aws

FlexMatch Panduan Pengembang

Amazon GameLift

Versi

Amazon GameLift: FlexMatch Panduan Pengembang

Table of Contents

Apa itu Amazon GameLift FlexMatch?

Amazon GameLift FlexMatch adalah layanan perjodohan yang dapat disesuaikan untuk game multipemain. Dengan FlexMatch, Anda dapat membangun seperangkat aturan khusus yang menentukan seperti apa pertandingan multipemain untuk game Anda, dan menentukan cara mengevaluasi dan memilih pemain yang kompatibel untuk setiap pertandingan. Anda juga dapat menyempurnakan aspek-aspek kunci dari algoritme perjodohan agar sesuai dengan kebutuhan game Anda.

Gunakan FlexMatch sebagai layanan perjodohan mandiri atau terintegrasi dengan solusi hosting GameLift game Amazon. Misalnya, Anda dapat menerapkan FlexMatch sebagai fitur mandiri dengan game dengan peer-to-peer arsitektur atau game yang menggunakan solusi komputasi cloud lainnya. Atau Anda dapat menambahkan FlexMatch ke hosting EC2 GameLift terkelola Amazon atau hosting lokal dengan Amazon. GameLift Anywhere Panduan ini memberikan informasi terperinci tentang cara membangun sistem FlexMatch perjodohan untuk skenario khusus Anda.

FlexMatch memberi Anda fleksibilitas untuk menetapkan prioritas perjodohan tergantung pada persyaratan permainan Anda. Misalnya, Anda dapat melakukan hal berikut:

- Menemukan keseimbangan antara kecepatan dan kualitas kecocokan. Tetapkan aturan pertandingan untuk menemukan pertandingan yang cukup baik dengan cepat, atau minta pemain menunggu sedikit lebih lama untuk menemukan kecocokan terbaik untuk pengalaman pemain yang optimal.
- Membuat kecocokan berdasarkan pemain yang cocok atau tim yang cocok. Buat pertandingan di mana semua pemain memiliki karakteristik yang sama seperti keterampilan atau pengalaman. Atau bentuk pertandingan di mana karakteristik gabungan masing-masing tim memenuhi kriteria umum.
- Prioritaskan bagaimana faktor latensi pemain menjadi perjodohan. Apakah Anda ingin menetapkan batas latensi untuk semua pemain, atau apakah latensi yang lebih tinggi dapat diterima selama semua orang dalam pertandingan memiliki latensi yang sama?

> Siap untuk mulai bekerja dengan FlexMatch?
>
> Untuk step-by-step panduan tentang cara membuat game Anda aktif dan berjalan FlexMatch, lihat topik berikut:
>
> - Tutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon

- Tutorial: Integrasikan FlexMatch dengan perjodohan mandiri

FlexMatch Fitur utama

Fitur-fitur berikut tersedia dengan semua FlexMatch skenario, baik Anda gunakan FlexMatch sebagai layanan mandiri atau dengan hosting GameLift game Amazon.

- Pencocokan pemain yang dapat disesuaikan. Rancang dan bangun pembuat kecocokan agar sesuai dengan semua mode game yang Anda tawarkan kepada pemain Anda. Buat seperangkat aturan khusus untuk mengevaluasi atribut pemain utama (seperti tingkat keterampilan atau peran) dan data latensi geografis untuk membentuk kecocokan pemain yang hebat untuk game Anda.
- Pencocokan berbasis latensi. Memberikan data latensi pemain dan membuat aturan kecocokan yang mengharuskan pemain dalam kecocokan memiliki waktu respons yang sama. Fitur ini berguna ketika kolam pencocokan pemain Anda mencakup beberapa wilayah geografis.
- Support untuk ukuran pertandingan hingga 200 pemain. Buat kecocokan hingga 40 pemain menggunakan aturan pertandingan yang disesuaikan untuk game Anda. Buat kecocokan hingga 200 pemain menggunakan proses pencocokan yang menggunakan proses pencocokan khusus yang disederhanakan agar waktu tunggu pemain dapat dikelola.
- Penerimaan pemain. Meminta pemain untuk ikut serta dalam kecocokan yang diusulkan sebelum menyelesaikan kecocokan dan memulai sesi game. Gunakan fitur ini untuk memulai alur kerja penerimaan kustom Anda dan melaporkan respons pemain FlexMatch sebelum menempatkan sesi permainan baru untuk pertandingan. Jika tidak semua pemain menerima pertandingan, pertandingan yang diusulkan gagal dan pemain yang menerima secara otomatis kembali ke kolam perjodohan.
- Dukungan pihak pemain. Menghasilkan kecocokan untuk grup pemain yang ingin bermain bersama di tim yang sama. Gunakan FlexMatch untuk menemukan pemain tambahan untuk mengisi pertandingan sesuai kebutuhan.
- Aturan pencocokan yang dapat diperluas. Secara bertahap kendurkan persyaratan kecocokan setelah beberapa waktu berlalu tanpa menemukan kecocokan yang berhasil. Perluasan aturan memungkinkan Anda memutuskan di mana dan kapan harus melonggarkan aturan pertandingan awal, sehingga pemain dapat masuk ke game yang dapat dimainkan lebih cepat.
- Isi ulang pertandingan. Isi slot pemain yang kosong di sesi game yang ada dengan pemain baru yang cocok. Sesuaikan kapan dan bagaimana meminta pemain baru, dan gunakan aturan pertandingan khusus yang sama untuk menemukan pemain tambahan.

FlexMatch dengan Amazon GameLift Hosting

FlexMatch menawarkan fitur tambahan berikut untuk digunakan dengan game yang Anda hosting dengan Amazon GameLift. Ini termasuk game dengan server game khusus atau Server Realtime.

- Penempatan sesi permainan. Ketika pertandingan berhasil dibuat, FlexMatch secara otomatis meminta penempatan sesi permainan baru dari Amazon GameLift. Data yang dihasilkan selama perjodohan, termasuk ID pemain dan tugas tim, diberikan ke server game sehingga dapat menggunakan informasi tersebut untuk memulai sesi permainan untuk pertandingan. FlexMatch kemudian meneruskan kembali informasi koneksi sesi game sehingga klien game dapat bergabung dengan game. Untuk meminimalkan latensi yang dialami pemain dalam pertandingan, penempatan sesi permainan dengan Amazon juga GameLift dapat menggunakan data latensi pemain regional, jika disediakan.
- Isi ulang pertandingan otomatis. Dengan fitur ini diaktifkan, FlexMatch secara otomatis mengirimkan permintaan pengisian ulang pertandingan ketika sesi permainan baru dimulai dengan slot pemain yang tidak terisi. Sistem perjodohan Anda memulai proses penempatan sesi permainan dengan jumlah pemain minimum, dan kemudian dengan cepat mengisi slot yang tersisa. Anda tidak dapat menggunakan pengisian ulang otomatis untuk menggantikan pemain yang keluar dari sesi permainan yang cocok.

Jika Anda menggunakan Amazon GameLift FleetiQ dengan game yang di-host dengan sumber daya Amazon Elastic Compute Cloud (Amazon EC2), terapkan sebagai layanan mandiri. FlexMatch

Harga untuk Amazon GameLift FlexMatch

Amazon GameLift mengenakan biaya untuk instans berdasarkan durasi penggunaan dan untuk bandwidth berdasarkan jumlah data yang ditransfer. Jika Anda meng-host game Anda di GameLift server Amazon, FlexMatch penggunaan sudah termasuk dalam biaya untuk Amazon GameLift. Jika Anda meng-host game Anda di solusi server lain, FlexMatch penggunaan dibebankan secara terpisah. Untuk daftar lengkap biaya dan harga Amazon GameLift, lihat GameLift Harga Amazon.

Untuk informasi tentang menghitung biaya hosting game atau perjodohan Anda dengan Amazon GameLift, lihat Menghasilkan perkiraan GameLift harga Amazon, yang menjelaskan cara menggunakan. AWS Pricing Calculator

Bagaimana Amazon GameLift FlexMatch bekerja

Topik ini memberikan gambaran umum tentang GameLift FlexMatch layanan Amazon, termasuk komponen inti dari suatu FlexMatch sistem dan bagaimana mereka berinteraksi.

Anda dapat menggunakan FlexMatch game yang menggunakan hosting GameLift terkelola Amazon atau dengan game yang menggunakan solusi hosting lain. Game yang di-host di GameLift server Amazon, termasuk Server Realtime, menggunakan GameLift layanan Amazon terintegrasi untuk secara otomatis menemukan server game yang tersedia dan memulai sesi permainan untuk pertandingan. Game yang digunakan FlexMatch sebagai layanan mandiri, termasuk Amazon GameLift FleetiQ, harus berkoordinasi dengan sistem hosting yang ada untuk menetapkan sumber daya hosting dan memulai sesi permainan untuk pertandingan.

Untuk panduan terperinci tentang pengaturan FlexMatch untuk game Anda, lihatMemulai dengan FlexMatch.

Komponen matchmaking

Sistem FlexMatch perjodohan mencakup beberapa atau semua komponen berikut.

GameLift Komponen Amazon

Ini adalah GameLift sumber daya Amazon yang mengontrol bagaimana FlexMatch layanan melakukan perjodohan untuk game Anda. Mereka dibuat dan dipelihara menggunakan GameLift alat Amazon, termasuk konsol dan AWS CLI atau, sebagai alternatif, secara terprogram menggunakan untuk AWS SDK Amazon. GameLift

- FlexMatch Konfigurasi perjodohan (juga disebut mak comblang) — Mak comblang adalah seperangkat nilai konfigurasi yang menyesuaikan proses perjodohan untuk game Anda. Sebuah game dapat memiliki beberapa matchmaker, masing-masing dikonfigurasi untuk mode game yang berbeda atau pengalaman yang diperlukan. Saat game Anda mengirim permintaan perjodohan FlexMatch, itu menentukan mak comblang mana yang akan digunakan.
- FlexMatch Set aturan perjodohan — Sebuah set aturan berisi semua informasi yang diperlukan untuk mengevaluasi pemain untuk pertandingan potensial dan menyetujui atau menolak. Rangkaian aturan mendefinisikan struktur tim match, menyatakan atribut pemain yang digunakan untuk evaluasi, dan menyediakan aturan yang menggambarkan kriteria untuk match yang dapat diterima. Aturan dapat berlaku untuk masing-masing pemain, tim, atau seluruh match. Misalnya, aturan mungkin mengharuskan setiap pemain dalam match memilih peta game yang sama, atau mungkin mengharuskan semua tim memiliki rata-rata keterampilan pemain yang sama.

- Antrian sesi GameLift game Amazon (hanya FlexMatch dengan hosting GameLift terkelola Amazon) - Antrian sesi game menemukan sumber daya hosting yang tersedia dan memulai sesi permainan baru untuk pertandingan. Konfigurasi antrian menentukan di mana Amazon GameLift mencari sumber daya hosting yang tersedia dan cara memilih host terbaik yang tersedia untuk kecocokan.

Komponen kustom

Komponen berikut mencakup fungsionalitas yang diperlukan untuk FlexMatch sistem lengkap yang harus Anda terapkan berdasarkan arsitektur game Anda.

- Antarmuka pemain untuk matchmaking — Antarmuka ini memungkinkan pemain untuk bergabung dengan match. Minimal, ini memulai permintaan matchmaking melalui komponen layanan matchmaking client dan menyediakan data spesifik pemain, seperti tingkat keterampilan dan data latensi sesuai kebutuhan, untuk proses matchmaking.

  > **Note**
  >
  > Sebagai praktik terbaik, komunikasi dengan FlexMatch layanan harus dilakukan oleh layanan backend, bukan dari klien game.

- Layanan perjodohan klien — Layanan ini memenuhi permintaan pemain bergabung dari antarmuka pemain, menghasilkan permintaan perjodohan, dan mengirimkannya ke layanan. FlexMatch Untuk permintaan dalam proses, ini memonitor kejadian matchmaking, melacak status matchmaking, dan mengambil tindakan yang diperlukan. Tergantung pada bagaimana Anda mengelola sesi game hosting dalam game Anda, layanan ini dapat mengembalikan informasi koneksi sesi game kembali ke pemain. Komponen ini menggunakan AWS SDK dengan Amazon GameLift API untuk berkomunikasi dengan FlexMatch layanan.
- Layanan penempatan pertandingan (hanya untuk FlexMatch layanan mandiri) - Komponen ini bekerja dengan sistem hosting game Anda yang ada untuk menemukan sumber daya hosting yang tersedia dan memulai sesi permainan baru untuk pertandingan. Komponen harus mendapatkan hasil perjodohan dan mengekstrak informasi yang diperlukan untuk memulai sesi permainan baru, termasuk pemainIDs, atribut, dan tugas tim untuk semua pemain dalam pertandingan.

FlexMatch proses perjodohan

Topik ini menjelaskan skenario perjodohan dasar dan interaksi antara berbagai komponen permainan Anda dan layanan. FlexMatch

Meminta matchmaking untuk pemain

Seorang pemain yang menggunakan client game Anda mengklik tombol "Join Game". Tindakan ini menyebabkan layanan perjodohan klien Anda mengirim permintaan perjodohan. FlexMatch Permintaan mengidentifikasi FlexMatch mak comblang yang akan digunakan saat memenuhi permintaan. Permintaan juga mencakup informasi pemain yang dibutuhkan matchmaker kustom Anda, seperti tingkat keterampilan, preferensi bermain, atau data latensi geografis. Anda dapat membuat permintaan matchmaking untuk satu pemain atau beberapa pemain.

Menambahkan permintaan ke kolam matchmaking

Ketika FlexMatch menerima permintaan perjodohan, itu menghasilkan tiket perjodohan dan menambahkannya ke kumpulan tiket mak comblang. Tiket tetap berada di kolam sampai menemukan match atau batas waktu maksimum tercapai. Layanan matchmaking klien Anda secara berkala dinotifikasi tentang acara matchmaking, termasuk perubahan status tiket.

Membangun match

Mak FlexMatch comblang Anda terus menjalankan proses berikut pada semua tiket di kolamnya:

1. Matchmaker mengurutkan kolam berdasarkan usia tiket, kemudian mulai membangun match potensial dimulai dengan tiket tertua.
2. Matchmaker menambahkan tiket kedua ke match potensial dan mengevaluasi hasilnya terhadap aturan matchmaking kustom Anda. Jika match potensial lolos evaluasi, pemain tiket ditugaskan ke tim.
3. Matchmaker menambahkan tiket berikutnya secara berurutan dan mengulangi proses evaluasi. Ketika semua slot pemain telah terisi, match sudah siap.

Matchmaking untuk match besar (41 hingga 200 pemain) menggunakan versi modifikasi dari proses yang dijelaskan di atas sehingga dapat membangun match dalam jangka waktu yang wajar. Alih-alih mengevaluasi setiap tiket secara individual, matchmaker membagi kumpulan tiket yang telah diurutkan sebelumnya menjadi match potensial, lalu menyeimbangkan setiap match berdasarkan karakteristik pemain yang telah Anda tentukan. Misalnya, matchmaker mungkin melakukan pra sortir tiket berdasarkan lokasi latensi rendah yang serupa, dan kemudian menggunakan penyeimbangan pasca match untuk memastikan bahwa tim disesuaikan secara merata dengan keterampilan pemain.

Laporan hasil matchmaking

Ketika match yang dapat diterima ditemukan, semua tiket yang cocok diperbarui dan kejadian matchmaking sukses dihasilkan untuk setiap tiket yang cocok.

- FlexMatch sebagai layanan mandiri: Game Anda menerima hasil pertandingan dalam acara perjodohan yang sukses. Data hasil mencakup daftar semua pemain yang cocok dan penugasan tim. Jika permintaan match berisi info latensi pemain, hasilnya juga menyarankan lokasi geografis yang optimal untuk match tersebut.
- FlexMatch dengan solusi GameLift hosting Amazon: Hasil pertandingan secara otomatis diteruskan ke GameLift antrian Amazon untuk penempatan sesi game. Matchmaker menentukan antrean yang digunakan untuk penempatan sesi game.

Memulai sesi game untuk match

Setelah match yang diusulkan berhasil dibentuk, sesi game baru dimulai. Server game Anda harus dapat menggunakan data hasil perjodohan, termasuk tugas pemain IDs dan tim, saat menyiapkan sesi permainan untuk pertandingan.

- FlexMatch sebagai layanan mandiri: Layanan penempatan pertandingan kustom Anda mendapatkan data hasil pertandingan dari acara perjodohan yang berhasil, dan terhubung ke sistem penempatan sesi game yang ada untuk menemukan sumber daya hosting yang tersedia untuk pertandingan. Setelah sumber daya hosting ditemukan, layanan penempatan match berkoordinasi dengan sistem hosting yang ada untuk memulai sesi game baru dan memperoleh informasi koneksi.
- FlexMatch dengan solusi GameLift hosting Amazon: Antrian sesi permainan menemukan server game terbaik yang tersedia untuk pertandingan. Tergantung pada bagaimana antrian dikonfigurasi, ia mencoba untuk menempatkan sesi game dengan sumber daya berbiaya terendah dan di mana pemain akan mengalami latensi rendah (jika data latensi pemain disediakan). Setelah sesi permainan berhasil ditempatkan, GameLift layanan Amazon meminta server game untuk memulai sesi permainan baru, meneruskan hasil perjodohan dan data game opsional lainnya.

Menghubungkan pemain ke match

Setelah sesi game dimulai, pemain terhubung ke sesi, mengklaim penugasan tim mereka, dan memulai gameplay.

- FlexMatch sebagai layanan mandiri: Game Anda menggunakan sistem manajemen sesi permainan yang ada untuk memberikan informasi koneksi kembali ke pemain.

- FlexMatch dengan solusi GameLift hosting Amazon: Pada penempatan sesi game yang sukses, FlexMatch perbarui semua tiket yang cocok dengan informasi koneksi sesi permainan dan ID sesi pemain.

FlexMatch didukung Wilayah AWS

Jika Anda menggunakan FlexMatch solusi GameLift hosting Amazon, Anda dapat menyelenggarakan sesi permainan yang cocok di lokasi mana pun Anda menghosting game. Lihat daftar lengkap Wilayah AWS dan lokasi untuk GameLift hosting Amazon.

Untuk semua FlexMatch pengguna, Anda dapat meng-host FlexMatch sumber daya, termasuk konfigurasi perjodohan dan kumpulan aturan, dalam yang didukung berikut ini. Wilayah AWS Lihat Pilih lokasi untuk mak comblang.

Wilayah AWS nama	Kode Wilayah
US East (Northern Virginia)	us-east-1
AS Barat (Oregon)	us-west-2
Asia Pasifik (Seoul)	ap-utara-2
Asia Pasifik (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Eropa (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
China (Wilayah Beijing dan Ningxia)	

Memulai dengan FlexMatch

Gunakan sumber daya di bagian ini untuk membantu Anda memulai membangun sistem perjodohan. FlexMatch

Topik

- Tutorial: Mengatur FlexMatch
- Tutorial: Integrasikan FlexMatch dengan perjodohan mandiri
- Tutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon

Tutorial: Mengatur FlexMatch

Amazon GameLift FlexMatch adalah AWS layanan, dan Anda harus memiliki AWS akun untuk menggunakan layanan ini. Membuat AWS akun gratis. Untuk informasi selengkapnya tentang apa yang dapat Anda lakukan dengan akun AWS , lihat Memulai dengan AWS.

Jika Anda menggunakan FlexMatch GameLift solusi Amazon lainnya, lihat topik berikut:

- Menyiapkan akses untuk GameLift hosting Amazon dan Server Realtime
- Menyiapkan akses untuk hosting di Amazon EC2 dengan Amazon GameLift FleetiQ

Untuk mengatur akun Anda untuk Amazon GameLift

1. Dapatkan akun. Buka Amazon Web Services lalu pilih Sign In ke Konsol. Ikuti petunjuk untuk membuat akun baru atau masuk ke akun yang sudah ada.
2. Siapkan grup pengguna administratif. Buka konsol layanan AWS Identity and Access Management (IAM) dan ikuti langkah-langkah untuk membuat atau memperbarui pengguna atau grup pengguna. IAMmengelola akses ke AWS layanan dan sumber daya Anda. Setiap orang yang mengakses FlexMatch sumber daya Anda, menggunakan GameLift konsol Amazon atau dengan menelepon Amazon GameLift APIs, harus diberikan akses eksplisit. Untuk petunjuk mendetail tentang penggunaan konsol (AWS CLIatau alat lainnya) untuk menyiapkan grup pengguna, lihat Membuat IAM Pengguna.
3. Lampirkan kebijakan izin ke pengguna atau grup pengguna Anda.Akses ke AWS layanan dan sumber daya dikelola dengan melampirkan IAMkebijakan ke pengguna atau grup pengguna.

Kebijakan izin menentukan serangkaian AWS layanan dan tindakan yang harus diakses pengguna.

Untuk Amazon GameLift, Anda harus membuat kebijakan izin khusus dan melampirkannya ke setiap pengguna atau grup pengguna. Kebijakan adalah JSON dokumen. Gunakan contoh di bawah ini untuk membuat kebijakan Anda.

Contoh berikut mengilustrasikan kebijakan izin sebaris dengan izin administratif untuk semua sumber daya dan tindakan Amazon GameLift . Anda dapat memilih untuk membatasi akses dengan menentukan hanya item FlexMatch -spesifik.

