques, comme la distance entre les niveaux de compétence des joueurs. Par exemple, une règle de distance peut exiger que tous les joueurs aient joué au jeu pendant au moins 30 heures.

> **Note**
>
> Les règles de distance par lots sont plus efficaces pour comparer les attributs des joueurs. Pour réduire la latence du matchmaking, utilisez une règle de distance par lots lorsque cela est possible.

Propriétés des règles de distance

- **`measurements`**— La valeur de l'attribut du joueur pour laquelle mesurer la distance. Il doit s'agir d'un attribut avec une valeur numérique.
- **`referenceValue`**— La valeur numérique par rapport à laquelle mesurer la distance pour une correspondance potentielle.
- **`minDistance`**/**`maxDistance`**— La valeur de distance minimale ou maximale pour une correspondance réussie.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent les valeurs minimale (`min`), maximale (`max`) et moyenne (`avg`) pour les joueurs d'un ticket. La valeur par défaut est `avg`.

Règle de collecte

```
collection
```

Les règles de collecte comparent les valeurs d'un groupe d'attributs de joueurs à celles des autres joueurs du lot ou à une valeur de référence. Une collection peut contenir des valeurs d'attributs pour plusieurs joueurs, un attribut de joueur sous forme de liste de chaînes, ou les deux. Par exemple, une règle de collection peut porter sur les personnages que les joueurs d'une équipe choisissent. La règle peut alors exiger que l'équipe possède au moins un personnage d'un certain personnage.

Propriétés des règles de collecte

- **`measurements`**— La collection de valeurs d'attributs des joueurs à comparer. Les valeurs des attributs doivent être des listes de chaînes.
- **`referenceValue`**— La valeur (ou la collection de valeurs) à utiliser pour comparer les mesures en vue d'une correspondance prospective.
- **`operation`**— La valeur qui détermine comment comparer un ensemble de mesures. Les opérations valides sont les suivantes :
  - `intersection`— Cette opération mesure le nombre de valeurs identiques dans les collections de tous les joueurs. Pour un exemple de règle utilisant l'opération d'intersection, reportez-vous à la sectionExemple 4 : utiliser un tri explicite pour trouver les meilleures correspondances.
  - `contains`— Cette opération mesure le nombre de collections d'attributs de joueurs contenant la valeur de référence spécifiée. Pour obtenir un exemple de règle utilisant l'opération contains,

reportez-vous à la sectionExemple 3 : définir les exigences et les limites de latence au niveau de l'équipe.

- `reference_intersection_count`— Cette opération mesure le nombre d'éléments d'une collection d'attributs de joueur qui correspondent à des éléments de la collection de valeurs de référence. Vous pouvez utiliser cette opération pour comparer différents attributs de joueurs. Pour un exemple de règle qui compare plusieurs collections d'attributs de joueurs, voirExemple 5 : rechercher des intersections entre plusieurs attributs de joueur.

- **`minCount`/`maxCount`**— La valeur de décompte minimale ou maximale pour une correspondance réussie.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Pour cette valeur, vous pouvez `union` combiner les attributs de tous les joueurs du groupe. Vous pouvez également `intersection` utiliser les attributs des joueurs que le groupe a en commun. La valeur par défaut est `union`.

Règle composée

```
compound
```

Les règles composées utilisent des énoncés logiques pour former des matchs de 40 joueurs ou moins. Vous pouvez utiliser plusieurs règles composées dans un seul ensemble de règles. Lorsque vous utilisez plusieurs règles composées, toutes doivent être vraies pour qu'il y ait correspondance.

Vous ne pouvez pas développer une règle composée à l'aide de règles d'extension, mais vous pouvez développer des règles sous-jacentes ou secondaires.

Propriétés des règles composées

- **`statement`**— La logique utilisée pour combiner des règles individuelles afin de former la règle composée. Les règles que vous spécifiez dans cette propriété doivent avoir été définies plus tôt dans votre ensemble de règles. Vous ne pouvez pas utiliser de `batchDistance` règles dans une règle composée.

  Cette propriété prend en charge les opérateurs logiques suivants :

  - `and`— L'expression est vraie si les deux arguments fournis sont vrais.
  - `or`— L'expression est vraie si l'un des deux arguments fournis est vrai.
  - `not`— Inverse le résultat de l'argument dans l'expression.

- `xor`— L'expression est vraie si un seul des arguments est vrai.

Example Exemple

L'exemple suivant associe des joueurs de différents niveaux de compétence en fonction du mode de jeu qu'ils ont sélectionné.

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

Règle de latence

```
latency
```

Les règles de latence mesurent la latence des joueurs par emplacement. Une règle de latence ignore tout emplacement dont la latence est supérieure au maximum. Un joueur doit avoir une valeur de latence inférieure au maximum à au moins un endroit pour que la règle de latence l'accepte. Vous pouvez utiliser ce type de règle avec des correspondances importantes en spécifiant la `maxLatency` propriété.

Propriétés des règles de latence

- **`maxLatency`**— La valeur de latence maximale acceptable pour un emplacement. Si aucun emplacement d'un ticket ne présente une latence inférieure au maximum, cela signifie que le ticket ne correspond pas à la règle de latence.
- **`maxDistance`**— La valeur maximale entre la latence de chaque ticket et la valeur de référence de distance.
- **`distanceReference`**— La valeur de latence à laquelle comparer la latence des tickets. Les tickets situés dans la limite maximale de la valeur de référence de distance donnent lieu à une correspondance réussie. Les options valides incluent les valeurs de latence minimale (`min`avg) et moyenne () du joueur.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent les valeurs minimale (`min`), maximale (`max`) et moyenne (`avg`) pour les joueurs d'un ticket. La valeur par défaut est `avg`.

> **Note**
>
> Une file d'attente peut placer une session de jeu dans une région qui ne correspond pas à une règle de latence. Pour plus d'informations sur les politiques de latence pour les files d'attente, voir Création d'une politique de latence pour les joueurs.

Règle de tri absolue

```
absoluteSort
```

Les règles de tri absolues trient un lot de tickets de matchmaking en fonction d'un attribut de joueur spécifié par rapport au premier ticket ajouté au lot.

Propriétés des règles de tri absolu

- **`sortDirection`**— L'ordre dans lequel trier les tickets de matchmaking. Les options valides incluent `ascending` et`descending`.
- **`sortAttribute`**— L'attribut du joueur selon lequel trier les tickets.
- **`mapKey`**— Les options permettant de trier l'attribut du joueur s'il s'agit d'une carte. Les options valides sont les suivantes :
  - `minValue`— La clé ayant la valeur la plus faible est la première.
  - `maxValue`— La clé ayant la valeur la plus élevée est la première.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent l'attribut de joueur minimum (`min`), l'attribut de joueur maximum (`max`) et la moyenne (`avg`) de tous les attributs de joueur des joueurs du groupe. La valeur par défaut est `avg`.

Example

Exemple

L'exemple de règle suivant trie les joueurs par niveau de compétence et fait la moyenne du niveau de compétence des groupes.

```
{
```

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

Règle de tri par distance

```
distanceSort
```

Les règles de tri par distance permettent de trier un lot de tickets de matchmaking en fonction de la distance entre un attribut de joueur spécifié et le premier ticket ajouté au lot.

Propriétés des règles de tri de la distance

- **`sortDirection`**— La direction pour trier les tickets de matchmaking. Les options valides incluent `ascending` et`descending`.
- **`sortAttribute`**— L'attribut du joueur selon lequel trier les tickets.
- **`mapKey`**— Les options permettant de trier l'attribut du joueur s'il s'agit d'une carte. Les options valides sont les suivantes :
  - `minValue`— Pour le premier ticket ajouté au lot, recherchez la clé dont la valeur est la plus faible.
  - `maxValue`— Pour le premier ticket ajouté au lot, recherchez la clé dont la valeur est la plus élevée.
- **`partyAggregation`**— La valeur qui détermine le mode FlexMatch de gestion des tickets avec plusieurs joueurs (parties). Les options valides incluent les valeurs minimale (`min`), maximale (`max`) et moyenne (`avg`) pour les joueurs d'un ticket. La valeur par défaut est `avg`.

FlexMatchexpressions de propriété

Les expressions de propriété peuvent être utilisées pour définir certaines propriétés liées au matchmaking. Ils vous permettent d'utiliser des calculs et de la logique pour définir la valeur d'une propriété. Les expressions de propriété prennent généralement l'une des deux formes suivantes :

- Données individuelles des joueurs.
- Collections calculées de données individuelles sur les joueurs.