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

Tutorial: Integrasikan FlexMatch dengan perjodohan mandiri

Topik ini menguraikan proses integrasi lengkap untuk diterapkan FlexMatch sebagai layanan perjodohan mandiri. Gunakan proses ini jika game multipemain Anda di-host menggunakan perangkat keras lokal yang dikonfigurasi khusus peer-to-peer, atau primitif komputasi awan lainnya. Proses ini juga untuk digunakan dengan Amazon GameLift FleetiQ, yang merupakan solusi pengoptimalan hosting untuk game yang di-host di Amazon. EC2 Jika Anda menghosting game menggunakan hosting GameLift terkelola Amazon (termasuk Server Realtime), lihatTutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon.

Sebelum memulai integrasi, Anda harus memiliki AWS akun dan mengatur izin akses untuk GameLift layanan Amazon. Untuk detailnya, lihat Tutorial: Mengatur FlexMatch. Semua tugas penting yang terkait dengan membuat dan mengelola mak GameLift FlexMatch comblang Amazon dan set aturan dapat dilakukan menggunakan konsol Amazon GameLift .

1. Buat FlexMatch set aturan perjodohan. Rangkaian aturan kustom Anda menyediakan petunjuk lengkap tentang cara membuat kecocokan. Di dalamnya, Anda menentukan struktur dan ukuran

masing-masing tim. Anda juga memberikan serangkaian persyaratan yang harus dipenuhi pertandingan agar valid, yang FlexMatch digunakan untuk memasukkan atau mengecualikan pemain dalam pertandingan. Persyaratan ini mungkin berlaku untuk pemain individu. Anda juga dapat menyesuaikan FlexMatch algoritme dalam kumpulan aturan, seperti membangun pertandingan besar dengan hingga 200 pemain. Lihat topik berikut:

- Bangun seperangkat FlexMatch aturan
- FlexMatch contoh set aturan

2. Siapkan notifikasi untuk acara perjodohan. Gunakan notifikasi untuk melacak aktivitas FlexMatch perjodohan, termasuk status permintaan pencocokan yang tertunda. Karena permintaan perjodohan tidak sinkron, Anda memerlukan cara untuk melacak status permintaan. Karena permintaan pencocokan tidak sinkron, Anda memerlukan cara untuk melacak status permintaan. Menggunakan notifikasi adalah pilihan yang lebih disukai untuk ini. Lihat topik berikut:

   - Siapkan pemberitahuan FlexMatch acara
   - FlexMatch acara perjodohan

3. Siapkan konfigurasi FlexMatch perjodohan. Juga disebut pembuat kecocokan, komponen ini menerima permintaan pencocokan dan memprosesnya. Anda mengonfigurasi pembuat kecocokan dengan menentukan set aturan, target notifikasi, dan waktu tunggu maksimum. Anda juga dapat mengaktifkan fitur opsional. Lihat topik berikut:

   - Desain FlexMatch mak comblang
   - Membuat konfigurasi pencocokan

4. Bangun layanan perjodohan klien. Buat atau perluas layanan klien game dengan fungsionalitas untuk membangun dan mengirim permintaan perjodohan. FlexMatch Untuk membuat permintaan pencocokan, komponen ini harus memiliki mekanisme untuk mendapatkan data pemain yang diperlukan oleh set aturan pencocokan dan, secara opsional, informasi latensi regional. Itu juga harus memiliki metode untuk membuat dan menetapkan tiket unik IDs untuk setiap permintaan. Anda juga dapat memilih untuk membangun alur kerja penerimaan pemain yang mengharuskan pemain untuk ikut serta dalam kecocokan yang diusulkan. Layanan ini juga harus memantau peristiwa pencocokan untuk mendapatkan hasil kecocokan dan memulai penempatan sesi game untuk kecocokan yang berhasil. Lihat topik ini:

   - Tambahkan FlexMatch ke klien game

5. Bangun layanan penempatan pertandingan. Buat mekanisme yang bekerja dengan sistem hosting game Anda yang ada untuk menemukan sumber daya hosting yang tersedia

dan memulai sesi game baru untuk kecocokan yang berhasil. Komponen ini harus dapat menggunakan informasi hasil kecocokan untuk mendapatkan server game yang tersedia dan memulai sesi game baru untuk kecocokan tersebut. Anda mungkin juga ingin menerapkan alur kerja untuk membuat permintaan backfill kecocokan, yang menggunakan pencocokan untuk mengisi slot terbuka di sesi game yang cocok yang sudah berjalan.

Tutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon

FlexMatch tersedia dengan GameLift hosting Amazon terkelola untuk server game khusus dan Server Realtime. Untuk menambahkan FlexMatch perjodohan ke game Anda, selesaikan tugas-tugas berikut.

- Siapkan mak comblang. Pembuat kecocokan menerima permintaan pencocokan dari pemain dan memprosesnya. Pembuat kecocokan mengelompokkan pemain berdasarkan set aturan yang ditentukan dan, untuk setiap kecocokan yang berhasil, membuat sesi game dan sesi pemain baru. Ikuti langkah berikut untuk menyiapkan pembuat kecocokan:
  - Buat set aturan. Satu set aturan memberitahu pembuat kecocokan tentang cara membangun kecocokan yang valid. Set ini menentukan struktur tim dan menentukan cara mengevaluasi pemain untuk dimasukkan dalam kecocokan. Lihat topik berikut:
    - Bangun seperangkat FlexMatch aturan
    - FlexMatch contoh set aturan
  - Buat antrian sesi permainan. Antrean menempatkan wilayah terbaik untuk setiap kecocokan dan menciptakan sesi game baru di wilayah itu. Gunakan antrean yang sudah ada atau buat yang baru untuk pencocokan. Lihat topik ini:
    - Buat antrian
  - Siapkan notifikasi (opsional). Karena permintaan perjodohan tidak sinkron, Anda memerlukan cara untuk melacak status permintaan. Pemberitahuan adalah opsi yang lebih disukai. Lihat topik ini:
    - Siapkan pemberitahuan FlexMatch acara
  - Konfigurasikan mak comblang. Setelah Anda memiliki set aturan, antrean, dan target notifikasi, buatlah konfigurasi untuk pembuat kecocokan Anda. Lihat topik berikut:
    - Desain FlexMatch mak comblang
    - Membuat konfigurasi pencocokan

- Integrasikan FlexMatch ke dalam layanan klien game Anda. Tambahkan fungsionalitas ke layanan klien game Anda untuk memulai sesi game baru dengan pencocokan. Permintaan untuk pencocokan menentukan pembuat kecocokan mana yang akan digunakan dan memberikan data pemain yang diperlukan untuk kecocokan tersebut. Lihat topik ini:
  - Tambahkan FlexMatch ke klien game
- Integrasikan FlexMatch ke server game Anda. Menambahkan fungsionalitas ke server game Anda untuk memulai sesi game yang dibuat melalui pencocokan. Permintaan untuk jenis sesi game ini mencakup informasi khusus kecocokan, termasuk pemain dan tugas tim. Server game perlu mengakses dan menggunakan informasi ini saat membuat sesi game untuk pencocokan. Lihat topik ini:
  - Tambahkan FlexMatch ke server game yang GameLift dihosting Amazon
- Siapkan FlexMatch isi ulang (opsional). Meminta pemain tambahan yang cocok untuk mengisi slot pemain yang terbuka di game yang ada. Anda dapat mengaktifkan pengisian ulang otomatis agar Amazon GameLift mengelola permintaan pengisian ulang. Atau Anda dapat mengelola backfill secara manual dengan menambahkan fungsionalitas ke layanan klien game atau server game Anda untuk memulai permintaan pengisian ulang yang cocok. Lihat topik ini:
  - Isi ulang game yang ada dengan FlexMatch

> **Note**
>
> FlexMatch backfill saat ini tidak tersedia untuk game yang menggunakan Server Realtime.

Membangun mak comblang Amazon GameLift FlexMatch

Proses FlexMatch mak comblang melakukan pekerjaan membangun pertandingan game. Ia mengelola kumpulan permintaan matchmaking yang diterima, membentuk tim untuk match, memproses dan memilih pemain untuk menemukan grup pemain terbaik, dan memulai proses menempatkan dan memulai sesi game untuk match. Topik ini menjelaskan aspek utama matchmaker dan cara mengonfigurasinya sesuai game Anda.

Untuk penjelasan rinci tentang bagaimana FlexMatch mak comblang memproses permintaan perjodohan yang diterimanya, lihat. FlexMatch proses perjodohan

Topik

Desain FlexMatch mak comblang

Topik ini memberikan panduan tentang cara mendesain pembuat kecocokan yang sesuai dengan game Anda.

Topik

Mengonfigurasikan pembuat kecocokan dasar

Setidaknya, pembuat kecocokan membutuhkan elemen-elemen berikut:

- Kumpulan aturan menentukan ukuran dan cakupan tim untuk kecocokan dan menetapkan sekumpulan aturan untuk digunakan saat mengevaluasi pemain untuk kecocokan. Setiap pembuat kecocokan dikonfigurasi untuk menggunakan satu kumpulan aturan. Lihat Bangun seperangkat FlexMatch aturan dan FlexMatch contoh set aturan.

- Target notifikasi menerima semua notifikasi peristiwa pencocokan. Anda perlu menyiapkan topik Amazon Simple Notification Service (SNS) dan kemudian menambahkan ID topik ke mak comblang. Lihat informasi selengkapnya tentang menyiapkan notifikasi di Siapkan pemberitahuan FlexMatch acara.
- Waktu tunggu permintaan menentukan berapa lama permintaan pencocokan dapat tetap berada di kolam permintaan dan dievaluasi untuk kemungkinan kecocokan. Setelah waktu permintaan habis, permintaan tersebut gagal membuat kecocokan dan dikeluarkan dari kolam.
- Saat menggunakan FlexMatch hosting GameLift terkelola Amazon, antrian sesi permainan menemukan sumber daya terbaik yang tersedia untuk menyelenggarakan sesi permainan untuk pertandingan, dan memulai sesi permainan baru. Setiap antrian dikonfigurasi dengan daftar lokasi dan jenis sumber daya (termasuk Instans Spot atau Sesuai Permintaan) yang menentukan tempat sesi permainan dapat ditempatkan. Untuk informasi selengkapnya tentang antrian, lihat Menggunakan antrian multi-lokasi.

Pilih lokasi untuk mak comblang

Tentukan di mana Anda ingin aktivitas perjodohan berlangsung dan buat konfigurasi perjodohan dan aturan yang ditetapkan di lokasi tersebut. Amazon GameLift mengelola kumpulan tiket untuk permintaan pertandingan game Anda di mana mereka diurutkan dan dievaluasi untuk pertandingan yang layak. Setelah membuat pertandingan, Amazon GameLift mengirimkan detail pertandingan untuk penempatan sesi permainan. Anda dapat menjalankan sesi permainan yang cocok di lokasi mana pun yang didukung oleh solusi hosting Anda.

Lihat FlexMatch didukung Wilayah AWS lokasi tempat Anda dapat membuat FlexMatch sumber daya.

Saat memilih Wilayah AWS untuk mak comblang Anda, pertimbangkan bagaimana lokasi dapat memengaruhi kinerja dan bagaimana hal itu dapat mengoptimalkan pengalaman pertandingan untuk pemain. Kami merekomendasikan praktik terbaik berikut:

- Tempatkan mak comblang di lokasi yang dekat dengan pemain Anda dan layanan klien Anda yang mengirimkan permintaan FlexMatch perjodohan. Pendekatan ini mengurangi efek latensi pada alur kerja permintaan pencocokan Anda dan membuatnya lebih efisien.
- Jika game Anda menjangkau audiens global, pertimbangkan untuk membuat mak comblang di beberapa lokasi dan merutekan permintaan pertandingan ke mak comblang yang paling dekat dengan pemain. Selain meningkatkan efisiensi, hal ini juga menyebabkan kolam tiket terbentuk dengan pemain yang secara geografis berdekatan satu sama lain, yang mana akan meningkatkan kemampuan pembuat kecocokan untuk mencocokkan pemain berdasarkan persyaratan latensi.

- Saat menggunakan FlexMatch hosting GameLift terkelola Amazon, tempatkan mak comblang dan antrean sesi permainan yang digunakannya di lokasi yang sama. Ini membantu meminimalkan latensi komunikasi antara pembuat kecocokan dan antrean.

Menambahkan elemen opsional

Selain persyaratan minimum ini, Anda dapat mengonfigurasi pembuat kecocokan Anda dengan opsi tambahan berikut. Jika Anda menggunakan FlexMatch dengan solusi GameLift hosting Amazon, banyak fitur bawaan. Jika Anda menggunakan FlexMatch sebagai layanan perjodohan mandiri, Anda mungkin ingin membangun fitur-fitur ini ke dalam sistem Anda.

Persetujuan Pemain

Anda dapat mengonfigurasi pembuat kecocokan untuk mewajibkan semua pemain yang dipilih untuk kecocokan harus menerima partisipasi. Jika sistem Anda memerlukan persetujuan, semua pemain harus diberi opsi untuk menyetujui atau menolak kecocokan yang diusulkan. Sebuah kecocokan harus mendapatkan persetujuan dari semua pemain dalam kecocokan yang diusulkan sebelum kecocokan tersebut dapat diselesaikan. Jika ada pemain yang menolak atau gagal menyetujui kecocokan, kecocokan yang diusulkan akan dibuang dan tiket ditangani dengan cara sebagai berikut. Tiket di mana semua pemain dalam tiket yang menyetujui kecocokan dikembalikan ke kolam pencocokan untuk diproses lebih lanjut. Tiket di mana setidaknya satu pemain menolak kecocokan atau gagal merespons dimasukkan ke dalam status gagal dan tidak lagi diproses. Persetujuan pemain membutuhkan batas waktu; semua pemain harus menyetujui kecocokan yang diusulkan dalam batas waktu untuk melanjutkan kecocokan.

Mode Backfill

Gunakan FlexMatch isi ulang untuk menjaga sesi permainan Anda diisi dengan pemain baru yang cocok sepanjang rentang hidup sesi permainan. Saat menangani permintaan isi ulang, FlexMatch gunakan mak comblang yang sama seperti yang digunakan untuk mencocokkan pemain asli. Anda dapat menyesuaikan bagaimana tiket backfill diprioritaskan dengan tiket untuk kecocokan baru, menempatkan tiket isi ulang ke baris depan atau akhir. Artinya, saat pemain baru memasuki kolam pencocokan, mereka cenderung ditempatkan di game yang sudah ada daripada di game yang baru dibentuk.

Isi ulang manual tersedia baik game Anda menggunakan FlexMatch GameLift hosting Amazon terkelola atau dengan solusi hosting lainnya. Backfill manual memberi Anda fleksibilitas untuk memutuskan kapan harus memicu permintaan backfill. Misalnya, Anda mungkin ingin menambahkan pemain baru hanya selama fase tertentu dari game Anda atau hanya ketika kondisi tertentu ada.

Isi ulang otomatis hanya tersedia untuk game yang menggunakan GameLift hosting Amazon terkelola. Dengan fitur ini diaktifkan, jika sesi permainan dimulai dengan slot pemain terbuka, Amazon GameLift mulai secara otomatis menghasilkan permintaan isi ulang untuk itu. Fitur ini memungkinkan Anda untuk mengatur pencocokan sehingga game baru dimulai dengan jumlah pemain minimum dan kemudian diisi dengan cepat saat pemain baru memasuki kolam pencocokan. Anda dapat menonaktifkan backfill otomatis kapan saja selama rentang masa pakai sesi game.

Properti Game

Untuk game yang digunakan FlexMatch dengan hosting GameLift terkelola Amazon, Anda dapat memberikan informasi tambahan untuk diteruskan ke server game setiap kali sesi game baru diminta. Ini bisa menjadi cara yang berguna untuk melewati konfigurasi mode game yang diperlukan untuk memulai sesi game untuk jenis kecocokan yang sedang dibuat. Semua sesi game untuk kecocokan yang dibuat oleh pembuat kecocokan menerima set properti game yang sama. Anda dapat memvariasikan informasi properti game dengan membuat konfigurasi pencocokan yang berbeda.

Slot Pemain yang Dicadangkan

Anda dapat menentukan bahwa slot pemain tertentu di setiap kecocokan akan dicadangkan dan diisi di lain waktu. Hal ini dilakukan dengan mengonfigurasikan properti “hitungan pemain tambahan” dari konfigurasi pencocokan.

Data Peristiwa Khusus

Gunakan properti ini untuk menyertakan sekumpulan informasi khusus di semua peristiwa terkait pencocokan untuk pembuat kecocokan. Fitur ini dapat berguna untuk melacak aktivitas tertentu yang unik untuk game Anda, termasuk melacak performa pembuat kecocokan Anda.

Bangun seperangkat FlexMatch aturan

Setiap FlexMatch mak comblang harus memiliki aturan yang ditetapkan. Set aturan menentukan dua elemen kunci kecocokan: struktur dan ukuran tim game Anda, dan cara mengelompokkan pemain untuk mendapatkan kecocokan terbaik.

Misalnya, set aturan mungkin menggambarkan kecocokan seperti ini: Buat kecocokan dengan dua tim yang masing-masing terdiri dari lima pemain, satu tim adalah bertahan dan tim lainnya adalah penyerang. Sebuah tim dapat memiliki pemain pemula dan berpengalaman, tetapi keterampilan rata-rata kedua tim harus berada dalam jarak 10 poin satu sama lain. Jika tidak ada kecocokan yang dibuat setelah 30 detik, secara bertahap kendurkan persyaratan keterampilan.

Topik di bagian ini menjelaskan cara merancang dan membangun set aturan pencocokan. Saat membuat kumpulan aturan, Anda dapat menggunakan GameLift konsol Amazon atau AWS CLI.

Topik

Desain set FlexMatch aturan

Topik ini mencakup struktur dasar dari aturan yang ditetapkan dan bagaimana membangun aturan yang ditetapkan untuk pertandingan kecil hingga 40 pemain. Aturan perjodohan melakukan dua hal: menyusun struktur dan ukuran tim pertandingan dan memberi tahu mak comblang bagaimana memilih pemain untuk membentuk pertandingan terbaik.

Tapi aturan perjodohan Anda bisa berbuat lebih banyak. Sebagai contoh, Anda dapat:

- Optimalkan algoritma perjodohan untuk game Anda.
- Siapkan persyaratan latensi pemain minimum untuk melindungi kualitas gameplay.
- Secara bertahap rileks persyaratan tim dan aturan pertandingan dari waktu ke waktu sehingga semua pemain aktif dapat menemukan pertandingan yang dapat diterima ketika mereka menginginkannya.
- Tentukan penanganan untuk permintaan perjodohan grup menggunakan agregasi pihak.
- Memproses pertandingan besar dari 40 pemain atau lebih. Untuk informasi selengkapnya tentang membangun korek api besar, lihatRancang set aturan FlexMatch kecocokan besar.

Saat membuat set aturan perjodohan, pertimbangkan tugas opsional dan wajib berikut ini:

- Jelaskan set aturan (wajib)
- Menyesuaikan algoritme pencocokan
- Menyatakan atribut pemain
- Menentukan tim kecocokan
- Mengatur aturan untuk pencocokan pemain
- Memungkinkan persyaratan untuk melonggar dari waktu ke waktu

Anda dapat membuat set aturan menggunakan GameLift konsol Amazon atau `CreateMatchmakingRuleSet` operasi.

Jelaskan set aturan (wajib)

Berikan rincian untuk set aturan.

- nama (opsional) — Label deskriptif untuk Anda gunakan sendiri. Nilai ini tidak terkait dengan nama set aturan yang Anda tentukan saat membuat aturan yang ditetapkan dengan Amazon GameLift.
- ruleLanguageVersion— Versi bahasa ekspresi properti yang digunakan untuk membuat FlexMatch aturan. Nilainya harus`1.0`.

Menyesuaikan algoritme pencocokan

FlexMatch mengoptimalkan algoritma default untuk sebagian besar game untuk membuat pemain masuk ke pertandingan yang dapat diterima dengan waktu tunggu minimal. Anda dapat menyesuaikan algoritme dan menyesuaikan perjodohan untuk gim Anda.

Berikut ini adalah algoritma FlexMatch perjodohan default:

1. FlexMatch menempatkan semua tiket perjodohan terbuka dan tiket isi ulang di kolam tiket.
2. FlexMatch mengelompokkan tiket secara acak di kolam menjadi satu atau lebih batch. Saat kumpulan tiket semakin besar, FlexMatch bentuk batch tambahan untuk mempertahankan ukuran batch yang optimal.
3. FlexMatch mengurutkan tiket berdasarkan usia, dalam setiap batch.
4. FlexMatch membangun kecocokan berdasarkan tiket tertua dari setiap batch.

Untuk menyesuaikan algoritme pencocokan, tambahkan komponen `algorithm` ke skema set aturan Anda. Lihat FlexMatchaturan set skema untuk informasi selengkapnya.

Gunakan penyesuaian opsional berikut untuk memengaruhi berbagai tahapan proses perjodohan Anda.

- Menambahkan penyortiran pra-batch
- Formulir batch berdasarkan atribut batchDistance
- Memprioritaskan tiket backfill

- Mendukung tiket lama dengan ekspansi

Menambahkan penyortiran pra-batch

Anda dapat mengurutkan kumpulan tiket sebelum membentuk batch. Jenis kustomisasi ini paling efektif dengan game dengan kolam tiket besar. Penyortiran pra-batch dapat membantu mempercepat proses perjodohan dan meningkatkan keseragaman pemain dalam karakteristik yang ditentukan.