Expressions de propriétés de matchmaking courantes

Une expression de propriété identifie une valeur spécifique pour un joueur, une équipe ou un match. Les expressions partielle suivantes illustrent comment identifier les équipes et les joueurs :

Objectif	Entrée	Signification	Sortie
Pour identifier une équipe spécifique dans une mise en relation :	`teams[red]`	L'équipe Red	Equipe
Pour identifier un ensemble d'équipes spécifiques lors d'un match :	`teams[red,blue]`	L'équipe rouge et l'équipe bleue	List<Team>
Pour identifier tous les équipes d'une mise en relation :	`teams[*]`	Toutes les équipes	List<Team>
Pour identifier les joueurs d'une équipe spécifique :	`team[red] .players`	Joueurs de l'équipe Red	List<Player>
Pour identifier les joueurs d'un ensemble d'équipes spécifiques lors d'un match :	`team[red, blue].players`	Joueurs de la mise en relation, regroupés par équipe	List<List<Player>>
Pour identifier les joueurs d'une mise en relation :	`team[*].players`	Joueurs de la mise en relation, regroupés par équipe	List<List<Player>>

Exemples d'expressions de propriété

Le tableau suivant illustre certaines expressions de propriété qui s'appuient sur les exemples précédents :

Expression	Signification	Type obtenu
`teams[red].players[playerId]`	Les ID joueur de tous les joueurs de l'équipe Red	List<string>
`teams[red].players.attributes[skill]`	Les attributs « skill » de tous les joueurs de l'équipe Red	List<number>
`teams[red,blue].players.attributes[skill]`	Les attributs de « compétence » de tous les joueurs de l'équipe rouge et de l'équipe bleue, regroupés par équipe	List<List<number>>
`teams[*].players.attributes[skill]`	Les attributs « skill » de tous les joueurs de la mise en relation, regroupés par équipe	List<List<number>>

Agrégations d'expressions de propriétés

Les expressions de propriété peuvent être utilisées pour regrouper les données d'équipe à l'aide des fonctions ou combinaisons de fonctions suivantes :

Agrégation	Entrée	Signification	Sortie
`min`	List<number>	Obtenir le minimum de tous les chiffres de la liste.	nombre
`max`	List<number>	Obtenir le maximum de tous les chiffres de la liste.	nombre

Agrégation	Entrée	Signification	Sortie
`avg`	List<number>	Obtenir la moyenne de tous les chiffres de la liste.	nombre
`median`	List<number>	Obtenir la valeur médiane de tous les chiffres de la liste.	nombre
`sum`	List<number>	Obtenir la somme de tous les chiffres de la liste.	nombre
`count`	List<?>	Obtenir le nombre d'éléments de la liste.	nombre
`stddev`	List<number>	Obtenir l'écart standard de tous les chiffres de la liste.	nombre
`flatten`	List<List<?>>	Transformer une collection de listes imbriquées en une seule liste contenant tous les éléments.	List<?>
`set_intersection`	List<List<string>>	Obtenez une liste de chaînes qui sont disponibles dans toutes les listes de chaînes d'une collection.	List<string>

Agrégation	Entrée	Signification	Sortie
Toutes les opérations ci-dessus	List<List<?>>	Toutes les opération s sur une liste imbriquée fonctionn ent sur chaque sous-liste individuellement pour produire une liste de résultats.	List<?>

Le tableau suivant illustre certaines expressions de propriété valides qui utilisent les fonctions d'agrégation :

Expression	Signification	Type obtenu
flatten(teams[*].players.attributes[skill])	Les attributs « skill » de tous les joueurs de la mise en relation (non regroupés)	List<number>
avg(teams[red].players.attributes[skill])	La compétence moyenne de tous les joueurs de l'équipe Red	nombre
avg (équipes [*] .players.attributes [compéten ce])	La compétence moyenne de chaque équipe de la mise en relation	List<number>
avg(flatten(teams[*].players.attributes[skill]))	Le niveau de compétence moyen de tous les joueurs de la mise en relation. Cette expression obtient une liste mise à plat des compétenc	nombre