Tentukan metode penyortiran pra-batch menggunakan properti `batchingPreference` algoritma. Pengaturan default-nya adalah `random`.

Opsi untuk menyesuaikan penyortiran pra-batch meliputi:

- Urutkan berdasarkan atribut pemain. Berikan daftar atribut pemain untuk mengurutkan kumpulan tiket.

  Untuk mengurutkan berdasarkan atribut pemain, atur `batchingPreference` ke`sorted`, dan tentukan daftar atribut pemain And`sortByAttributes`. Untuk menggunakan atribut, pertama-tama deklarasikan atribut dalam `playerAttributes` komponen kumpulan aturan.

  Dalam contoh berikut, FlexMatch urutkan kumpulan tiket berdasarkan peta permainan pilihan pemain dan kemudian berdasarkan keterampilan pemain. Batch yang dihasilkan lebih cenderung berisi pemain dengan keterampilan serupa yang ingin menggunakan peta yang sama.

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

- Urutkan berdasarkan latensi. Buat kecocokan dengan latensi terendah yang tersedia atau buat kecocokan dengan cepat dengan latensi yang dapat diterima. Kustomisasi ini berguna untuk set aturan yang membentuk pertandingan besar lebih dari 40 pemain.

  Atur properti algoritma `strategy` ke`balanced`. Strategi seimbang membatasi jenis pernyataan aturan yang tersedia. Untuk informasi selengkapnya, lihat Rancang set aturan FlexMatch kecocokan besar.

  FlexMatch mengurutkan tiket berdasarkan data latensi yang dilaporkan pemain dengan salah satu cara berikut:

- Lokasi latensi terendah. Kumpulan tiket diurutkan berdasarkan lokasi di mana pemain melaporkan nilai latensi terendah mereka. FlexMatch kemudian mengumpulkan tiket dengan latensi rendah di lokasi yang sama, menciptakan pengalaman bermain game yang lebih baik. Ini juga mengurangi jumlah tiket di setiap batch, sehingga perjodohan bisa memakan waktu lebih lama. Untuk menggunakan kustomisasi ini, atur `batchingPreference` ke`fastestRegion`, seperti yang ditunjukkan pada contoh berikut.

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

- Latensi yang dapat diterima cocok dengan cepat. Kumpulan tiket diurutkan berdasarkan lokasi di mana pemain melaporkan nilai latensi yang dapat diterima. Ini membentuk lebih sedikit batch yang berisi lebih banyak tiket. Dengan lebih banyak tiket di setiap batch, menemukan pertandingan yang dapat diterima lebih cepat. Untuk menggunakan kustomisasi ini, atur properti `batchingPreference` ke `largestPopulation`, seperti yang ditunjukkan pada contoh berikut.

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

> **Note**
>
> Nilai default untuk strategi seimbang adalah`largestPopulation`.

Memprioritaskan tiket backfill

Jika game Anda menerapkan pengisian ulang otomatis atau pengisian ulang manual, Anda dapat menyesuaikan cara FlexMatch memproses tiket perjodohan berdasarkan jenis permintaan. Jenis permintaan dapat berupa permintaan kecocokan atau pengisian ulang baru. Secara default, FlexMatch memperlakukan kedua jenis permintaan dengan cara yang sama.

Prioritas pengisian ulang berdampak pada cara FlexMatch menangani tiket setelah menggabungkannya. Prioritas isi ulang memerlukan set aturan untuk menggunakan strategi pencarian lengkap.

FlexMatch tidak cocok dengan beberapa tiket isi ulang bersama-sama.

Untuk mengubah prioritas tiket isi ulang, atur properti. `backfillPriority`

- Cocokkan tiket isi ulang terlebih dahulu. Opsi ini mencoba mencocokkan tiket isi ulang sebelum membuat pertandingan baru. Ini berarti bahwa pemain yang masuk memiliki peluang lebih tinggi untuk bergabung dengan game yang ada.

  Cara terbaik untuk menggunakan ini jika game Anda menggunakan auto-backfill. Auto-backfill sering digunakan dalam game dengan sesi permainan pendek dan perputaran pemain tinggi. Auto-backfill membantu game-game ini membentuk pertandingan minimum yang layak dan memulainya saat FlexMatch mencari lebih banyak pemain untuk mengisi slot terbuka.

  Atur `backfillPriority` ke `high` .

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

- Match backfill tiket terakhir. Opsi ini mengabaikan tiket isi ulang sampai mengevaluasi semua tiket lainnya. Ini berarti bahwa FlexMatch mengisi kembali pemain yang masuk ke dalam game yang ada ketika tidak dapat mencocokkan mereka ke dalam game baru.

  Opsi ini berguna ketika Anda ingin menggunakan backfill sebagai opsi kesempatan terakhir untuk memasukkan pemain ke dalam permainan, seperti ketika tidak ada cukup pemain untuk membentuk pertandingan baru.

  Atur `backfillPriority` ke `low`.

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

Mendukung tiket lama dengan ekspansi

Aturan ekspansi mengendurkan kriteria kecocokan saat pertandingan sulit diselesaikan. Amazon GameLift menerapkan aturan ekspansi ketika tiket dalam pertandingan yang diselesaikan sebagian

mencapai usia tertentu. Stempel waktu pembuatan tiket menentukan kapan Amazon GameLift menerapkan aturan; secara default, FlexMatch melacak stempel waktu tiket yang paling cocok.

Untuk mengubah saat FlexMatch menerapkan aturan ekspansi, tetapkan properti `expansionAgeSelection` sebagai berikut:

- Perluas berdasarkan tiket terbaru. Opsi ini menerapkan aturan perluasan berdasarkan tiket terbaru yang ditambahkan ke kecocokan potensial. Setiap kali FlexMatch cocok dengan tiket baru, jam waktu diatur ulang. Dengan opsi ini, pertandingan yang dihasilkan cenderung memiliki kualitas yang lebih tinggi tetapi membutuhkan waktu lebih lama untuk dicocokkan; permintaan pertandingan mungkin habis sebelum diselesaikan jika terlalu lama untuk dicocokkan. Setel `expansionAgeSelection` ke`newest`. `newest`adalah default.
- Perluas berdasarkan tiket tertua. Opsi ini menerapkan aturan ekspansi berdasarkan tiket tertua dalam pertandingan potensial. Dengan opsi ini, FlexMatch menerapkan ekspansi lebih cepat, yang meningkatkan waktu tunggu untuk pemain paling awal yang cocok, tetapi menurunkan kualitas pertandingan untuk semua pemain. Atur `expansionAgeSelection` ke `oldest`.

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

Menyatakan atribut pemain

Di bagian ini, daftar atribut pemain individu untuk disertakan dalam permintaan perjodohan. Ada dua alasan Anda mungkin mendeklarasikan atribut pemain dalam set aturan:

- Ketika set aturan berisi aturan yang bergantung pada atribut pemain.
- Bila Anda ingin meneruskan atribut pemain ke sesi permainan melalui permintaan pertandingan. Misalnya, Anda mungkin ingin meneruskan pilihan karakter pemain ke sesi permainan sebelum setiap pemain terhubung.

Saat menyatakan atribut pemain, sertakan informasi berikut:

- name (diperlukan) – Nilai ini harus unik untuk set aturan.
- type (required) — Tipe data dari nilai atribut. Jenis data yang valid adalah angka, string, atau peta string.

- default (opsional) - Masukkan nilai default untuk digunakan jika permintaan perjodohan tidak memberikan nilai atribut. Jika tidak ada default yang dideklarasikan dan permintaan tidak menyertakan nilai, tidak FlexMatch dapat memenuhi permintaan.

Menentukan tim kecocokan

Menjelaskan struktur dan ukuran tim untuk kecocokan. Setiap kecocokan harus memiliki setidaknya satu tim, dan Anda dapat menentukan tim sebanyak yang Anda inginkan. Tim Anda dapat memiliki jumlah pemain yang sama atau asimetris. Misalnya, Anda dapat menentukan tim monster pemain tunggal dan tim pemburu dengan 10 pemain.

FlexMatch proses mencocokkan permintaan sebagai kecocokan kecil atau kecocokan besar, berdasarkan cara kumpulan aturan menentukan ukuran tim. Potensi pertandingan hingga 40 pemain adalah pertandingan kecil, pertandingan dengan lebih dari 40 pemain adalah pertandingan besar. Untuk menentukan ukuran kecocokan potensial kumpulan aturan, tambahkan maxPlayerpengaturan untuk semua tim yang ditentukan dalam kumpulan aturan.

- name (diperlukan) – Tetapkan nama yang unik untuk setiap tim. Anda menggunakan nama ini dalam aturan dan ekspansi, dan FlexMatch referensi untuk data perjodohan dalam sesi permainan.
- maxPlayers(wajib) — Tentukan jumlah maksimum pemain yang akan ditetapkan ke tim.
- minPlayers(wajib) — Tentukan jumlah minimum pemain yang akan ditetapkan ke tim.
- kuantitas (opsional) - Tentukan jumlah tim yang akan dibuat dengan definisi ini. Saat FlexMatch membuat pertandingan, ini memberi tim ini nama yang disediakan dengan nomor yang ditambahkan. Misalnya`Red-Team1`,`Red-Team2`, dan`Red-Team3`.

FlexMatch mencoba untuk mengisi tim dengan ukuran pemain maksimum tetapi membuat tim dengan lebih sedikit pemain. Jika Anda ingin semua tim dalam kecocokan memiliki ukuran yang sama, Anda dapat membuat aturan untuk itu. Lihat FlexMatch contoh set aturan topik untuk contoh `EqualTeamSizes` aturan.

Mengatur aturan untuk pencocokan pemain

Buat seperangkat pernyataan aturan yang mengevaluasi pemain untuk diterima dalam pertandingan. Aturan mungkin menetapkan persyaratan yang berlaku untuk pemain individu, tim, atau seluruh kecocokan. Ketika Amazon GameLift memproses permintaan pertandingan, itu dimulai dengan pemain tertua di kumpulan pemain yang tersedia dan membangun pertandingan di sekitar pemain itu. Untuk bantuan mendetail tentang membuat FlexMatch aturan, lihatFlexMatchjenis aturan.

- name (required) - Nama bermakna yang secara unik mengidentifikasi aturan dalam set aturan. Nama aturan juga dirujuk dalam log peristiwa dan metrik yang melacak aktivitas yang terkait dengan aturan ini.
- description (opsional) – Gunakan elemen ini untuk melampirkan deskripsi teks bentuk bebas.
- type (diperlukan) – Elemen jenis mengidentifikasi operasi yang akan digunakan saat memproses aturan. Setiap jenis aturan memerlukan satu set properti tambahan. Lihat daftar jenis aturan dan properti yang valid di FlexMatchbahasa aturan.
- Properti tipe aturan (mungkin diperlukan) - Bergantung pada jenis aturan yang ditentukan, Anda mungkin perlu menetapkan properti aturan tertentu. Pelajari lebih lanjut tentang properti dan cara menggunakan bahasa ekspresi FlexMatch properti diFlexMatchbahasa aturan.

Memungkinkan persyaratan untuk melonggar dari waktu ke waktu

Ekspansi memungkinkan Anda untuk melonggarkan kriteria aturan dari waktu ke waktu ketika tidak FlexMatch dapat menemukan kecocokan. Fitur ini memastikan bahwa FlexMatch membuat yang terbaik tersedia ketika tidak dapat membuat pasangan yang sempurna. Dengan melonggarkan aturan Anda dengan ekspansi, Anda secara bertahap memperluas kolam pemain yang merupakan kecocokan yang dapat diterima.

Ekspansi dimulai ketika usia tiket terbaru dalam pertandingan yang tidak lengkap cocok dengan waktu tunggu ekspansi. Saat FlexMatch menambahkan tiket baru ke pertandingan, jam waktu tunggu ekspansi mungkin diatur ulang. Anda dapat menyesuaikan bagaimana ekspansi dimulai di `algorithm` bagian set aturan.

Berikut adalah contoh perluasan yang secara bertahap meningkatkan level keterampilan minimum yang diperlukan untuk kecocokan. Set aturan menggunakan pernyataan aturan jarak, dinamai SkillDeltauntuk mengharuskan semua pemain dalam pertandingan berada dalam 5 tingkat keterampilan satu sama lain. Jika tidak ada pertandingan baru yang dibuat selama lima belas detik, ekspansi ini mencari perbedaan tingkat keterampilan 10, dan kemudian sepuluh detik kemudian mencari perbedaan 20.

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

```
            "waitTimeSeconds": 25,
            "value": 20
        }]
    }]
```

Dengan pembuat kecocokan yang mengaktifkan backfill otomatis, jangan terlalu cepat melonggarkan persyaratan jumlah pemain Anda. Diperlukan beberapa detik untuk memulai sesi game baru dan memulai backfill otomatis. Pendekatan yang lebih baik adalah memulai ekspansi Anda setelah pengisian ulang otomatis cenderung dimulai untuk game Anda. Waktu ekspansi bervariasi tergantung pada komposisi tim Anda, jadi lakukan pengujian untuk menemukan strategi ekspansi terbaik untuk game Anda.

Rancang set aturan FlexMatch kecocokan besar

Jika set aturan Anda membuat kecocokan yang memungkinkan 41 hingga 200 pemain, Anda perlu melakukan beberapa penyesuaian pada konfigurasi set aturan Anda. Penyesuaian ini mengoptimalkan algoritme kecocokan sehingga dapat membangun kecocokan besar yang layak sambil juga mempersingkat waktu tunggu pemain. Akibatnya, set aturan pencocokan besar menggantikan aturan khusus yang memakan waktu dengan solusi standar yang dioptimalkan untuk prioritas pencocokan umum.

Berikut cara menentukan apakah Anda perlu mengoptimalkan set aturan untuk kecocokan besar:

1. Untuk setiap tim yang ditentukan dalam set aturan Anda, dapatkan nilai maxPlayer,
2. Tambahkan semua maxPlayernilai. Jika totalnya melebihi 40, Anda memiliki set aturan kecocokan yang besar.

Untuk mengoptimalkan set aturan Anda untuk kecocokan besar, buat penyesuaian yang dijelaskan sebagai berikut. Lihat skema untuk aturan kecocokan besar yang diatur di Skema kumpulan aturan untuk kecocokan besar dan contoh set aturan di Referensi: Buat kecocokan besar.

Sesuaikan algoritme kecocokan untuk kecocokan besar

Tambahkan komponen algoritme ke set aturan, jika belum ada. Mengatur properti berikut.

- `strategy` (diperlukan) – Atur properti `strategy` ke “seimbang”. Pengaturan ini memicu FlexMatch untuk melakukan pemeriksaan pasca-pertandingan tambahan untuk menemukan keseimbangan tim yang optimal berdasarkan atribut pemain tertentu, yang didefinisikan dalam

properti. `balancedAttribute` Strategi seimbang menggantikan kebutuhan akan aturan khusus untuk membangun tim yang seimbang.

- `balancedAttribute` (diperlukan) – Identifikasi atribut pemain untuk digunakan saat menyeimbangkan tim dalam kecocokan. Atribut ini harus memiliki jenis data numerik (ganda atau bilangan bulat). Misalnya, jika Anda memilih untuk menyeimbangkan keterampilan pemain FlexMatch , cobalah untuk menetapkan pemain sehingga semua tim memiliki tingkat keterampilan agregat yang sama rata mungkin. Atribut penyeimbang harus dinyatakan dalam atribut pemain set aturan.
- `batchingPreference` (opsional) – Tentukan seberapa besar penekanan yang ingin Anda berikan untuk membentuk kecocokan dengan latensi serendah mungkin untuk pemain Anda. Pengaturan ini memengaruhi cara tiket kecocokan diurutkan sebelum membuat kecocokan. Opsi termasuk:
  - Populasi terbesar. FlexMatch memungkinkan kecocokan menggunakan semua tiket di kolam yang memiliki nilai latensi yang dapat diterima di setidaknya satu lokasi yang sama. Akibatnya, kolam tiket potensial cenderung berukuran besar, sehingga membuatnya lebih mudah untuk mengisi kecocokan dengan lebih cepat. Pemain dapat ditempatkan dalam game dengan latensi yang dapat diterima, tetapi tidak selalu optimal. Jika `batchingPreference` properti tidak disetel, ini adalah perilaku default saat `strategy` disetel ke “balanced”.
  - Lokasi tercepat. FlexMatch pra-urutkan semua tiket di kolam berdasarkan di mana mereka melaporkan nilai latensi terendah. Akibatnya, pertandingan cenderung dibentuk dengan pemain yang melaporkan latensi rendah di lokasi yang sama. Pada saat yang sama, kolam tiket potensial untuk setiap kecocokan lebih kecil, yang dapat meningkatkan waktu yang dibutuhkan untuk mengisi kecocokan. Selain itu, karena prioritas yang lebih tinggi ditempatkan pada latensi, pemain dalam pertandingan dapat bervariasi lebih luas sehubungan dengan atribut penyeimbangan.

Contoh berikut mengonfigurasi algoritme pencocokan untuk berperilaku sebagai berikut: (1) Pra-urutkan kumpulan tiket untuk mengelompokkan tiket berdasarkan lokasi di mana mereka memiliki nilai latensi yang dapat diterima; (2) Bentuk kumpulan tiket yang diurutkan untuk pencocokan; (3) Buat pertandingan dengan tiket dalam satu batch dan seimbangkan tim untuk meratakan keterampilan pemain rata-rata.

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

```
},
```

Menyatakan atribut pemain

Pastikan Anda menyatakan atribut pemain yang digunakan sebagai atribut penyeimbang dalam algoritme set aturan. Atribut ini harus dimasukkan untuk setiap pemain dalam permintaan pencocokan. Anda dapat memberikan nilai default untuk atribut pemain, tetapi penyeimbangan atribut berfungsi paling baik saat nilai khusus pemain diberikan.

Menentukan tim

Proses mendefinisikan ukuran dan struktur tim sama dengan pertandingan kecil, tetapi cara FlexMatch mengisi tim berbeda. Hal ini memengaruhi tampilan kecocokan jika hanya diisi sebagian. Anda mungkin ingin menyesuaikan ukuran tim minimum Anda sebagai tanggapan.

FlexMatch menggunakan aturan berikut saat menugaskan pemain ke tim. Pertama: cari tim yang belum mencapai persyaratan minimum pemain. Kedua: dari tim-tim tersebut, temukan salah satu tim yang slotnya paling terbuka.

Untuk kecocokan yang menentukan beberapa tim berukuran sama, pemain ditambahkan secara berurutan ke setiap tim hingga penuh. Akibatnya, tim dalam kecocokan selalu memiliki jumlah pemain yang hampir sama, bahkan saat pertandingan tidak penuh. Saat ini, tidak ada cara untuk memaksa tim berukuran sama dalam kecocokan besar. Untuk kecocokan dengan tim berukuran asimetris, prosesnya sedikit lebih rumit. Dalam skenario ini, pemain awalnya ditetapkan ke tim terbesar yang memiliki slot paling terbuka. Karena jumlah slot terbuka menjadi lebih merata di semua tim, pemain ditempatkan ke tim yang lebih kecil.

Misalnya, Anda memiliki aturan yang ditetapkan dengan tiga tim. Tim Merah dan Biru, keduanya diatur ke `maxPlayers`=10, `minPlayers`=5. Tim Hijau diatur ke `maxPlayers`=3, `minPlayers`=2. Berikut adalah urutan pengisiannya:

1. Tidak ada tim yang mencapai `minPlayers`. Tim Merah dan Biru memiliki 10 slot terbuka, sedangkan Hijau memiliki 3. 10 pemain pertama ditugaskan (5 masing-masing) ke tim Merah dan Biru. Kedua tim kini telah mencapai `minPlayers`.
2. Tim hijau belum mencapai `minPlayers`. 2 pemain berikutnya ditetapkan ke tim Hijau. Tim Hijau sekarang telah mencapai `minPlayers`.
3. Dengan semua tim di`minPlayers`, pemain tambahan sekarang ditugaskan berdasarkan jumlah slot terbuka. Tim Merah dan Biru masing-masing memiliki 5 slot terbuka, sedangkan tim Green

memiliki 1. 8 pemain berikutnya ditetapkan (masing-masing 4) ke tim Merah dan Biru. Semua tim sekarang memiliki 1 slot terbuka.

4. 3 slot pemain yang tersisa ditetapkan (masing-masing 1) ke tim tanpa urutan tertentu.

Tetapkan aturan untuk pertandingan besar

Pencocoka untuk kecocokan besar bergantung terutama pada strategi penyeimbangan dan pengoptimalan batching latensi. Sebagian besar aturan kustom tidak tersedia. Namun, Anda dapat memasukkan jenis aturan berikut:

- Aturan yang menetapkan batas keras pada latensi pemain. Gunakan tipe `latency` aturan dengan propertimaxLatency. Lihat Latensi aturan referensi. Berikut adalah contoh yang menyetel latensi pemain maksimum ke 200 milidetik:

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

- Aturan untuk batch pemain berdasarkan kedekatan dalam atribut pemain tertentu. Ini berbeda dengan mendefinisikan atribut balancing sebagai bagian dari algoritma pertandingan besar, yang berfokus pada membangun tim yang cocok secara merata. Aturan ini mengumpulkan tiket perjodohan berdasarkan kesamaan dalam nilai atribut yang ditentukan, seperti keterampilan pemula atau ahli, yang cenderung mengarah ke pemain pertandingan yang selaras erat pada atribut yang ditentukan. Gunakan tipe `batchDistance` aturan, identifikasi atribut berbasis numerik, dan tentukan rentang terluas untuk mengizinkan. Lihat Aturan jarak batch referensi. Berikut adalah contoh yang meminta pemain pertandingan berada dalam satu tingkat keterampilan satu sama lain:

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

Melonggarkan persyaratan kecocokan besar

Seperti halnya kecocokan kecil, Anda dapat menggunakan perluasan untuk melonggarkan persyaratan kecocokan dari waktu ke waktu ketika tidak ada kecocokan valid yang memungkinkan. Dengan kecocokan besar, Anda memiliki opsi untuk melonggarkan aturan latensi atau jumlah pemain tim.

Jika Anda menggunakan pengisian ulang pertandingan otomatis untuk pertandingan besar, hindari merilekskan jumlah pemain tim Anda terlalu cepat. FlexMatch mulai menghasilkan permintaan pengisian ulang hanya setelah sesi permainan dimulai, yang mungkin tidak terjadi selama beberapa detik setelah pertandingan dibuat. Selama waktu itu, FlexMatch dapat membuat beberapa sesi permainan baru yang terisi sebagian, terutama ketika aturan hitungan pemain diturunkan. Akibatnya, Anda mendapatkan dengan lebih banyak sesi game daripada yang Anda butuhkan dan persebaran pemain terlalu tipis. Praktik terbaik adalah memberikan langkah pertama dalam perluasan jumlah pemain Anda waktu tunggu yang lebih lama, cukup lama untuk memulai sesi game Anda. Karena permintaan backfill diberi prioritas lebih tinggi untuk kecocokan besar, pemain yang masuk akan ditempatkan ke dalam game yang ada sebelum game baru dimulai. Anda mungkin perlu bereksperimen untuk menemukan waktu tunggu yang ideal untuk game Anda.

Berikut adalah contoh yang secara bertahap menurunkan jumlah pemain tim Kuning, dengan waktu tunggu awal yang lebih lama. Perlu diingat bahwa waktu tunggu dalam perluasan set aturan adalah mutlak, tidak ditambah. Jadi perluasan pertama terjadi pada lima detik, dan perluasan kedua terjadi lima detik kemudian, yaitu pada sepuluh detik.

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

Tutorial: Buat set aturan perjodohan

Sebelum Anda membuat aturan perjodohan yang ditetapkan untuk mak GameLift FlexMatch comblang Amazon Anda, sebaiknya periksa sintaks set aturan. Setelah membuat aturan yang

ditetapkan menggunakan GameLift konsol Amazon atau AWS Command Line Interface (AWS CLI), Anda tidak dapat mengubahnya.

Perhatikan bahwa ada kuota layanan untuk jumlah maksimum set aturan yang dapat Anda miliki di AWS Wilayah, jadi sebaiknya hapus kumpulan aturan yang tidak digunakan.

Console

Buat set aturan

1. Buka GameLift konsol Amazon di https://console.aws.amazon.com/gamelift/.
2. Beralih ke AWS Wilayah tempat Anda ingin membuat set aturan. Tentukan kumpulan aturan di Wilayah yang sama dengan konfigurasi perjodohan yang menggunakannya.
3. Di panel navigasi, pilih FlexMatch, set aturan Matchmaking.
4. Pada halaman Matchmaking rule sets, pilih Create rule set.
5. Pada halaman set aturan Create matchmaking, lakukan hal berikut:
   a. Di bawah Setelan aturan, untuk Nama, masukkan nama deskriptif unik yang dapat Anda gunakan untuk mengidentifikasinya dalam daftar atau dalam tabel peristiwa dan metrik.
   b. Untuk Rule set, masukkan aturan yang ditetapkanJSON. Untuk informasi tentang mendesain set aturan, lihatDesain set FlexMatch aturan. Anda juga dapat menggunakan salah satu set aturan contoh dariFlexMatch contoh set aturan.
   c. Pilih Validasi untuk memverifikasi bahwa sintaks kumpulan aturan Anda sudah benar. Anda tidak dapat mengedit kumpulan aturan setelah dibuat, jadi sebaiknya validasi terlebih dahulu.
   d. (Opsional) Di bawah Tag, tambahkan tag untuk membantu Anda mengelola dan melacak AWS sumber daya Anda.
6. Pilih Buat. Jika pembuatan berhasil, Anda dapat menggunakan aturan yang ditetapkan dengan mak comblang.

AWS CLI

Buat set aturan

Buka jendela baris perintah dan gunakan perintah create-matchmaking-rule-set.

Perintah contoh ini membuat aturan perjodohan sederhana yang mengatur satu tim. Pastikan untuk membuat aturan yang ditetapkan di AWS Wilayah yang sama dengan konfigurasi perjodohan yang menggunakannya.

```
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}]}'
```

Jika permintaan pembuatan berhasil, Amazon GameLift mengembalikan MatchmakingRuleSetobjek yang menyertakan pengaturan yang Anda tentukan. Seorang mak comblang sekarang dapat menggunakan set aturan baru.

Console

Hapus satu set aturan

1. Buka GameLift konsol Amazon di https://console.aws.amazon.com/gamelift/.
2. Beralih ke Wilayah tempat Anda membuat aturan yang ditetapkan.
3. Di panel navigasi, pilih FlexMatch, set aturan Matchmaking.
4. Pada halaman Matchmaking rule sets, pilih set aturan yang ingin Anda hapus, lalu pilih Delete.
5. Dalam kotak dialog Hapus aturan set, pilih Hapus untuk mengonfirmasi penghapusan.

> **Note**
>
> Jika konfigurasi pencocokan menggunakan set aturan, Amazon akan GameLift menampilkan pesan kesalahan (Tidak dapat menghapus kumpulan aturan). Jika ini terjadi, ubah konfigurasi perjodohan untuk menggunakan kumpulan aturan yang berbeda, lalu coba lagi. Untuk mengetahui konfigurasi perjodohan mana yang menggunakan kumpulan aturan, pilih nama set aturan untuk melihat halaman detailnya.

AWS CLI

Hapus satu set aturan

Buka jendela baris perintah dan gunakan perintah delete-matchmaking-rule-setuntuk menghapus set aturan perjodohan.

Jika konfigurasi perjodohan menggunakan set aturan, Amazon GameLift mengembalikan pesan kesalahan. Jika ini terjadi, ubah konfigurasi perjodohan untuk menggunakan kumpulan aturan yang berbeda, lalu coba lagi. Untuk mendapatkan daftar konfigurasi perjodohan mana yang menggunakan kumpulan aturan, gunakan perintah describe-matchmaking-configurationsdan tentukan nama kumpulan aturan.

Perintah contoh ini memeriksa penggunaan set aturan perjodohan dan kemudian menghapus kumpulan aturan.

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

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

FlexMatch contoh set aturan

FlexMatch set aturan dapat mencakup berbagai skenario perjodohan. Contoh berikut sesuai dengan struktur FlexMatch konfigurasi dan bahasa ekspresi properti. Salin set aturan ini secara keseluruhan atau pilih komponen sesuai kebutuhan.

Untuk informasi selengkapnya tentang penggunaan FlexMatch aturan dan kumpulan aturan, lihat topik berikut:

> **Note**
>
> Saat mengevaluasi tiket perjodohan yang mencakup beberapa pemain, semua pemain dalam permintaan harus memenuhi persyaratan pertandingan.

Topik

- Referensi: Buat dua tim dengan pemain yang serasi
- Referensi: Buat tim yang tidak rata (Hunters vs Monster)
- Referensi: Tetapkan persyaratan tingkat tim dan batas latensi

- Referensi: Gunakan pengurutan eksplisit untuk menemukan kecocokan terbaik
- Referensi: Temukan persimpangan di beberapa atribut pemain
- Referensi: Bandingkan atribut di semua pemain
- Referensi: Buat kecocokan besar
- Referensi: Buat pertandingan besar multi-tim
- Referensi: Buat pertandingan besar dengan pemain dengan atribut serupa
- Referensi: Gunakan aturan gabungan untuk membuat pertandingan dengan pemain dengan atribut serupa atau pilihan serupa
- Referensi: Buat aturan yang menggunakan daftar blokir pemain

Referensi: Buat dua tim dengan pemain yang serasi

Contoh ini menggambarkan cara mengatur dua tim pemain yang sama-sama cocok dengan instruksi berikut.

- Buat dua tim pemain.
  - Sertakan antara empat dan delapan pemain di setiap tim.
  - Tim final harus memiliki jumlah pemain yang sama.
- Sertakan tingkat keterampilan pemain (jika tidak disediakan, default untuk 10).
- Pilih pemain berdasarkan kemiripan tingkat keterampilan mereka dengan pemain lain. Pastikan kedua tim memiliki keterampilan pemain rata-rata dalam jarak 10 poin satu sama lain.
- Jika pertandingan tidak diisi dengan cepat, longgarkan persyaratan keterampilan pemain untuk menyelesaikan pertandingan dalam waktu yang wajar.
  - Setelah 5 detik, perluas pencarian untuk memungkinkan tim dengan keterampilan pemain rata-rata dalam 50 poin.
  - Setelah 15 detik, perluas pencarian untuk memungkinkan tim dengan keterampilan pemain rata-rata dalam 100 poin.

Catatan tentang penggunaan set aturan ini:

- Contoh ini memungkinkan tim memiliki ukuran antara empat dan delapan pemain (walaupun ukurannya harus sama). Untuk tim dengan berbagai ukuran yang valid, pembuat kecocokan melakukan upaya terbaik untuk mencocokkan jumlah maksimum pemain yang diizinkan.

- Aturan `FairTeamSkill` aturan memastikan bahwa tim dicocokkan secara merata berdasarkan keterampilan pemain. Untuk mengevaluasi aturan ini untuk setiap calon pemain baru, FlexMatch sementara menambahkan pemain ke tim dan menghitung rata-rata. Jika aturan gagal, calon pemain tidak ditambahkan ke kecocokan.
- Karena kedua tim memiliki struktur yang identik, Anda dapat memilih untuk membuat hanya satu definisi tim dan mengatur jumlah tim ke "2". Dalam skenario ini, jika Anda menamai tim "alien", maka tim Anda akan diberi nama "alien_1" dan "alien_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
        }]
    }]
}
```

Referensi: Buat tim yang tidak rata (Hunters vs Monster)

Contoh ini menjelaskan mode game di mana sekelompok pemain berburu monster tunggal. Orang-orang memilih peran pemburu atau monster. Pemburu menentukan tingkat keterampilan minimum untuk monster yang ingin mereka hadapi. Ukuran minimum tim pemburu dapat dilonggarkan dari waktu ke waktu untuk menyelesaikan pertandingan. Skenario ini menetapkan petunjuk berikut:

- Buat satu tim yang terdiri dari lima pemburu.
- Buat tim terpisah yang terdiri dari tepat satu monster.
- Sertakan atribut pemain berikut:
  - Tingkat keterampilan pemain (jika tidak disediakan, default ke 10).
  - Tingkat keterampilan monster pilihan pemain (jika tidak disediakan, default ke 10).
  - Apakah pemain ingin menjadi monster (jika tidak disediakan, default ke 0 atau salah).
- Pilih pemain untuk menjadi monster berdasarkan kriteria berikut:
  - Pemain harus meminta peran monster.
  - Pemain harus memenuhi atau melampaui tingkat keterampilan tertinggi yang disukai oleh pemain yang sudah ditambahkan ke tim pemburu.
- Pilih pemain untuk tim pemburu berdasarkan kriteria berikut:
  - Pemain yang meminta peran monster tidak dapat bergabung dengan tim pemburu.
  - Jika peran monster sudah terisi, pemain pasti menginginkan level keterampilan monster yang lebih rendah dari keterampilan monster yang diajukan.

- Jika kecocokan tidak cepat terisi, longgarkan ukuran minimum tim pemburu sebagai berikut:
  - Setelah 30 detik, biarkan game dimulai dengan hanya empat pemain di tim pemburu.
  - Setelah 60 detik, biarkan game dimulai dengan hanya tiga orang dalam tim pemburu.

Catatan tentang penggunaan set aturan ini:

- Dengan menggunakan dua tim terpisah untuk pemburu dan monster, Anda dapat mengevaluasi keanggotaan berdasarkan serangkaian kriteria yang berbeda.

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

Referensi: Tetapkan persyaratan tingkat tim dan batas latensi

Contoh ini menggambarkan cara mengatur tim pemain dan menerapkan seperangkat aturan untuk setiap tim, bukan setiap pemain individu. Contoh ini menggunakan definisi tunggal untuk membuat tiga tim yang sama-sama cocok. Ini juga menetapkan latensi maksimum untuk semua pemain. Latensi maksimum dapat dilonggarkan dari waktu ke waktu untuk menyelesaikan kecocokan. Contoh ini menetapkan instruksi berikut:

- Buat tiga tim pemain.
  - Sertakan antara tiga hingga lima pemain di setiap tim.
  - Tim final harus berisi jumlah pemain yang sama atau hampir sama (dalam satu tim).

- Sertakan atribut pemain berikut:
  - Tingkat keterampilan pemain (jika tidak disediakan, default ke 10).
  - Peran karakter pemain (jika tidak disediakan, defaultnya adalah "petani").
- Pilih pemain berdasarkan kemiripan tingkat keahlian mereka dengan pemain lain dalam kecocokan.
  - Pastikan bahwa setiap tim memiliki keterampilan pemain rata-rata dalam 10 poin satu sama lain.
- Batasi tim dengan jumlah karakter “medis” berikut:
  - Seluruh pertandingan dapat memiliki maksimum lima petugas medis.
- Hanya cocokkan pemain yang melaporkan latensi 50 milidetik atau kurang.
- Jika pertandingan tidak diisi dengan cepat, longgarkan persyaratan latensi pemain sebagai berikut:
  - Setelah 10 detik, izinkan nilai latensi pemain hingga 100 md.
  - Setelah 20 detik, izinkan nilai latensi pemain hingga 150 md.

Catatan tentang penggunaan set aturan ini:

- Set aturan memastikan bahwa tim dicocokkan secara merata berdasarkan keterampilan pemain. Untuk mengevaluasi `FairTeamSkill` aturan, FlexMatch secara tentatif menambahkan calon pemain ke tim dan menghitung keterampilan rata-rata pemain dalam tim. Kemudian FlexMatch membandingkannya dengan keterampilan rata-rata pemain di kedua tim. Jika aturan gagal, calon pemain tidak ditambahkan ke kecocokan.
- Persyaratan level tim dan kecocokan (jumlah total petugas medis) dicapai melalui aturan pengumpulan. Jenis aturan ini mengambil daftar atribut karakter untuk semua pemain dan memeriksanya terhadap jumlah maksimum. Gunakan `flatten` untuk membuat daftar semua pemain di semua tim.
- Saat mengevaluasi berdasarkan latensi, perhatikan hal berikut:
  - Data latensi disediakan dalam permintaan pencocokan sebagai bagian dari objek Pemain. Ini bukan atribut pemain, jadi tidak perlu dicantumkan sebagai satu.
  - Pembuat kecocokan mengevaluasi latensi berdasarkan wilayah. Setiap wilayah dengan latensi lebih tinggi dari maksimum akan diabaikan. Agar dapat diterima untuk kecocokan, pemain harus memiliki setidaknya satu wilayah dengan latensi di bawah maksimum.
  - Jika permintaan pencocokan menghilangkan data latensi satu atau beberapa pemain, permintaan tersebut ditolak untuk semua kecocokan.

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

Referensi: Gunakan pengurutan eksplisit untuk menemukan kecocokan terbaik

Contoh ini mengatur kecocokan sederhana dengan dua tim yang terdiri dari tiga pemain. Ini menggambarkan cara menggunakan aturan penyortiran eksplisit untuk membantu menemukan kecocokan terbaik secepat mungkin. Aturan ini menyortir semua tiket pencocokan aktif untuk membuat kecocokan terbaik berdasarkan persyaratan utama tertentu. Contoh ini diimplementasikan dengan instruksi berikut:

- Buat dua tim pemain.
- Sertakan tepat tiga pemain di setiap tim.
- Sertakan atribut pemain berikut:
  - Level pengalaman (jika tidak disediakan, default ke 50).
  - Mode game pilihan (dapat mencantumkan beberapa nilai) (jika tidak disediakan, default ke "coop" dan "deathmatch").
  - Peta game pilihan, termasuk nama peta dan bobot preferensi (jika tidak disediakan, default ke `"defaultMap"` dengan bobot 100).
- Menyiapkan penyortiran:

  - Urutkan pemain berdasarkan preferensi mereka untuk peta game yang sama dengan pemain jangkar. Pemain dapat memiliki beberapa peta game favorit, jadi contoh ini menggunakan nilai preferensi.
  - Urutkan pemain berdasarkan seberapa dekat level pengalaman mereka dengan pemain jangkar. Dengan penyortiran ini, semua pemain di semua tim akan memiliki level pengalaman yang sedekat mungkin.
- Semua pemain di semua tim harus memilih setidaknya satu mode game yang sama.
- Semua pemain di semua tim harus memilih setidaknya satu peta game yang sama.

Catatan tentang penggunaan set aturan ini:

- Penyortiran peta permainan menggunakan pengurutan absolut yang membandingkan nilai mapPreference atribut. Karena penyortiran tersebut merupakan yang pertama dalam kumpulan aturan, penyortiran ini dilakukan terlebih dahulu.
- Penyortira pengalaman menggunakan penyortiran jarak untuk membandingkan level keterampilan calon pemain dengan keterampilan pemain jangkar.
- Penyortiran dilakukan sesuai urutan yang tercantum dalam set aturan. Dalam skenario ini, pemain disortir berdasarkan preferensi peta game, dan kemudian berdasarkan level pengalaman.

```
{
    "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
    }]
}
```

Referensi: Temukan persimpangan di beberapa atribut pemain

Contoh ini menggambarkan cara menggunakan aturan kumpulan untuk menemukan persimpangan dalam dua atribut pemain atau lebih. Saat bekerja dengan koleksi, Anda dapat menggunakan operasi `intersection` untuk atribut tunggal, dan operasi `reference_intersection_count` untuk beberapa atribut.

Untuk menggambarkan pendekatan ini, contoh ini mengevaluasi pemain dalam kecocokan berdasarkan preferensi karakter mereka. Contoh permainan adalah gaya "free-for-all" di mana semua pemain dalam pertandingan adalah lawan. Setiap pemain diminta untuk (1) memilih karakter untuk diri mereka sendiri, dan (2) memilih karakter yang ingin mereka lawan. Kita membutuhkan aturan yang memastikan bahwa setiap pemain dalam kecocokan menggunakan karakter yang ada di daftar lawan pilihan semua pemain lain.

Set aturan contoh menjelaskan kecocokan dengan karakteristik berikut:

- Struktur tim: Satu tim terdiri dari lima pemain
- Atribut pemain:
  - myCharacter: Karakter yang dipilih pemain.
  - preferredOpponents: Daftar karakter yang ingin dimainkan pemain.
- Aturan kecocokan: Kecocokan potensial dapat diterima jika setiap karakter yang digunakan ada di daftar lawan pilihan setiap pemain.

Untuk menerapkan aturan kecocokan, contoh ini menggunakan aturan kumpulan dengan nilai properti berikut:

- Operasi – Menggunakan operasi `reference_intersection_count` operasi untuk mengevaluasi bagaimana setiap daftar string dalam nilai pengukuran bersimpangan dengan daftar string dalam nilai referensi.
- Pengukuran - Menggunakan ekspresi `flatten` properti untuk membuat daftar daftar string, dengan setiap daftar string berisi nilai myCharacteratribut satu pemain.
- Nilai referensi - Menggunakan ekspresi `set_intersection` properti untuk membuat daftar string dari semua nilai preferredOpponentsatribut yang umum untuk setiap pemain dalam pertandingan.
- Pembatasan – `minCount` diatur ke 1 untuk memastikan bahwa karakter yang dipilih setiap pemain (daftar string dalam pengukuran) cocok dengan setidaknya salah satu lawan pilihan yang umum untuk semua pemain. (string dalam nilai referensi).
- Perluasan — Jika pertandingan tidak diisi dalam waktu 15 detik, longgarkan persyaratan persimpangan minimum.

Alur proses untuk aturan ini adalah sebagai berikut:

1. Seorang pemain ditambahkan ke kecocokan prospektif. Nilai referensi (daftar string) dihitung ulang untuk menyertakan perpotongan dengan daftar lawan pilihan pemain baru. Nilai pengukuran (daftar daftar string) dihitung ulang untuk menambahkan karakter yang dipilih pemain baru sebagai daftar string baru.
2. Amazon GameLift memverifikasi bahwa setiap daftar string dalam nilai pengukuran (karakter yang dipilih pemain) berpotongan dengan setidaknya satu string dalam nilai referensi (lawan pilihan pemain). Karena dalam contoh ini setiap daftar string dalam pengukuran hanya berisi satu nilai, simpangannya adalah 0 atau 1.
3. Jika ada daftar string dalam pengukuran yang tidak besimpangan dengan daftar string nilai referensi, aturan gagal dan pemain baru dikeluarkan dari kecocokan prospektif.
4. Jika kecocokan tidak terisi dalam waktu 15 detik, batalkan persyaratan kecocokan lawan untuk mengisi slot pemain yang tersisa dalam kecocokan.

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

Referensi: Bandingkan atribut di semua pemain

Contoh ini menggambarkan cara membandingkan atribut pemain di seluruh grup pemain.

Set aturan contoh menjelaskan kecocokan dengan karakteristik berikut:

- Struktur tim: Dua tim pemain tunggal
- Atribut pemain:
  - gameMode: Jenis permainan yang dipilih oleh pemain (jika tidak disediakan, default ke “berbasis giliran”).
  - gameMap: Dunia game yang dipilih oleh pemain (jika tidak disediakan, default ke 1).

  - character: Karakter yang dipilih oleh pemain (tidak ada nilai default berarti pemain harus menentukan karakter).
- Aturan kecocokan: Pemain yang cocok harus memenuhi persyaratan berikut:
  - Pemain harus memilih mode game yang sama.
  - Pemain harus memilih peta game yang sama.
  - Pemain banyak memilih karakter yang berbeda.

Catatan tentang penggunaan set aturan ini:

- Untuk menerapkan aturan kecocokan, contoh ini menggunakan aturan perbandingan untuk memeriksa nilai atribut semua pemain. Untuk mode game dan peta, aturan memverifikasi bahwa nilainya sama. Untuk karakter, aturan memverifikasi bahwa nilai-nilai yang berbeda.
- Contoh ini menggunakan definisi satu pemain dengan properti kuantitas untuk membuat kedua tim pemain. Tim diberi nama berikut: "player_1" dan "player_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])"]
    }]
}
```

Referensi: Buat kecocokan besar

Contoh ini menggambarkan cara menyiapkan set aturan untuk pertandingan yang dapat melebihi 40 pemain. Ketika aturan yang ditetapkan menggambarkan tim dengan maxPlayer jumlah total lebih besar dari 40, itu diproses sebagai pertandingan besar. Pelajari selengkapnya di Rancang set aturan FlexMatch kecocokan besar.