Expression	Signification	Type obtenu
	es des joueurs, puis calcule leur moyenne.	
count(teams[red].players)	Nombre de joueurs de l'équipe Red	nombre
count (teams[*].players)	Nombre de joueurs de chaque équipe de la mise en relation	List<number>
max(avg(teams[*].players.attributes[skill]))	Plus haut niveau de compétence d'équipe de la mise en relation	nombre

FlexMatchévénements de matchmaking

Amazon GameLift FlexMatch émet des événements pour chaque ticket de matchmaking au fur et à mesure de son traitement. Vous pouvez publier ces événements dans une rubrique Amazon SNS, comme décrit dansConfigurer les notifications FlexMatch d'événements. Ces événements sont également transmis à Amazon CloudWatch Events en temps quasi réel et dans la mesure du possible.

Cette rubrique décrit la structure des FlexMatch événements et fournit un exemple pour chaque type d'événement. Pour plus d'informations sur les statuts des tickets de matchmaking, consultez MatchmakingTicketla référence des GameLift API Amazon.

MatchmakingSearching

La demande a été saisie dans la mise en relation. Cela inclut les nouvelles demandes et les demandes qui faisaient partie d'une mise en relation qui a échoué.

Ressource : ConfigurationArn

Détail : type, billetsestimatedWaitMillis, gameSessionInfo

Exemple

```
{
```

```
  "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

Une mise en relation potentielle a été créée. Le ticket est émis pour toutes les nouvelles mises en relation potentielles, que l'acceptation soit obligatoire ou non.

Ressource : ConfigurationArn

Détails : type, tickets, délai d'acceptation, acceptation requise,, ID de correspondance ruleEvaluationMetrics gameSessionInfo

Exemple


```
{
  "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

Les joueurs ont accepté une mise en relation potentielle. Cet événement contient l'état d'acceptation actuel de chaque joueur de la mise en relation. Des données manquantes signifient qu'AcceptMatchaucun appel n'a été effectué pour ce joueur.

Ressource : ConfigurationArn

Détails : type, billets, MatchID, gameSessionInfo

Exemple

```
{
  "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

L'acceptation de la mise en relation est terminée, du fait de l'acceptation par un joueur, du rejet par un joueur ou de l'expiration de l'acceptation.

Ressource : ConfigurationArn

Détails : type, tickets, acceptation, MatchID, gameSessionInfo

Exemple

```
{
  "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

La mise en relation a été réalisée avec succès et une session de jeu a été créée.

Ressource : ConfigurationArn

Détails : type, billets, MatchID, gameSessionInfo

Exemple

```
{
  "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

La demande de mise en relation a échoué car elle a expiré.

Ressource : ConfigurationArn

Détail : type, tickets, messageruleEvaluationMetrics, MatchID, gameSessionInfo

Exemple

```
{
  "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

La demande de mise en relation a été annulée.

Ressource : ConfigurationArn

Détail : type, tickets, messageruleEvaluationMetrics, MatchID, gameSessionInfo

Exemple

```
{
  "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

La demande de mise en relation a rencontré une erreur. Cette situation peut être due au fait que la file d'attente de sessions de jeu n'est pas accessible ou à une erreur interne.

Ressource : ConfigurationArn

Détail : type, tickets, messageruleEvaluationMetrics, MatchID, gameSessionInfo

Exemple

```
{
```

```
  "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"
  }
}
```

Sécurité avec FlexMatch

Chez AWS, la sécurité dans le cloud est notre priorité numéro 1. En tant que client AWS, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des organisations les plus pointilleuses en termes de sécurité.

La sécurité est une responsabilité partagée entre AWS et vous-même. Pour plus d'informations sur la manière d'appliquer le modèle de responsabilité partagée lors de l'utilisationFlexMatch, consultez la section Sécurité sur Amazon GameLift.

Notes GameLift FlexMatch de mise à jour d'Amazon et versions du SDK

Les notes GameLift de mise à jour d'Amazon fournissent des détails sur les nouvelles FlexMatch fonctionnalités, les mises à jour et les correctifs liés au service. Cette page inclut également l'historique des versions d'Amazon GameLift SDK.

Ressources GameLift pour développeurs Amazon

Pour consulter toute GameLift la documentation Amazon et les ressources destinées aux développeurs, consultez la page d'accueil de GameLiftla documentation Amazon.

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le Glossaire AWS dans la Référence Glossaire AWS.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.