Set aturan contoh membuat kecocokan menggunakan petunjuk berikut:

- Buat satu tim dengan maksimal 200 pemain, dengan persyaratan minimal 175 pemain.
- Kriteria penyeimbangan: Pilih pemain berdasarkan level keterampilan yang sama. Semua pemain harus melaporkan level keterampilan mereka untuk dicocokkan.
- Preferensi batching: Mengelompokkan pemain dengan kriteria penyeimbangan yang sama saat membuat kecocokan.
- Aturan latensi: Setel latensi pemain maksimum yang dapat diterima 150 milidetik.
- Jika kecocokan tidak diisi dengan cepat, longgarkan persyaratan untuk menyelesaikan kecocokan dalam waktu yang wajar.
  - Setelah 10 detik, terima tim dengan 150 pemain.

- Setelah 12 detik, naikkan latensi maksimum yang dapat diterima menjadi 200 milidetik.
- Setelah 15 detik, terima tim dengan 100 pemain.

Catatan tentang penggunaan set aturan ini:

- Karena algoritme menggunakan preferensi batching "populasi terbesar", pemain pertama-tama diurutkan berdasarkan kriteria penyeimbangan. Akibatnya, kecocokan cenderung lebih penuh dan berisi pemain yang lebih mirip dalam keterampilan. Semua pemain memenuhi persyaratan latensi yang dapat diterima, tetapi mereka mungkin tidak mendapatkan latensi terbaik untuk lokasi mereka.
- Strategi algoritme yang digunakan dalam set aturan ini, "populasi terbesar", adalah pengaturan default. Untuk menggunakan default, Anda dapat memilih untuk menghilangkan pengaturan.
- Jika Anda telah mengaktifkan backfill kecocokan, jangan terlalu cepat melonggarkan persyaratan jumlah pemain, atau Anda mungkin mendapatkan terlalu banyak sesi game yang terisi sebagian. Pelajari selengkapnya di Melonggarkan persyaratan kecocokan besar.

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

Referensi: Buat pertandingan besar multi-tim

Contoh ini menggambarkan cara menyiapkan kumpulan aturan untuk kecocokan dengan beberapa tim yang dapat melebihi 40 pemain. Contoh ini menggambarkan cara membuat beberapa tim identik dengan satu definisi dan bagaimana tim berukuran asimetris diisi selama pembuatan kecocokan.

Set aturan contoh membuat kecocokan menggunakan petunjuk berikut:

- Buat sepuluh tim "pemburu" identik dengan maksimal 15 pemain, dan satu tim "monster" dengan tepat 5 pemain.
- Kriteria penyeimbangan: Pilih pemain berdasarkan jumlah membunuh monster. Jika pemain tidak melaporkan jumlah membunuh mereka, gunakan nilai default 5.
- Preferensi batching: Kelompok pemain berdasarkan wilayah di mana mereka melaporkan latensi pemain tercepat.
- Aturan latensi: Menetapkan latensi pemain maksimum yang dapat diterima sebesar 200 milidetik.
- Jika kecocokan tidak diisi dengan cepat, longgarkan persyaratan untuk menyelesaikan kecocokan dalam waktu yang wajar.
  - Setelah 15 detik, terima tim dengan 10 pemain.
  - Setelah 20 detik, terima tim dengan 8 pemain.

Catatan tentang penggunaan set aturan ini:

- Set aturan ini mendefinisikan tim yang berpotensi dapat menampung hingga 155 pemain, yang menjadikannya kecocokan besar. (10 x 15 pemburu + 5 monster = 155)
- Karena algoritme menggunakan preferensi batching "wilayah tercepat", pemain cenderung ditempatkan di wilayah yang melaporkan latensi lebih cepat dan bukan di wilayah yang melaporkan latensi tinggi (tetapi dapat diterima). Pada saat yang sama, kecocokan cenderung memiliki lebih sedikit pemain, dan kriteria penyeimbangan (jumlah keterampilan monster) mungkin lebih bervariasi.
- Saat perluasan didefinisikan untuk definisi multi-tim (kuantitas > 1), perluasan berlaku untuk semua tim yang dibuat dengan definisi tersebut. Jadi dengan melonggarkan pengaturan pemain minimum tim pemburu, kesepuluh tim pemburu dapat terpengaruh secara merata.
- Karena aturan ini dioptimalkan untuk meminimalkan latensi pemain, aturan latensi bertindak sebagai catch-all untuk mengecualikan pemain yang tidak memiliki opsi koneksi yang dapat diterima. Kita tidak perlu melonggarkan persyaratan ini.
- Berikut cara FlexMatch mengisi kecocokan untuk aturan ini yang ditetapkan sebelum ekspansi apa pun berlaku:
  - Belum ada tim yang mencapai minPlayers hitungan. Tim Pemburu memiliki 15 slot terbuka, sedangkan tim Monster memiliki 5 slot terbuka.
    - 100 pemain pertama ditugaskan (10 masing-masing) ke sepuluh tim pemburu.
    - 22 pemain berikutnya ditugaskan secara berurutan (masing-masing 2) ke tim pemburu dan tim monster.
  - Tim Hunter telah mencapai minPlayers jumlah masing-masing 12 pemain. Tim monster memiliki 2 pemain dan belum mencapai minPlayers hitungan.
    - Tiga pemain berikutnya ditugaskan ke tim monster.
  - Semua tim telah mencapai minPlayers hitungan. Tim pemburu masing-masing memiliki tiga slot terbuka. Tim Monster sudah penuh.
    - 30 pemain terakhir ditugaskan secara berurutan ke tim pemburu, sehingga memastikan bahwa semua tim pemburu memiliki ukuran yang hampir sama (plus atau minus satu pemain).
- Jika Anda telah mengaktifkan pengisian ulang untuk pertandingan yang dibuat dengan kumpulan aturan ini, jangan terlalu cepat melonggarkan persyaratan jumlah pemain, atau Anda mungkin berakhir dengan terlalu banyak sesi game yang terisi sebagian. Pelajari selengkapnya di Melonggarkan persyaratan kecocokan besar.

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

Referensi: Buat pertandingan besar dengan pemain dengan atribut serupa

Contoh ini menggambarkan cara menyiapkan set aturan untuk kecocokan dengan dua tim menggunakan `batchDistance`. Contohnya:

- Aturan `SimilarLeague` memastikan semua pemain dalam kecocokan memiliki `league` dalam 2 pemain lainnya.
- Aturan `SimilarSkill` memastikan semua pemain dalam kecocokan memiliki `skill` dalam 10 pemain lainnya. Jika seorang pemain telah menunggu 10 detik, jaraknya diperluas menjadi 20. Jika seorang pemain telah menunggu 20 detik, jaraknya diperluas menjadi 40.
- Aturan `SameMap` memastikan semua pemain dalam kecocokan telah meminta hal yang sama `map`.
- Aturan`SameMode` memastikan semua pemain dalam kecocokan telah meminta hal yang sama `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
        }]
    }]
}
```

Referensi: Gunakan aturan gabungan untuk membuat pertandingan dengan pemain dengan atribut serupa atau pilihan serupa

Contoh ini menggambarkan cara menyiapkan set aturan untuk kecocokan dengan dua tim menggunakan `compound`. Contohnya:

- Aturan `SimilarLeagueDistance` memastikan semua pemain dalam kecocokan memiliki `league` dalam 2 pemain lainnya.
- Aturan `SimilarSkillDistance` memastikan semua pemain dalam kecocokan memiliki `skill` dalam 10 pemain lainnya. Jika seorang pemain telah menunggu 10 detik, jaraknya diperluas menjadi 20. Jika seorang pemain telah menunggu 20 detik, jaraknya diperluas menjadi 40.

- Aturan `SameMapComparison` memastikan semua pemain dalam kecocokan telah meminta hal yang sama `map`.
- Aturan`SameModeComparison` memastikan semua pemain dalam kecocokan telah meminta hal yang sama `mode`.
- `CompoundRuleMatchmaker`Aturan memastikan kecocokan jika setidaknya salah satu dari kondisi berikut benar:
  - Pemain dalam pertandingan telah meminta hal yang sama `map` dan sama`mode`.
  - Pemain dalam pertandingan memiliki `league` atribut `skill` dan atribut yang sebanding.

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

Referensi: Buat aturan yang menggunakan daftar blokir pemain

Contoh ini menggambarkan aturan yang memungkinkan pemain menghindari kecocokan dengan pemain tertentu lainnya. Pemain dapat membuat daftar blok, yang dievaluasi oleh mak comblang

selama pemilihan pemain untuk pertandingan. Untuk panduan lebih lanjut tentang menambahkan daftar blokir atau menghindari fitur daftar, lihat AWS untuk Blog Game.

Contoh ini menetapkan instruksi berikut:

- Buat dua tim yang terdiri dari lima pemain.
- Masukkan daftar blok pemain, yang merupakan daftar pemain IDs (hingga 100).
- Bandingkan semua pemain dengan daftar blok masing-masing pemain dan tolak pertandingan yang diusulkan jika ada pemain IDs yang diblokir ditemukan.

Catatan tentang penggunaan set aturan ini:

- Saat mengevaluasi pemain baru untuk ditambahkan ke pertandingan yang diusulkan (atau untuk mengisi kembali tempat dalam pertandingan yang ada), pemain mungkin ditolak karena salah satu alasan berikut:
  - Jika pemain baru ada di daftar blok untuk setiap pemain yang sudah dipilih untuk pertandingan.
  - Jika ada pemain yang sudah dipilih untuk pertandingan ada di daftar blok pemain baru.
- Seperti yang ditunjukkan, set aturan ini mencegah pencocokan pemain dengan pemain mana pun di daftar bloknya. Anda dapat mengubah persyaratan ini ke preferensi (juga disebut daftar “hindari”) dengan menambahkan perluasan aturan dan meningkatkan `maxCount` nilainya.

```
{
    "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
    }]
}
```

Membuat konfigurasi pencocokan

Untuk menyiapkan GameLift FlexMatch mak comblang Amazon untuk memproses permintaan perjodohan, buat konfigurasi perjodohan. Gunakan GameLift konsol Amazon atau AWS Command Line Interface (AWS CLI). Untuk informasi selengkapnya tentang membuat mak comblang, lihat. Desain FlexMatch mak comblang

Topik

- Tutorial: Buat mak comblang untuk hosting Amazon GameLift
- Tutorial: Buat mak comblang untuk mandiri FlexMatch
- Tutorial: Mengedit konfigurasi perjodohan

Tutorial: Buat mak comblang untuk hosting Amazon GameLift

Sebelum membuat konfigurasi perjodohan, buat kumpulan aturan dan antrian sesi GameLift game Amazon untuk digunakan dengan mak comblang.

Console

1. Di GameLift konsol Amazon, di panel navigasi, pilih konfigurasi Matchmaking.
2. Beralih ke AWS Wilayah tempat Anda ingin membuat mak comblang.
3. Pada halaman Matchmaking configurations, pilih Create matchmaking configuration.
4. Pada halaman Tentukan detail konfigurasi, di bawah detail konfigurasi Matchmaking, lakukan hal berikut:

    a. Untuk Nama, masukkan nama mak comblang yang dapat membantu Anda mengidentifikasinya dalam daftar dan metrik. Nama mak comblang harus unik di dalam

Wilayah. Permintaan perjodohan mengidentifikasi mak comblang mana yang akan digunakan dengan nama dan Wilayahnya.

b. (Opsional) Untuk Deskripsi, tambahkan deskripsi untuk membantu mengidentifikasi mak comblang.

c. Untuk set Aturan, pilih aturan yang ditetapkan dari daftar yang akan digunakan dengan mak comblang. Daftar ini berisi semua kumpulan aturan yang telah Anda buat di Wilayah saat ini.

d. Untuk FlexMatch mode, pilih Dikelola untuk hosting GameLift terkelola Amazon. Mode ini meminta FlexMatch untuk meneruskan pertandingan yang berhasil ke antrian sesi permainan yang ditentukan.

e. Untuk AWS Wilayah, pilih Wilayah tempat Anda mengonfigurasi antrean sesi permainan yang ingin Anda gunakan dengan mak comblang.

f. Untuk Antrian, pilih antrean sesi permainan yang ingin Anda gunakan dengan mak comblang.

5. Pilih Berikutnya.

6. Pada halaman Konfigurasi pengaturan, di bawah pengaturan Matchmaking, lakukan hal berikut:

   a. Untuk batas waktu Permintaan, tetapkan jumlah waktu maksimum, dalam hitungan detik, agar mak comblang menyelesaikan kecocokan untuk setiap permintaan. FlexMatch membatalkan permintaan perjodohan yang melebihi waktu ini.

   b. Untuk mode Backfill, pilih mode untuk menangani backfill pertandingan.

      - Untuk mengaktifkan fitur pengisian ulang otomatis, pilih Otomatis.
      - Untuk membuat manajemen permintaan isi ulang Anda sendiri atau tidak menggunakan fitur isi ulang, pilih Manual.

   c. (Opsional) Untuk jumlah pemain tambahan, atur jumlah slot pemain agar tetap terbuka dalam pertandingan. FlexMatch dapat mengisi slot ini dengan pemain di masa depan.

   d. (Opsional) Di bawah opsi penerimaan pertandingan, untuk Penerimaan diperlukan, jika Anda ingin meminta setiap pemain dalam pertandingan yang diusulkan untuk secara aktif menerima partisipasi dalam pertandingan, pilih Diperlukan. Jika Anda memilih opsi ini, maka untuk batas waktu Penerimaan, atur berapa lama, dalam detik, Anda ingin mak comblang menunggu penerimaan pemain sebelum membatalkan pertandingan.

7. (Opsional) Di bawah pengaturan pemberitahuan Acara, lakukan hal berikut:

a. (Opsional) Untuk SNStopik, pilih topik Amazon Simple Notification Service (AmazonSNS) untuk menerima pemberitahuan acara perjodohan. Jika Anda belum menyiapkan SNS topik, Anda dapat memilih ini nanti dengan mengedit konfigurasi perjodohan. Untuk informasi selengkapnya, lihat Siapkan pemberitahuan FlexMatch acara.

b. (Opsional) Untuk data peristiwa khusus, masukkan data kustom apa pun yang ingin Anda kaitkan dengan mak comblang ini dalam pesan acara. FlexMatch termasuk data ini di setiap acara yang terkait dengan mak comblang.

8. (Opsional) Perluas Data game tambahan, lalu lakukan hal berikut:

a. (Opsional) Untuk data sesi Game, masukkan informasi terkait game tambahan yang ingin Anda kirimkan FlexMatch ke sesi permainan baru yang dimulai dengan pertandingan yang dibuat menggunakan konfigurasi perjodohan ini.

b. (Opsional) Untuk properti Game, tambahkan properti pasangan kunci-nilai yang berisi informasi tentang sesi permainan baru.

9. (Opsional) Di bawah Tag, tambahkan tag untuk membantu Anda mengelola dan melacak AWS sumber daya Anda.

10. Pilih Berikutnya.

11. Pada halaman Tinjau dan buat, tinjau pilihan Anda, lalu pilih Buat. Setelah penciptaan berhasil, mak comblang siap menerima permintaan perjodohan.

AWS CLI

Untuk membuat konfigurasi perjodohan dengan AWS CLI, buka jendela baris perintah dan gunakan create-matchmaking-configurationperintah untuk menentukan mak comblang baru.

Perintah contoh ini membuat konfigurasi perjodohan baru yang memerlukan penerimaan pemain dan memungkinkan pengisian ulang otomatis. Ini juga mencadangkan dua slot pemain FlexMatch untuk menambahkan pemain nanti, dan menyediakan beberapa data sesi permainan.

```
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"
```

Jika permintaan pembuatan konfigurasi matchmaking berhasil, Amazon GameLift mengembalikan MatchmakingConfigurationobjek dengan pengaturan yang Anda minta untuk mak comblang. Mak comblang baru siap menerima permintaan perjodohan.

Tutorial: Buat mak comblang untuk mandiri FlexMatch

Sebelum membuat konfigurasi perjodohan, buat aturan yang akan digunakan dengan mak comblang.

Console

1. Buka GameLift konsol Amazon di https://console.aws.amazon.com/gamelift/rumah.
2. Beralih ke AWS Wilayah tempat Anda ingin membuat mak comblang. Untuk daftar Wilayah yang mendukung konfigurasi FlexMatch perjodohan, lihat. Pilih lokasi untuk mak comblang
3. Di panel navigasi, pilih FlexMatch, konfigurasi Matchmaking.
4. Pada halaman Matchmaking configurations, pilih Create matchmaking configuration.
5. Pada halaman Tentukan detail konfigurasi, di bawah detail konfigurasi Matchmaking, lakukan hal berikut:

    a. Untuk Nama, masukkan nama mak comblang yang dapat membantu Anda mengidentifikasinya dalam daftar dan metrik. Nama mak comblang harus unik di dalam Wilayah. Permintaan perjodohan mengidentifikasi mak comblang mana yang akan digunakan dengan nama dan Wilayahnya.

    b. (Opsional) Untuk Deskripsi, tambahkan deskripsi untuk membantu mengidentifikasi mak comblang.

    c. Untuk set Aturan, pilih aturan yang ditetapkan dari daftar yang akan digunakan dengan mak comblang. Daftar ini berisi semua kumpulan aturan yang telah Anda buat di Wilayah saat ini.

d. Untuk FlexMatch mode, pilih Standalone. Ini menunjukkan bahwa Anda memiliki mekanisme khusus untuk memulai sesi permainan baru pada solusi hosting di luar Amazon GameLift.

6. Pilih Berikutnya.
7. Pada halaman Konfigurasi pengaturan, di bawah pengaturan Matchmaking, lakukan hal berikut:

   a. Untuk batas waktu Permintaan, tetapkan jumlah waktu maksimum, dalam hitungan detik, agar mak comblang menyelesaikan kecocokan untuk setiap permintaan. Permintaan perjodohan yang melebihi waktu ini ditolak.

   b. (Opsional) Di bawah opsi penerimaan pertandingan, untuk Penerimaan diperlukan, jika Anda ingin meminta setiap pemain dalam pertandingan yang diusulkan untuk secara aktif menerima partisipasi dalam pertandingan, pilih Diperlukan. Jika Anda memilih opsi ini, maka untuk batas waktu Penerimaan, atur berapa lama, dalam detik, Anda ingin mak comblang menunggu penerimaan pemain sebelum membatalkan pertandingan.

8. (Opsional) Di bawah pengaturan pemberitahuan Acara, lakukan hal berikut:

   a. (Opsional) Untuk SNStopik, pilih SNS topik Amazon untuk menerima pemberitahuan acara perjodohan. Jika Anda belum menyiapkan SNS topik, Anda dapat memilih ini nanti dengan mengedit konfigurasi perjodohan. Untuk informasi selengkapnya, lihat Siapkan pemberitahuan FlexMatch acara.

   b. (Opsional) Untuk data peristiwa khusus, masukkan data kustom apa pun yang ingin Anda kaitkan dengan mak comblang ini dalam pesan acara. FlexMatch termasuk data ini di setiap acara yang terkait dengan mak comblang.

9. (Opsional) Di bawah Tag, tambahkan tag untuk membantu Anda mengelola dan melacak AWS sumber daya Anda.
10. Pilih Berikutnya.
11. Pada halaman Tinjau dan buat, tinjau pilihan Anda, lalu pilih Buat. Setelah penciptaan berhasil, mak comblang siap menerima permintaan perjodohan.

AWS CLI

Untuk membuat konfigurasi perjodohan dengan AWS CLI, buka jendela baris perintah dan gunakan create-matchmaking-configurationperintah untuk menentukan mak comblang baru.

Perintah contoh ini membuat konfigurasi perjodohan baru untuk mak comblang mandiri yang membutuhkan penerimaan pemain.

```
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"
```

Jika permintaan pembuatan konfigurasi matchmaking berhasil, Amazon GameLift mengembalikan MatchmakingConfigurationobjek dengan pengaturan yang Anda minta untuk mak comblang. Mak comblang baru siap menerima permintaan perjodohan.

Tutorial: Mengedit konfigurasi perjodohan

Untuk mengedit konfigurasi perjodohan, pilih konfigurasi Matchmaking dari bilah navigasi dan pilih konfigurasi yang ingin Anda edit. Anda dapat memperbarui bidang apa pun dalam konfigurasi yang ada kecuali namanya.

Saat memperbarui set aturan konfigurasi, kumpulan aturan baru dapat menjadi tidak kompatibel jika ada tiket perjodohan aktif yang ada karena alasan berikut:

- Nama tim baru atau berbeda atau jumlah tim
- Atribut pemain baru
- Perubahan pada tipe atribut pemain yang ada

Untuk membuat salah satu perubahan ini pada set aturan Anda, buat konfigurasi perjodohan baru dengan set aturan yang diperbarui.

Siapkan pemberitahuan FlexMatch acara

Anda dapat menggunakan pemberitahuan acara untuk melacak status permintaan perjodohan individu. Semua game dalam produksi, atau dalam pra-produksi dengan aktivitas perjodohan volume tinggi harus menggunakan pemberitahuan acara.

Terdapat dua pilihan untuk mengatur notifikasi kejadian.

- Mintalah mak comblang Anda mempublikasikan pemberitahuan acara ke topik Amazon Simple Notification Service SNS (Amazon).
- Gunakan EventBridge acara Amazon yang dipublikasikan secara otomatis dan rangkaian alatnya untuk mengelola acara.

Untuk daftar FlexMatch peristiwa yang GameLift dipancarkan Amazon, lihatFlexMatch acara perjodohan.

Topik

- Tutorial: Mengatur EventBridge acara
- Tutorial: Mengatur SNS topik Amazon
- Tutorial: Siapkan SNS topik dengan enkripsi sisi server
- Tutorial: Konfigurasikan langganan topik untuk menjalankan fungsi Lambda

Tutorial: Mengatur EventBridge acara

Amazon GameLift secara otomatis memposting semua acara perjodohan ke Amazon. EventBridge Dengan EventBridge, Anda dapat mengatur aturan agar acara perjodohan diarahkan ke target untuk diproses. Misalnya, Anda dapat menetapkan aturan untuk merutekan acara "PotentialMatchCreated" ke AWS Lambda fungsi yang menangani penerimaan pemain. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge?

> **Note**
>
> Saat Anda mengonfigurasi mak comblang, biarkan bidang target notifikasi tetap kosong atau referensi SNS topik jika Anda ingin menggunakan keduanya dan EventBridge Amazon. SNS

Tutorial: Mengatur SNS topik Amazon

Anda dapat meminta Amazon GameLift mempublikasikan semua acara yang dihasilkan mak FlexMatch comblang ke topik AmazonSNS.

Untuk membuat SNS topik untuk pemberitahuan GameLift acara Amazon

1. Buka SNSkonsol Amazon.
2. Di panel navigasi, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih Create topic (Buat topik).
4. Buat topik di konsol. Untuk informasi selengkapnya, lihat Untuk membuat topik menggunakan Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon. AWS Management Console
5. Pada halaman Detail untuk topik Anda, pilih Edit.
6. (Opsional) Pada halaman Edit untuk topik Anda, perluas kebijakan Access, lalu tambahkan sintaks tebal dari pernyataan kebijakan AWS Identity and Access Management (IAM) berikut ke akhir kebijakan yang ada. (Seluruh kebijakan ditampilkan di sini untuk kejelasan.) Pastikan untuk menggunakan detail Amazon Resource Name (ARN) untuk SNS topik Anda sendiri dan konfigurasi GameLift perjodohan 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. Pilih Simpan perubahan.

Tutorial: Siapkan SNS topik dengan enkripsi sisi server

Anda dapat menggunakan enkripsi sisi server (SSE) untuk menyimpan data sensitif dalam topik terenkripsi. SSEmelindungi konten pesan dalam SNS topik Amazon menggunakan kunci yang dikelola di AWS Key Management Service (AWS KMS). Untuk informasi selengkapnya tentang enkripsi sisi server dengan AmazonSNS, lihat Enkripsi saat istirahat di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.

Untuk menyiapkan SNS topik dengan enkripsi sisi server, tinjau topik berikut:

- Membuat kunci dalam Panduan AWS Key Management Service Pengembang
- Mengaktifkan SSE topik di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon

Saat membuat KMS kunci Anda, gunakan kebijakan KMS kunci berikut:

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

Tutorial: Konfigurasikan langganan topik untuk menjalankan fungsi Lambda

Anda dapat menjalankan fungsi Lambda menggunakan pemberitahuan acara yang dipublikasikan ke topik Amazon SNS Anda. Saat mengonfigurasi mak comblang, pastikan untuk menetapkan target notifikasi ke topik AndaSNS. ARN

AWS CloudFormation Template berikut mengonfigurasi langganan ke SNS topik bernama `MyFlexMatchEventTopic` untuk memanggil fungsi Lambda bernama. `FlexMatchEventHandlerLambdaFunction` Template membuat kebijakan IAM izin yang memungkinkan Amazon menulis GameLift ke SNS topik. Template kemudian menambahkan izin untuk SNS topik untuk menjalankan fungsi 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
```

Mempersiapkan game untuk FlexMatch

Gunakan Amazon GameLift FlexMatch untuk menambahkan fungsionalitas perjodohan pemain ke game Anda. FlexMatchtersedia dengan GameLift solusi Amazon yang dikelola untuk server game khusus dan Server Realtime.

FlexMatchmemasangkan layanan perjodohan dengan mesin aturan yang dapat disesuaikan. Ini memungkinkan Anda merancang cara mencocokkan pemain bersama berdasarkan atribut pemain dan mode permainan yang masuk akal untuk permainan Anda, dan mengandalkan FlexMatch untuk mengelola mur dan baut membentuk grup pemain dan menempatkannya ke dalam game. Lihat detail lebih lanjut tentang pencocokan kustom di FlexMatch contoh set aturan.

FlexMatchdibangun di atas fitur Antrian. Setelah pertandingan terbentuk, FlexMatch berikan detail pertandingan ke antrian pilihan Anda. Antrian mencari sumber daya hosting yang tersedia di GameLift armada Amazon Anda dan memulai sesi permainan baru untuk pertandingan.

Topik di bagian ini mencakup cara menambahkan dukungan pencocokan ke server game dan klien game Anda. Untuk membuat kecocokan untuk game Anda, lihat Membangun mak comblang Amazon GameLift FlexMatch . Untuk informasi lebih lanjut tentang cara FlexMatch kerjanya, lihatBagaimana Amazon GameLift FlexMatch bekerja.

Tambahkan FlexMatch ke klien game

Topik ini menjelaskan cara menambahkan dukungan FlexMatch perjodohan ke layanan game sisi klien Anda. Prosesnya pada dasarnya sama apakah Anda menggunakan FlexMatch hosting GameLift terkelola Amazon atau dengan solusi hosting lain. Untuk mempelajari lebih lanjut tentang FlexMatch dan cara menyiapkan mak comblang khusus untuk game Anda, lihat topik berikut:

- Tutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon
- Bagaimana Amazon GameLift FlexMatch bekerja
- Membangun mak comblang Amazon GameLift FlexMatch
- FlexMatch contoh set aturan

Untuk mengaktifkan FlexMatch perjodohan dalam game Anda, tambahkan fungsionalitas berikut:

- Bersiaplah untuk meminta pencocokan untuk satu atau beberapa pemain (wajib).

- Melacak status permintaan pencocokan (wajib).
- Minta persetujuan pemain untuk pertandingan yang diusulkan (opsional).
- Setelah sesi game dibuat untuk pertandingan baru, dapatkan informasi koneksi pemain dan bergabunglah dalam game.

Bersiaplah untuk meminta pencocokan untuk pemain

Kami sangat menyarankan agar klien game Anda membuat permintaan pencocokan melalui layanan game sisi klien. Dengan menggunakan sumber tepercaya, Anda dapat lebih mudah melindungi dari upaya peretasan dan data pemain palsu. Jika game Anda memiliki directory service sesi, ini adalah opsi yang baik untuk menangani permintaan pencocokan.

Untuk mempersiapkan layanan klien Anda, lakukan tugas berikut:

- Tambahkan Amazon GameLift API. Layanan klien Anda menggunakan fungsionalitas di Amazon GameLift API, yang merupakan bagian dari AWS SDK. Lihat Amazon GameLift SDKs untuk layanan klien untuk mempelajari lebih lanjut tentang AWS SDK dan mengunduh versi terbaru. Tambahkan ini SDK ke proyek layanan klien game Anda.
- Siapkan sistem tiket perjodohan. Semua permintaan pencocokan harus diberikan ID tiket unik. Anda memerlukan mekanisme untuk menghasilkan unik IDs dan menetapkannya ke permintaan kecocokan baru. ID tiket dapat menggunakan format string apa pun, hingga maksimum 128 karakter.
- Dapatkan informasi mak comblang. Dapatkan nama konfigurasi pencocokan yang ingin Anda gunakan. Anda juga memerlukan daftar atribut pemain yang diperlukan pembuat kecocokan, yang didefinisikan dalam rangkaian aturan pembuat kecocokan.
- Dapatkan data pemain. Sediakan cara untuk mendapatkan data yang relevan untuk setiap pemain. Ini termasuk ID pemain, nilai atribut pemain, dan data latensi yang diperbarui untuk setiap wilayah tempat pemain kemungkinan ditempatkan ke dalam game.
- (opsional) Aktifkan isi ulang pertandingan. Tentukan cara Anda ingin melakukan backfill pada game yang cocok yang ada. Jika pembuat kecocokan Anda memiliki mode backfill yang disetel ke "manual", Anda mungkin ingin menambahkan dukungan backfill ke game Anda. Jika mode backfill disetel ke “otomatis”, Anda mungkin perlu cara untuk mematikannya untuk setiap sesi game. Pelajari selengkapnya tentang mengelola backfill kecocokan di Isi ulang game yang ada dengan FlexMatch.

Permintaan pencocokan untuk pemain

Tambahkan kode ke layanan klien Anda untuk membuat dan mengelola permintaan perjodohan ke mak comblang. FlexMatch Proses meminta FlexMatch perjodohan identik untuk game yang digunakan dengan GameLiftmanaged hosting FlexMatch Amazon dan untuk game yang digunakan FlexMatch sebagai solusi mandiri.

Buat permintaan pencocokan:

- Hubungi Amazon GameLift API StartMatchmaking. Setiap permintaan harus berisi informasi berikut.

  Pembuat kecocokan

  Nama konfigurasi pencocokan yang akan digunakan untuk permintaan. FlexMatch menempatkan setiap permintaan ke dalam pool untuk mak comblang yang ditentukan, dan permintaan diproses berdasarkan bagaimana mak comblang dikonfigurasi. Ini termasuk menegakkan batas waktu, apakah akan meminta pemain menerima pertandingan, antrean mana yang akan digunakan saat menempatkan sesi game yang dihasilkan, dll. Pelajari selengkapnya tentang pembuat kecocokan dan set aturan di Desain FlexMatch mak comblang.

  ID Tiket

  ID tiket unik yang ditetapkan untuk permintaan. Segala sesuatu yang terkait dengan permintaan, termasuk peristiwa dan pemberitahuan, akan merujuk pada ID tiket.

  Data pemain

  Daftar pemain yang ingin Anda buatkan kecocokan. Jika salah satu pemain dalam permintaan tidak memenuhi persyaratan pertandingan, berdasarkan aturan pertandingan dan minimum latensi, permintaan pencocokan tidak akan menghasilkan pertandingan yang sukses. Anda dapat menyertakan hingga sepuluh pemain dalam permintaan kecocokan. Ketika ada beberapa pemain dalam permintaan, cobalah FlexMatch untuk membuat satu pertandingan dan menetapkan semua pemain ke tim yang sama (dipilih secara acak). Jika permintaan berisi terlalu banyak pemain untuk ditampung di salah satu tim pertandingan, permintaan akan gagal dicocokkan. Misalnya, jika Anda telah menyiapkan pembuat kecocokan untuk membuat pertandingan 2v2 (dua tim yang terdiri dari dua pemain), Anda tidak dapat mengirim permintaan pencocokan yang berisi lebih dari dua pemain.

> **Note**
>
> Seorang pemain (diidentifikasi dengan ID pemainnya) hanya dapat disertakan dalam satu permintaan pencocokan aktif dalam satu waktu. Saat Anda membuat permintaan baru untuk seorang pemain, setiap tiket pencocokan aktif dengan ID pemain yang sama akan dibatalkan secara otomatis.

Untuk setiap pemain yang terdaftar, sertakan data berikut:

- ID Pemain –Setiap pemain harus memiliki ID pemain unik yang Anda buatkan. Lihat Menghasilkan pemain IDs.
- Atribut pemain – Jika pembuat kecocokan yang digunakan memanggil atribut pemain, permintaan harus menyediakan atribut tersebut untuk setiap pemain. Atribut pemain yang diperlukan ditentukan dalam set aturan pembuat kecocokan, yang juga menentukan tipe data untuk atribut tersebut. Sebuah atribut pemain bersifat opsional hanya ketika aturan set menentukan nilai default untuk atribut. Jika permintaan pertandingan tidak memberikan atribut pemain yang diperlukan untuk semua pemain, permintaan pencocokan tidak akan pernah berhasil. Pelajari selengkapnya tentang set aturan pembuat kecocokan dan atribut pemain di Bangun seperangkat FlexMatch aturan dan FlexMatch contoh set aturan.
- Latensi pemain – Jika pembuat kecocokan yang digunakan memiliki aturan latensi pemain, permintaan harus melaporkan latensi untuk setiap pemain. Data latensi pemain adalah daftar satu atau lebih nilai per pemain. Ini mewakili latensi yang dialami pemain untuk wilayah dalam antrean pembuat kecocokan. Jika tidak ada nilai latensi untuk pemain yang disertakan dalam permintaan, pemain tidak dapat dicocokkan, dan permintaan gagal.

Mengambil detail permintaan kecocokan:

- Setelah permintaan kecocokan dikirim, Anda dapat melihat detail permintaan DescribeMatchmakingdengan menelepon dengan ID tiket permintaan. Panggilan ini mengembalikan informasi permintaan, termasuk status saat ini. Setelah permintaan berhasil diselesaikan, tiket juga berisi informasi yang dibutuhkan klien game untuk terhubung ke kecocokan.

Membatalkan permintaan kecocokan:

- Anda dapat membatalkan permintaan perjodohan kapan saja StopMatchmakingdengan menelepon dengan ID tiket permintaan.

Melacak peristiwa pencocokan

Siapkan notifikasi untuk melacak peristiwa yang GameLift dipancarkan Amazon untuk proses perjodohan. Anda dapat mengatur notifikasi baik secara langsung, dengan membuat SNS topik, atau dengan menggunakan Amazon EventBridge. Untuk informasi lebih lanjut tentang mengatur notifikasi, lihat Siapkan pemberitahuan FlexMatch acara. Setelah Anda mengatur notifikasi, tambahkan listener di layanan klien Anda untuk mendeteksi peristiwa dan merespons sesuai kebutuhan.

Sebaiknya Anda mencadangkan notifikasi dengan melakukan polling secara berkala untuk pembaruan status ketika periode waktu yang signifikan berlalu tanpa notifikasi. Untuk meminimalkan dampak pada performa pencocokan, pastikan untuk melakukan polling setelah menunggu setidaknya 30 detik setelah tiket pencocokan dikirimkan atau setelah notifikasi terakhir diterima.

Ambil tiket permintaan perjodohan, termasuk status saat ini, dengan menelepon DescribeMatchmakingdengan ID tiket permintaan. Sebaiknya polling dilakukan maksimal sekali setiap 10 detik. Pendekatan ini hanya untuk digunakan selama skenario pengembangan volume rendah.

> **Note**
>
> Anda harus menyiapkan game Anda dengan pemberitahuan peristiwa sebelum Anda memiliki penggunaan pencocokan bervolume tinggi, seperti dengan pengujian beban pra-produksi. Semua game dalam rilis publik harus menggunakan pemberitahuan terlepas dari volumenya. Pendekatan polling berkelanjutan hanya cocok untuk game yang sedang dikembangkan dengan penggunaan pencocokan yang rendah.

Minta persetujuan pemain

Jika Anda menggunakan pembuat kecocokan yang mengaktifkan persetujuan pemain, tambahkan kode ke layanan klien Anda untuk mengelola proses persetujuan pemain. Proses mengelola penerimaan pemain identik untuk game yang digunakan dengan hosting yang GameLift dikelola FlexMatch Amazon dan untuk game yang digunakan FlexMatch sebagai solusi mandiri.

Meminta persetujuan pemain untuk kecocokan yang diusulkan:

1. Deteksi kapan pertandingan yang diusulkan membutuhkan penerimaan pemain. Memantau tiket pencocokan untuk mendeteksi ketika status berubah ke `REQUIRES_ACCEPTANCE`. Perubahan pada status ini memicu FlexMatch peristiwa`MatchmakingRequiresAcceptance`.
2. Dapatkan penerimaan dari semua pemain. Buat mekanisme untuk menyajikan detail kecocokan yang diusulkan kepada setiap pemain di tiket pencocokan. Pemain harus dapat menunjukkan bahwa mereka menyetujui atau menolak kecocokan yang diajukan. Anda dapat mengambil detail kecocokan dengan menelepon DescribeMatchmaking. Pemain memiliki waktu terbatas untuk merespons sebelum mak comblang menarik pertandingan yang diusulkan dan melanjutkan.
3. Laporkan tanggapan pemain ke FlexMatch. Laporkan tanggapan pemain dengan menelepon AcceptMatchdengan menerima atau menolak. Semua pemain dalam permintaan pencocokan harus menyetujui kecocokan agar dapat dilanjutkan.
4. Tangani tiket dengan penerimaan yang gagal. Permintaan gagal ketika pemain mana pun dalam kecocokan yang diusulkan menolak kecocokan atau gagal merespons dalam batas waktu persetujuan. Tiket untuk pemain yang menerima kecocokan secara otomatis dikembalikan ke kolam tiket. Tiket untuk pemain yang tidak menerima pertandingan pindah ke FAILURE status dan tidak lagi diproses. Untuk tiket dengan banyak pemain, jika ada pemain di tiket yang tidak menyetujui kecocokan, seluruh tiket gagal.

Connect ke pertandingan

Tambahkan kode ke layanan klien Anda untuk menangani kecocokan yang berhasil dibentuk (status `COMPLETED` atau peristiwa `MatchmakingSucceeded`). Ini termasuk memberi tahu pemain yang cocok dan menyerahkan informasi koneksi ke klien game mereka.

Untuk game yang menggunakan hosting GameLift terkelola Amazon, ketika permintaan perjodohan berhasil dipenuhi, informasi koneksi sesi game ditambahkan ke tiket perjodohan. Ambil tiket perjodohan yang sudah selesai dengan menelepon. DescribeMatchmaking Informasi koneksi mencakup alamat IP dan port sesi game, serta ID sesi pemain untuk setiap ID pemain. Pelajari selengkapnya di GameSessionConnectionInfo. Klien game Anda dapat menggunakan informasi ini untuk terhubung langsung ke sesi game untuk kecocokan. Permintaan koneksi harus mencakup ID sesi pemain dan ID pemain. Data ini mengaitkan pemain yang terhubung ke data pertandingan sesi permainan, yang mencakup tugas tim (lihat GameSession).

Untuk game yang menggunakan solusi hosting lain, termasuk Amazon GameLift FleetiQ, Anda harus membangun mekanisme untuk memungkinkan pemain pertandingan terhubung ke sesi permainan yang sesuai.

Permintaan pencocokan sampel

Potongan kode berikut membangun permintaan pencocokan untuk beberapa pembuat kecocokan yang berbeda. Seperti yang dijelaskan, permintaan harus menyediakan atribut pemain yang diperlukan oleh pembuat kecocokan yang digunakan, seperti yang didefinisikan dalam set aturan pembuat kecocokan. Atribut yang diberikan harus menggunakan tipe data yang sama, angka (N) atau string (S) yang didefinisikan dalam set aturan.

```
# 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)
```

Tambahkan FlexMatch ke server game yang GameLift dihosting Amazon

Topik ini menjelaskan cara menambahkan dukungan FlexMatch perjodohan ke server game khusus yang menggunakan hosting GameLift terkelola Amazon. Untuk mempelajari lebih lanjut tentang menambahkan FlexMatch ke game Anda, lihat topik berikut:

- Bagaimana Amazon GameLift FlexMatch bekerja
- Tutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon

Informasi dalam topik ini mengasumsikan bahwa Anda telah berhasil mengintegrasikan GameLift Server Amazon SDK ke dalam proyek server game Anda, seperti yang dijelaskan dalam Tambahkan Amazon GameLift ke server game Anda. Dengan selesainya pekerjaan ini, Anda memiliki sebagian besar mekanisme yang Anda butuhkan. Bagian dalam topik ini mencakup pekerjaan yang tersisa untuk menangani game yang disiapkan FlexMatch.

Mengatur server game Anda untuk pencocokan

Untuk menyiapkan server game Anda untuk menangani game yang cocok, selesaikan tugas berikut.

1. Mulai sesi permainan yang dibuat dengan perjodohan. Untuk meminta sesi permainan baru, Amazon GameLift mengirimkan `onStartGameSession()` permintaan ke server game Anda dengan objek sesi game (lihat GameSession). Server game Anda menggunakan informasi sesi game, termasuk data game yang disesuaikan, untuk memulai sesi game yang diminta. Untuk lebih detailnya, lihat Mulai sesi game.

   Untuk game yang cocok, objek sesi game juga berisi kumpulan data pembuat kecocokan. Data pembuat kecocokan mencakup informasi yang dibutuhkan server game Anda untuk memulai sesi game baru untuk kecocokan tersebut. Ini termasuk struktur tim pertandingan, tugas tim, dan atribut pemain tertentu yang mungkin relevan dengan game Anda. Misalnya, game Anda mungkin membuka fitur atau level tertentu berdasarkan tingkat keterampilan pemain rata-rata, atau memilih peta berdasarkan preferensi pemain. Pelajari selengkapnya di Bekerja dengan data pembuat kecocokan.
2. Tangani koneksi pemain. Saat menghubungkan ke game yang cocok, klien game merujuk ID pemain dan ID sesi pemain (lihat Memvalidasi pemain baru). Server game Anda menggunakan ID pemain untuk mengaitkan pemain yang masuk dengan informasi pemain di data pembuat kecocokan. Data Matchmaker mengidentifikasi tugas tim pemain dan dapat memberikan informasi lain untuk mewakili pemain dengan benar dalam game.
3. Laporkan saat pemain meninggalkan permainan. Pastikan server game Anda memanggil Server API `RemovePlayerSession()` untuk melaporkan pemain yang dijatuhkan (lihat Laporkan sesi pemain berakhir). Langkah ini penting jika Anda menggunakan FlexMatch backfill untuk mengisi slot kosong di game yang ada. Sangat penting jika game Anda memulai permintaan backfill melalui layanan game sisi klien. Pelajari lebih lanjut tentang menerapkan FlexMatch pengurukan diIsi ulang game yang ada dengan FlexMatch.
4. Minta pemain baru untuk sesi permainan yang cocok (opsional). Tentukan cara Anda ingin melakukan backfill pada game yang cocok yang ada. Jika pembuat kecocokan Anda memiliki mode backfill yang disetel ke "manual", Anda mungkin ingin menambahkan dukungan backfill

ke game Anda. Jika mode backfill diatur ke "otomatis", Anda mungkin memerlukan cara untuk menonaktifkannya untuk setiap sesi game. Misalnya, Anda mungkin ingin berhenti melakukan backfill pada sesi game setelah titik tertentu dalam game tercapai. Pelajari selengkapnya tentang mengelola backfill kecocokan di Isi ulang game yang ada dengan FlexMatch.

Bekerja dengan data pembuat kecocokan

Server game Anda harus dapat mengenali dan menggunakan informasi game dalam suatu GameSessionobjek. GameLift Layanan Amazon meneruskan objek-objek ini ke server game Anda setiap kali sesi game dimulai atau diperbarui. Informasi sesi game inti mencakup ID dan nama sesi game, jumlah pemain maksimum, informasi koneksi, dan data game khusus (jika tersedia).

Untuk sesi permainan yang dibuat menggunakan FlexMatch, `GameSession` objek juga berisi satu set data mak comblang. Selain ID kecocokan unik, ID ini mengidentifikasi pembuat kecocokan yang menciptakan kecocokan dan menggambarkan tim, tugas tim, dan pemain. Ini mencakup atribut pemain dari permintaan pencocokan asli (lihat objek Pemain). Itu tidak termasuk latensi pemain; jika Anda memerlukan data latensi pada pemain saat ini, seperti untuk backfill kecocokan, kami sarankan untuk mendapatkan data baru.

> **Note**
>
> Data Matchmaker menentukan konfigurasi pencocokan lengkapARN, yang mengidentifikasi nama konfigurasi, akun, dan wilayah. AWS Saat meminta backfill yang cocok dari klien atau layanan game, Anda hanya memerlukan nama konfigurasi. Anda dapat mengekstrak nama konfigurasi dengan menguraikan string yang mengikuti ":matchmakingconfiguration/". Dalam contoh yang ditampilkan, nama konfigurasi perjodohan adalah ""MyMatchmakerConfig.

Berikut ini JSON menunjukkan sekumpulan data mak comblang yang khas. Contoh ini menjelaskan game dua pemain, dengan pemain yang dicocokkan berdasarkan peringkat keterampilan dan level tertinggi yang dicapai. Pembuat kecocokan juga cocok berdasarkan karakter, dan memastikan bahwa pemain yang cocok memiliki setidaknya satu preferensi peta yang sama. Dalam skenario ini, server game harus dapat menentukan peta mana yang paling disukai dan menggunakannya dalam sesi game.

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

Isi ulang game yang ada dengan FlexMatch

Match backfill menggunakan FlexMatch mekanisme Anda untuk menemukan pemain baru untuk sesi permainan yang cocok. Meskipun Anda selalu dapat menambahkan pemain untuk game apapun (lihat Bergabung dengan pemain ke sesi game), backfill kecocokan memastikan bahwa pemain baru memenuhi kriteria kecocokan yang sama dengan pemain saat ini. Selain itu, backfill kecocokan menetapkan pemain baru ke tim, mengelola penerimaan pemain, dan mengirimkan informasi kecocokan yang diperbarui ke server game. Pelajari selengkapnya tentang backfill kecocokan di FlexMatch proses perjodohan.

> **Note**
>
> FlexMatch backfill saat ini tidak tersedia untuk game yang menggunakan Server Realtime.

Ada dua jenis mekanisme backfill:

- Untuk mengisi sesi game yang dimulai dengan jumlah pemain yang kurang dari jumlah maksimum yang diizinkan, aktifkan backfill otomatis.
- Untuk mengganti pemain yang keluar dari sesi game yang sedang berlangsung, tambahkan fungsionalitas ke server game Anda untuk mengirim permintaan backfill.

Mengaktifkan backfill otomatis

Dengan pengisian ulang pertandingan otomatis, Amazon GameLift secara otomatis memicu permintaan pengisian ulang setiap kali sesi permainan dimulai dengan satu atau lebih slot pemain yang tidak terisi. Fitur ini memungkinkan game dimulai segera setelah jumlah minimum pemain yang cocok ditemukan dan mengisi slot yang tersisa nanti saat pemain tambahan dicocokkan. Anda dapat memilih untuk menghentikan backfill otomatis kapan saja.

Sebagai contoh, pertimbangkan permainan yang dapat menampung enam hingga sepuluh pemain. FlexMatch awalnya menempatkan enam pemain, membentuk pertandingan, dan memulai sesi permainan baru. Dengan backfill otomatis, sesi game baru bisa langsung meminta tambahan empat pemain. Tergantung pada gaya permainan, kami mungkin ingin mengizinkan pemain baru untuk bergabung kapan saja selama sesi game. Atau, kita mungkin ingin menghentikan backfill otomatis setelah tahap penyiapan awal dan sebelum gameplay dimulai.

Untuk menambahkan backfill otomatis ke permainan Anda, buat pembaruan berikut untuk game Anda.

1. Aktifkan pengisian ulang otomatis. Backfill otomatis dikelola dalam konfigurasi pencocokan. Bila diaktifkan, backfill digunakan dengan semua sesi game yang cocok yang dibuat dengan pembuat kecocokan itu. Amazon GameLift mulai membuat permintaan pengisian ulang untuk sesi game non-penuh segera setelah sesi game dimulai di server game.

   Untuk mengaktifkan pengisian ulang otomatis, buka konfigurasi kecocokan dan atur mode pengisian ulang ke ""AUTOMATIC. Untuk detail selengkapnya, lihat Membuat konfigurasi pencocokan
2. Aktifkan pemrioritasan backfill. Sesuaikan proses pencocokan Anda untuk memprioritaskan pengisian permintaan backfill sebelum membuat kecocokan baru. Dalam rangkaian aturan pencocokan Anda, tambahkan komponen algoritme dan atur prioritas backfill ke “tinggi”. Untuk detail selengkapnya, lihat Menyesuaikan algoritme pencocokan.
3. Perbarui sesi game dengan data mak comblang baru. Amazon GameLift memperbarui server game Anda dengan informasi kecocokan menggunakan fungsi SDK panggilan balik Server

`onUpdateGameSession` (lihat Menginisialisasi proses server). Tambahkan kode ke server game Anda untuk menangani objek sesi game yang diperbarui sebagai hasil dari aktivitas backfill. Pelajari selengkapnya di Memperbarui data kecocokan pada server game.

4. Matikan pengisian ulang otomatis untuk sesi permainan. Anda dapat memilih untuk menghentikan backfill otomatis kapan saja selama sesi game individu. Untuk menghentikan pengisian ulang otomatis, tambahkan kode ke klien game atau server game Anda untuk melakukan GameLift API panggilan StopMatchmakingAmazon. Panggilan ini memerlukan ID tiket. Gunakan ID tiket backfill dari permintaan backfill terbaru. Anda bisa mendapatkan informasi ini dari sesi game pencocokan data, yang diperbarui seperti yang dijelaskan pada langkah sebelumnya.

Kirim permintaan backfill (dari server game)

Anda dapat memulai permintaan backfill kecocokan langsung dari proses server game yang menghosting sesi game. Proses server memiliki up-to-date informasi terbanyak tentang pemain saat ini yang terhubung ke permainan dan status slot pemain kosong.

Topik ini mengasumsikan bahwa Anda telah membangun FlexMatch komponen yang diperlukan dan berhasil menambahkan proses perjodohan ke server game Anda dan layanan game sisi klien. Untuk detail lebih lanjut tentang pengaturan FlexMatch, lihatTutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon.

Untuk mengaktifkan backfill kecocokan untuk game Anda, tambahkan fungsionalitas berikut:

- Kirim permintaan backfill kecocokan ke pembuat kecocokan dan lacak status permintaan.
- Memperbarui informasi pencocokan untuk sesi game. Lihat Memperbarui data kecocokan pada server game.

Seperti fungsi server lainnya, server game menggunakan Amazon GameLift ServerSDK. Ini SDK tersedia dalam C ++ dan C #.

Untuk membuat permintaan backfill kecocokan dari server game Anda, selesaikan tugas-tugas berikut.

1. Memicu permintaan pengisian ulang kecocokan. Umumnya, Anda ingin memulai permintaan backfill setiap kali game yang cocok memiliki satu slot pemain kosong atau lebih. Anda mungkin ingin mengaitkan permintaan backfill dengan keadaan tertentu, seperti untuk mengisi peran

karakter penting atau menyeimbangkan tim. Anda mungkin juga ingin membatasi aktivitas backfill berdasarkan usia sesi game.

2. Buat permintaan isi ulang. Tambahkan kode untuk membuat dan mengirim permintaan pengisian ulang kecocokan ke FlexMatch mak comblang. Permintaan isi ulang ditangani menggunakan server ini: APIs

   - StartMatchBackfill()
   - StopMatchBackfill()

   Untuk membuat permintaan backfill, hubungi `StartMatchBackfill` dengan informasi berikut. Untuk membatalkan permintaan backfill, hubungi `StopMatchBackfill` dengan ID tiket permintaan backfill.

   - ID Tiket — Berikan ID tiket pencocokan (atau memilih untuk membuatnya otomatis). Anda dapat menggunakan mekanisme yang sama untuk menetapkan tiket IDs ke permintaan perjodohan dan pengisian ulang. Tiket untuk pencocokan dan backfill diproses dengan cara yang sama.
   - Pembuat kecocokan — Identifikasi pembuat kecocokan mana yang akan digunakan untuk permintaan backfill. Umumnya, Anda harus menggunakan pembuat kecocokan yang sama dengan yang digunakan untuk membuat kecocokan asli. Permintaan ini membutuhkan konfigurasi perjodohan. ARN Informasi ini disimpan dalam objek sesi permainan (GameSession), yang disediakan untuk proses server oleh Amazon GameLift saat mengaktifkan sesi permainan. Konfigurasi ARN perjodohan termasuk dalam properti. `MatchmakerData`
   - Sesi permainan ARN — Identifikasi sesi permainan yang diisi kembali. Anda bisa mendapatkan sesi permainan ARN dengan memanggil server API GetGameSessionId(). Selama proses matchmaking, tiket untuk permintaan baru tidak memiliki ID sesi game, sedangkan tiket untuk permintaan isi ulang memiliki ID sesi game. Kehadiran di ID sesi game merupakan salah satu cara untuk membedakan antara tiket kecocokan baru dan tiket backfill.
   - Data pemain — Sertakan informasi pemain (Pemain) untuk semua pemain saat ini dalam sesi game yang sedang Anda backfill. Informasi ini memungkinkan pembuat kecocokan untuk menemukan kecocokan pemain terbaik untuk para pemain saat ini dalam sesi game. Anda harus menyertakan keanggotaan tim untuk setiap pemain. Jangan tentukan tim jika Anda tidak menggunakan backfill. Jika server game Anda telah melaporkan status koneksi pemain secara akurat, Anda seharusnya dapat memperoleh data ini sebagai berikut:

1. Proses server hosting sesi permainan harus memiliki up-to-date informasi paling banyak yang pemain saat ini terhubung ke sesi permainan.
2. Untuk mendapatkan pemainIDs, atribut, dan tugas tim, tarik data pemain dari objek sesi permainan (GameSession), `MatchmakerData` properti (lihatBekerja dengan data pembuat kecocokan). Data pembuat kecocokan mencakup semua pemain yang dicocokkan dengan sesi game, jadi Anda harus menarik data pemain hanya untuk pemain yang saat ini terhubung.
3. Untuk latensi pemain, jika pembuat kecocokan meminta data latensi, kumpulkan nilai latensi baru dari semua pemain saat ini dan sertakan di setiap objek `Player`. Jika data latensi dihilangkan dan pembuat kecocokan memiliki aturan latensi, permintaan tidak akan berhasil dicocokkan. Permintaan backfill memerlukan data latensi hanya untuk wilayah tempat sesi game saat ini berada. Anda bisa mendapatkan wilayah sesi permainan dari `GameSessionId` properti `GameSession` objek; nilai ini adalahARN, yang mencakup wilayah.

3. Lacak status permintaan isi ulang. Amazon GameLift memperbarui server game Anda tentang status permintaan pengisian ulang menggunakan fungsi SDK panggilan balik Server `onUpdateGameSession` (lihat Menginisialisasi proses server). Menambahkan kode untuk menangani pesan status—serta objek sesi game yang diperbarui sebagai hasil dari permintaan backfill yang berhasil—di Memperbarui data kecocokan pada server game.

   Pembuat kecocokan hanya dapat memproses satu permintaan backfill kecocokan dari sesi game dalam satu waktu. Jika Anda perlu membatalkan permintaan, hubungi StopMatchBackfill(). Jika Anda perlu mengubah permintaan, panggil `StopMatchBackfill` kemudian kirimkan permintaan yang diperbarui.

Kirim permintaan backfill (dari layanan klien)

Sebagai alternatif untuk mengirim permintaan backfill dari server game, Anda mungkin ingin mengirimnya dari layanan game sisi klien. Untuk menggunakan opsi ini, layanan sisi klien harus memiliki akses ke data terkini tentang aktivitas sesi game dan koneksi pemain; jika game Anda menggunakan layanan direktori sesi, ini mungkin pilihan yang baik.

Topik ini mengasumsikan bahwa Anda telah membangun FlexMatch komponen yang diperlukan dan berhasil menambahkan proses perjodohan ke server game Anda dan layanan game sisi klien. Untuk detail lebih lanjut tentang pengaturan FlexMatch, lihatTutorial: Integrasikan FlexMatch dengan GameLift hosting Amazon.

Untuk mengaktifkan backfill kecocokan untuk game Anda, tambahkan fungsionalitas berikut:

- Kirim permintaan backfill kecocokan ke pembuat kecocokan dan lacak status permintaan.
- Perbarui informasi pertandingan untuk sesi game. Lihat Memperbarui data kecocokan pada server game

Seperti fungsionalitas klien lainnya, layanan permainan sisi klien menggunakan dengan AWS SDK Amazon. GameLift API Ini SDK tersedia dalam C ++, C #, dan beberapa bahasa lainnya. Untuk deskripsi umum klienAPIs, lihat API Referensi GameLift Layanan Amazon, yang menjelaskan layanan tingkat rendah API untuk tindakan GameLift terkait Amazon dan menyertakan tautan ke panduan referensi khusus bahasa.

Untuk menyiapkan layanan game sisi klien untuk mengisi ulang game yang cocok, selesaikan tugas berikut.

1. Memicu permintaan untuk penimbunan ulang. Umumnya, game memulai permintaan backfill setiap kali game yang cocok memiliki satu atau beberapa slot pemain kosong. Anda mungkin ingin mengaitkan permintaan backfill dengan keadaan tertentu, seperti untuk mengisi peran karakter penting atau menyeimbangkan tim. Anda mungkin juga ingin membatasi backfill berdasarkan usia sesi game. Apa pun yang Anda gunakan untuk pemicu, minimal Anda akan membutuhkan informasi berikut. Anda bisa mendapatkan informasi ini dari objek sesi permainan (GameSession) dengan menelepon DescribeGameSessionsdengan ID sesi game.

   - Jumlah slot pemain yang saat ini kosong. Nilai ini dapat dihitung dari batas pemain maksimum sesi game dan jumlah pemain saat ini. Jumlah pemain saat ini diperbarui setiap kali server game Anda menghubungi GameLift layanan Amazon untuk memvalidasi koneksi pemain baru atau melaporkan pemain yang terputus.

   - Kebijakan pembuatan. Pengaturan ini menunjukkan apakah sesi game saat ini menerima pemain baru.

   Objek sesi game berisi informasi lain yang berpotensi berguna, termasuk waktu mulai sesi game, properti game kustom, dan data pembuat kecocokan.
2. Buat permintaan isi ulang. Tambahkan kode untuk membuat dan mengirim permintaan pengisian ulang kecocokan ke FlexMatch mak comblang. Permintaan isi ulang ditangani menggunakan klien ini: APIs

- StartMatchBackfill
- StopMatchmaking

Untuk membuat permintaan backfill, panggil `StartMatchBackfill` dengan menyertakan informasi berikut. Permintaan backfill mirip dengan permintaan pencocokan (lihat Permintaan pencocokan untuk pemain), tetapi juga mengidentifikasi sesi game yang ada. Untuk membatalkan permintaan backfill, hubungi `StopMatchmaking` dengan menyertakan ID tiket permintaan backfill.

- ID Tiket — Berikan ID tiket pencocokan (atau memilih untuk membuatnya otomatis). Anda dapat menggunakan mekanisme yang sama untuk menetapkan tiket IDs ke permintaan perjodohan dan pengisian ulang. Tiket untuk pencocokan dan backfill diproses dengan cara yang sama.
- Pembuat kecocokan — Identifikasi nama konfigurasi pencocokan untuk digunakan. Umumnya, Anda ingin menggunakan pembuat kecocokan yang sama untuk backfill yang digunakan untuk membuat kecocokan asli. Informasi ini ada di objek sesi permainan (GameSession), `MatchmakerData` properti, di bawah konfigurasi perjodohan. ARN Nilai nama adalah string berikut "" konfigurasi perjodohan / ". (Misalnya, dalam ARN nilai “arn:aws:gamelift:us-west-2:111122223333:matchmakingConfiguration/mm-4v4", nama konfigurasi perjodohan adalah “mm-4v4".)
- Sesi permainan ARN - Tentukan sesi permainan yang diisi kembali. Gunakan `GameSessionId` properti dari objek sesi permainan; ID ini menggunakan ARN nilai yang Anda butuhkan. Tiket perjodohan (MatchmakingTicket) untuk permintaan isi ulang memiliki ID sesi permainan saat sedang diproses; tiket untuk permintaan perjodohan baru tidak mendapatkan ID sesi permainan sampai pertandingan ditempatkan; kehadiran ID sesi permainan adalah salah satu cara untuk membedakan antara tiket untuk pertandingan baru dan tiket untuk pengisian ulang.
- Data pemain — Sertakan informasi pemain (Pemain) untuk semua pemain saat ini dalam sesi game yang sedang Anda backfill. Informasi ini memungkinkan pembuat kecocokan untuk menemukan kecocokan pemain terbaik untuk para pemain saat ini dalam sesi game. Anda harus menyertakan keanggotaan tim untuk setiap pemain. Jangan tentukan tim jika Anda tidak menggunakan backfill. Jika server game Anda telah melaporkan status koneksi pemain secara akurat, Anda seharusnya dapat memperoleh data ini sebagai berikut:

1. Call DescribePlayerSessions() dengan ID sesi game untuk menemukan semua pemain yang saat ini terhubung ke sesi permainan. Setiap sesi pemain menyertakan ID pemain. Anda dapat menambahkan filter status untuk mengambil sesi pemain aktif saja.
2. Tarik data pemain dari objek sesi permainan (GameSession), `MatchmakerData` properti (lihatBekerja dengan data pembuat kecocokan. Gunakan pemain yang IDs diperoleh pada langkah sebelumnya untuk mendapatkan data hanya untuk pemain yang terhubung saat ini. Karena data pembuat kecocokan tidak diperbarui saat pemain keluar, Anda hanya perlu mengekstrak data untuk pemain saat ini.
3. Untuk latensi pemain, jika pembuat kecocokan meminta data latensi, kumpulkan nilai latensi baru dari semua pemain saat ini dan sertakan dalam objek `Player`. Jika data latensi dihilangkan dan pembuat kecocokan memiliki aturan latensi, permintaan tidak akan berhasil dicocokkan. Permintaan backfill memerlukan data latensi hanya untuk wilayah tempat sesi game saat ini berada. Anda bisa mendapatkan wilayah sesi permainan dari `GameSessionId` properti `GameSession` objek; nilai ini adalahARN, yang mencakup wilayah.

3. Lacak status permintaan isi ulang. Tambahkan kode untuk mendengarkan pembaruan status tiket pencocokan. Anda dapat menggunakan mekanisme yang diatur untuk melacak tiket untuk permintaan pencocokan baru (lihat Melacak peristiwa pencocokan) menggunakan pemberitahuan acara (lebih disukai) atau polling. Meskipun Anda tidak perlu memicu aktivitas penerimaan pemain dengan permintaan backfill, dan informasi pemain diperbarui di server game, Anda masih perlu memantau status tiket untuk menangani kegagalan permintaan dan pengiriman ulang.

   Pembuat kecocokan hanya dapat memproses satu permintaan backfill kecocokan dari sesi game pada satu waktu. Jika Anda perlu membatalkan permintaan, hubungi StopMatchmaking. Jika Anda perlu mengubah permintaan, panggil `StopMatchmaking` kemudian kirim permintaan yang diperbarui.

   Setelah permintaan pengurukan pertandingan berhasil, server game Anda menerima `GameSession` objek dan menangani tugas-tugas yang diperlukan untuk bergabung dengan pemain baru untuk sesi permainan. Lihat selengkapnya di Memperbarui data kecocokan pada server game.

Memperbarui data kecocokan pada server game

Tidak peduli bagaimana Anda memulai permintaan pengisian ulang pertandingan di game Anda, server game Anda harus dapat menangani pembaruan sesi game yang GameLift diberikan Amazon sebagai hasil dari permintaan pengisian ulang pertandingan.

Saat Amazon GameLift menyelesaikan permintaan pengisian ulang kecocokan—berhasil atau tidak—, Amazon akan memanggil server game Anda menggunakan fungsi callback. `onUpdateGameSession` Panggilan ini memiliki tiga parameter input: ID tiket isi ulang pertandingan, pesan status, dan GameSession objek yang berisi data up-to-date perjodohan terbanyak termasuk informasi pemain. Anda perlu menambahkan kode berikut ke server permainan Anda sebagai bagian dari integrasi server game:

1. Menerapkan fungsi `onUpdateGameSession`. Fungsi ini harus mampu menangani pesan status berikut (`updateReason`):
   - MATCHMAKING_ DATA _ UPDATED — Pemain baru berhasil dicocokkan dengan sesi permainan. Objek `GameSession` berisi data pembuat kecocokan yang diperbarui, termasuk data pemain tentang pemain yang ada dan pemain baru yang cocok.
   - BACKFILL_ FAILED — Upaya pengisian ulang pertandingan gagal karena kesalahan internal. Objek `GameSession` tidak berubah.
   - BACKFILL_ TIMED _ OUT — Mak comblang gagal menemukan kecocokan isi ulang dalam batas waktu. Objek `GameSession` tidak berubah.
   - BACKFILL_ CANCELLED — Permintaan pengisian ulang pertandingan dibatalkan dengan panggilan ke StopMatchmaking (klien) atau StopMatchBackfill (server). Objek `GameSession` tidak berubah.
2. Untuk backfill kecocokan yang berhasil, gunakan data pembuat kecocokan yang diperbarui untuk menangani pemain baru saat mereka terhubung ke sesi game. Minimal, Anda harus menggunakan tugas tim untuk pemain baru, serta atribut pemain lain yang diperlukan untuk memulai game.
3. Dalam panggilan server game Anda ke SDK tindakan Server ProcessReady(), tambahkan nama metode `onUpdateGameSession` callback sebagai parameter proses.

Keamanan dengan FlexMatch

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Untuk informasi tentang cara menerapkan model tanggung jawab bersama saat menggunakanFlexMatch, lihat Keamanan di Amazon GameLift.

GameLiftFlexMatchReferensi Amazon

Bagian ini berisi dokumentasi referensi untuk perjodohan dengan Amazon. GameLift FlexMatch

Topik

- GameLift FlexMatch APIReferensi Amazon (AWS SDK)
- FlexMatchbahasa aturan
- FlexMatch acara perjodohan

GameLift FlexMatch APIReferensi Amazon (AWS SDK)

Topik ini menyediakan daftar API operasi berbasis tugas untuk Amazon. GameLift FlexMatch GameLift FlexMatchLayanan Amazon API dikemas ke dalam AWS SDK `aws.gamelift` namespace. Unduh AWS SDK atau lihat dokumentasi GameLift API referensi Amazon.

Amazon GameLift FlexMatch menyediakan layanan perjodohan untuk digunakan dengan game yang di-host dengan solusi GameLift hosting Amazon (termasuk hosting terkelola untuk server game khusus atau Server Realtime, dan hosting di Amazon dengan EC2 Amazon GameLift FleetiQ), serta dengan sistem hosting lain seperti, lokal peer-to-peer, atau komputasi awan primitif. Lihat Panduan GameLift Pengembang Amazon untuk informasi selengkapnya tentang opsi GameLift hosting Amazon lainnya.

Topik

- Menyiapkan aturan dan proses pencocokan
- Meminta kecocokan untuk satu atau beberapa pemain
- Bahasa pemrograman yang tersedia
- Catatan dan SDK versi GameLift FlexMatch rilis Amazon
- Sumber daya GameLift pengembang Amazon

Menyiapkan aturan dan proses pencocokan

Panggil operasi ini untuk membuat FlexMatch mak comblang, konfigurasikan proses perjodohan untuk game Anda, dan tentukan seperangkat aturan khusus untuk membuat pertandingan dan tim.

Konfigurasi perjodohan

- CreateMatchmakingConfiguration— Buat konfigurasi perjodohan dengan instruksi untuk mengevaluasi kelompok pemain dan membangun tim pemain. Saat menggunakan Amazon GameLift untuk hosting, tentukan juga cara membuat sesi permainan baru untuk pertandingan.
- DescribeMatchmakingConfigurations— Ambil konfigurasi perjodohan yang ditentukan wilayah Amazon. GameLift
- UpdateMatchmakingConfiguration— Ubah pengaturan untuk konfigurasi perjodohan. antrian.
- DeleteMatchmakingConfiguration— Hapus konfigurasi perjodohan dari wilayah tersebut.

Aturan perjodohan ditetapkan

- CreateMatchmakingRuleSet— Buat seperangkat aturan untuk digunakan saat mencari pertandingan pemain.
- DescribeMatchmakingRuleSets— Ambil set aturan perjodohan yang ditentukan di wilayah Amazon. GameLift
- ValidateMatchmakingRuleSet— Verifikasi sintaks untuk satu set aturan perjodohan.
- DeleteMatchmakingRuleSet— Hapus aturan perjodohan yang ditetapkan dari wilayah tersebut.

Meminta kecocokan untuk satu atau beberapa pemain

Panggil operasi ini dari layanan klien game Anda untuk mengelola permintaan pencocokan pemain.

- StartMatchmaking— Minta perjodohan untuk satu pemain atau grup yang ingin bermain di pertandingan yang sama.
- DescribeMatchmaking— Dapatkan detail tentang permintaan perjodohan, termasuk status.
- AcceptMatch— Untuk pertandingan yang membutuhkan penerimaan pemain, beri tahu Amazon GameLift saat pemain menerima pertandingan yang diusulkan.
- StopMatchmaking— Batalkan permintaan perjodohan.
- StartMatchBackfill- Minta pertandingan pemain tambahan untuk mengisi slot kosong di sesi permainan yang ada.

Bahasa pemrograman yang tersedia

AWS SDKDengan dukungan untuk Amazon GameLift tersedia dalam bahasa berikut. Untuk informasi tentang dukungan untuk lingkungan pengembangan, lihat dokumentasi untuk setiap bahasa.

- C++ (SDKdokumen) (Amazon) GameLift
- Java (SDKdokumen) (Amazon GameLift)
- . NET(SDKdokumen) (Amazon GameLift)
- Pergi (SDKdokumen) (Amazon GameLift)
- Python (SDKdokumen) (Amazon) GameLift
- Ruby (SDKdokumen) (Amazon) GameLift
- PHP(SDKdokumen) (Amazon GameLift)
- JavaScript/Node.js (SDKdokumen) (Amazon GameLift)

Catatan dan SDK versi GameLift FlexMatch rilis Amazon

Catatan GameLift rilis Amazon memberikan detail tentang FlexMatch fitur, pembaruan, dan perbaikan baru yang terkait dengan layanan. Halaman ini juga menyertakan riwayat GameLift SDK versi Amazon.

Sumber daya GameLift pengembang Amazon

Untuk melihat semua GameLift dokumentasi Amazon dan sumber daya pengembang, lihat halaman beranda GameLift Dokumentasi Amazon.

FlexMatchbahasa aturan

Topik referensi di bagian ini menjelaskan sintaks dan semantik yang digunakan untuk membuat aturan perjodohan untuk digunakan dengan Amazon. GameLift FlexMatch Untuk bantuan rinci dengan menulis aturan perjodohan dan set aturan, lihat. Bangun seperangkat FlexMatch aturan

Topik

- FlexMatchaturan set skema
- FlexMatchaturan set definisi properti
- FlexMatchjenis aturan
- FlexMatchekspresi properti

FlexMatchaturan set skema

FlexMatchset aturan menggunakan skema standar untuk aturan pertandingan kecil dan pertandingan besar. Untuk deskripsi rinci dari setiap bagian, lihatFlexMatchaturan set definisi properti.

Skema kumpulan aturan untuk kecocokan kecil

Skema berikut mendokumentasikan semua properti yang mungkin dan nilai yang diizinkan untuk kumpulan aturan yang digunakan untuk membuat kecocokan hingga 40 pemain.

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

Skema kumpulan aturan untuk kecocokan besar

Skema berikut ini mendokumentasikan semua kemungkinan properti dan nilai yang diizinkan untuk kumpulan aturan yang digunakan untuk membuat kecocokan dengan lebih dari 40 pemain. Jika total `maxPlayers` nilai untuk semua tim dalam kumpulan aturan melebihi 40, maka FlexMatch proses mencocokkan permintaan yang menggunakan aturan ini yang ditetapkan di bawah pedoman pertandingan besar.

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

FlexMatchaturan set definisi properti

Bagian ini mendefinisikan setiap properti dalam skema kumpulan aturan. Untuk bantuan tambahan dalam membuat kumpulan aturan, lihat Bangun seperangkat FlexMatch aturan.

`name`

Label deskriptif untuk kumpulan aturan. Nilai ini tidak terkait dengan nama yang ditetapkan ke GameLift MatchmakingRuleSetsumber daya Amazon. Nilai ini termasuk dalam data perjodohan yang menjelaskan kecocokan yang telah selesai, tetapi tidak digunakan oleh proses Amazon apa pun. GameLift

Nilai yang diizinkan: String

Wajib? Tidak

`ruleLanguageVersion`

Versi bahasa ekspresi FlexMatch properti yang digunakan.

Nilai yang diizinkan: “1.0”

Wajib? Ya

`playerAttributes`

Kumpulan data pemain yang disertakan dalam permintaan pencocokan dan digunakan dalam proses pencocokan. Anda juga dapat mendeklarasikan atribut di sini agar data pemain disertakan dalam data pencocokan yang diteruskan ke server game, meskipun data tersebut tidak digunakan dalam proses pencocokan.

Wajib? Tidak

`name`

Nama unik untuk atribut pemain yang akan digunakan oleh pembuat kecocokan. Nama ini harus cocok dengan nama atribut pemain yang dirujuk dalam permintaan pencocokan.

Nilai yang diizinkan: String

Wajib? Ya

`type`

Jenis data nilai atribut pemain.

Nilai yang diizinkan: "string", "number", "string_list", "string_number_map"

Wajib? Ya

`default`

Nilai default untuk digunakan saat permintaan pencocokan tidak menyediakannya untuk pemain.

Nilai yang diizinkan: Setiap nilai yang diizinkan untuk atribut pemain.

Wajib? Tidak

`algorithm`

Pengaturan konfigurasi opsional untuk menyesuaikan proses pencocokan.

Wajib? Tidak

`strategy`

Metode untuk digunakan saat membangun kecocokan. Jika properti ini tidak disetel, perilaku defaultnya adalah "exhaustiveSearch".

Nilai yang diizinkan:

- "exhaustiveSearch" – Metode pencocokan standar. FlexMatchmembentuk kecocokan di sekitar tiket tertua dalam batch dengan mengevaluasi tiket lain di pangkalan berdasarkan serangkaian aturan kecocokan khusus. Strategi ini digunakan untuk kecocokan dengan 40 pemain atau kurang. Saat menggunakan strategi ini, `batchingPreference` harus diatur ke "random" atau "sorted".
- “balanced” — Metode yang dioptimalkan untuk membentuk kecocokan besar dengan cepat. Strategi ini hanya digunakan untuk kecocokan dengan 41 hingga 200 pemain. Ini membentuk kecocokan dengan menyortir kolam tiket, membangun kecocokan potensial dan menugaskan pemain ke tim, dan kemudian menyeimbangkan setiap tim dalam kecocokan menggunakan atribut pemain yang ditentukan. Misalnya, strategi ini dapat digunakan untuk menyamakan tingkat keterampilan rata-rata semua tim dalam suatu kecocokan. Saat menggunakan strategi ini, `balancedAttribute` harus disetel, dan `batchingPreference` harus disetel ke "Populasi terbesar" atau "Wilayah tercepat". Sebagian besar jenis aturan khusus tidak dikenali dengan strategi ini.

Wajib? Ya

`batchingPreference`

Metode pra-penyortiran yang digunakan sebelum mengelompokkan tiket untuk pembuatan kecocokan. Pra-penyortiran kolam tiket menyebabkan tiket dikelompokkan bersama berdasarkan karakteristik tertentu, yang cenderung meningkatkan keseragaman antar pemain di kecocokan akhir.

Nilai yang diizinkan:

- "random" – Hanya berlaku dengan `strategy` = "exhaustiveSearch". Tidak ada pra-penyortiran dilakukan; tiket di kolam di-batch secara acak. Ini adalah perilaku default untuk strategi pencarian lengkap.
- "sorted" – Hanya berlaku dengan `strategy` = "exhaustiveSearch". Kolam tiket diurutkan sebelumnya berdasarkan atribut pemain yang tercantum di `sortbyAttributes`.
- "largestPopulation" – Hanya berlaku dengan `strategy` = "balanced". Kolam tiket diurutkan sebelumnya berdasarkan wilayah tempat pemain melaporkan tingkat latensi yang dapat diterima. Ini adalah perilaku default untuk strategi yang seimbang.

- "fastestRegion" – Hanya berlaku dengan `strategy` = "balanced". Kolam tiket diurutkan terlebih dahulu berdasarkan wilayah tempat pemain melaporkan tingkat latensi terendah mereka. Kecocokan yang dihasilkan membutuhkan waktu lebih lama untuk diselesaikan, tetapi latensi untuk semua pemain cenderung rendah.

Wajib? Ya

`balancedAttribute`

Nama atribut pemain untuk digunakan saat membuat kecocokan besar dengan strategi seimbang.

Nilai yang diizinkan: Atribut apa pun yang dinyatakan dalam `playerAttributes` dengan `type` = "number".

Wajib? Ya, jika `strategy` = "balanced".

`sortByAttributes`

Daftar atribut pemain untuk digunakan saat menyortir kolam tiket sebelum di-batch. Properti ini hanya digunakan saat pra-penyortiran dengan strategi pencarian lengkap. Urutan daftar atribut menentukan urutan penyortiran. FlexMatchmenggunakan konvensi penyortiran standar untuk nilai alfa dan numerik.

Nilai yang diizinkan: Atribut apa pun yang dinyatakan dalam `playerAttributes`.

Wajib? Ya, jika `batchingPreference` = "sorted".

`backfillPriority`

Metode prioritas untuk mencocokkan tiket backfill. Properti ini menentukan kapan FlexMatch memproses tiket isi ulang dalam batch. Ini hanya digunakan saat pra-penyortiran dengan strategi pencarian lengkap. Jika properti ini tidak disetel, perilaku defaultnya adalah "normal".

Nilai yang diizinkan:

- "normal" – Jenis permintaan tiket (isi ulang atau kecocokan baru) tidak dipertimbangkan saat membentuk kecocokan.
- “tinggi” - Kumpulan tiket diurutkan berdasarkan jenis permintaan (dan kemudian berdasarkan usia), dan FlexMatch upaya untuk mencocokkan tiket isi ulang terlebih dahulu.
- “rendah” - Kumpulan tiket diurutkan berdasarkan jenis permintaan (dan kemudian berdasarkan usia), dan FlexMatch upaya untuk mencocokkan tiket non-isi ulang terlebih dahulu.

Wajib? Tidak

`expansionAgeSelection`

Metode untuk menghitung waktu tunggu untuk perluasan aturan kecocokan. Perluasan digunakan untuk melonggarkan syarat kecocokan jika kecocokan belum diselesaikan setelah beberapa waktu berlalu. Waktu tunggu dihitung berdasarkan usia tiket yang sudah ada di kecocokan yang telah terisi sebagian. Jika properti ini tidak disetel, perilaku defaultnya adalah "newest".

Nilai yang diizinkan:

- "newest" – Waktu tunggu perluasan dihitung berdasarkan tiket dengan stempel waktu pembuatan terbaru dalam kecocokan yang sebagian telah selesai. Perluasan cenderung dipicu lebih lambat, karena satu tiket yang lebih baru dapat memulai kembali jam waktu tunggu.
- "oldest" – Waktu tunggu perluasan dihitung berdasarkan tiket dengan stempel waktu pembuatan tertua dalam kecocokan. Ekspansi cenderung dipicu lebih cepat.

Wajib? Tidak

`teams`

Konfigurasi tim dalam pencocokan. Berikan nama tim dan rentang ukuran untuk setiap tim. Kumpulan aturan harus menentukan setidaknya satu tim.

`name`

Nama unik untuk tim. Nama tim dapat dirujuk dalam aturan dan perluasan. Pada kecocokan yang sukses, pemain ditugaskan berdasarkan nama tim dalam data pencocokan.

Nilai yang diizinkan: String

Wajib? Ya

`maxPlayers`

Jumlah maksimum pemain yang dapat ditetapkan ke tim.

Nilai yang diizinkan: Jumlah

Wajib? Ya

`minPlayers`

Jumlah minimum pemain yang harus ditetapkan ke tim sebelum kecocokan dianggap layak.

Nilai yang diizinkan: Jumlah

Wajib? Ya

`quantity`

Jumlah tim jenis ini untuk dibuat dalam kecocokan. Tim dengan jumlah lebih besar dari 1 ditetapkan dengan nomor tambahan ("Merah_1", "Merah_2", dll.). Jika properti ini tidak diatur, nilai default adalah “1".

Nilai yang diizinkan: Jumlah

Wajib? Tidak

`rules`

Kumpulan pernyataan aturan yang menentukan cara mengevaluasi pemain untuk kecocokan.

Wajib? Tidak

`name`

Nama unik untuk aturan. Semua aturan dalam kumpulan aturan harus memiliki nama yang unik. Nama aturan direferensikan dalam log peristiwa dan metrik yang melacak aktivitas yang terkait dengan aturan.

Nilai yang diizinkan: String

Wajib? Ya

`description`

Deskripsi teks untuk aturan. Informasi ini dapat digunakan untuk mengidentifikasi tujuan aturan. Ini tidak digunakan dalam proses perjodohan.

Nilai yang diizinkan: String

Wajib? Tidak

`type`

Jenis pernyataan aturan. Setiap jenis aturan memiliki properti tambahan yang harus ditetapkan. Untuk detail lebih lanjut tentang struktur dan penggunaan setiap jenis aturan, lihatFlexMatchjenis aturan.

Nilai yang diizinkan:

- “AbsoluteSort” - Mengurutkan menggunakan metode penyortiran eksplisit yang memesan tiket dalam batch berdasarkan apakah atribut pemain tertentu dibandingkan dengan tiket tertua dalam batch.
- "collection" – Mengevaluasi nilai dalam koleksi, seperti atribut pemain yang merupakan koleksi, atau kumpulan nilai untuk beberapa pemain.
- "comparison" – Membandingkan dua nilai.
- “compound” - Mendefinisikan aturan perjodohan majemuk menggunakan kombinasi logis dari aturan lain dalam set aturan. Didukung hanya untuk pertandingan 40 atau lebih sedikit pemain.
- "distance" – Mengukur jarak antara nilai angka.
- “BatchDistance” - Mengukur perbedaan antara nilai atribut dan menggunakannya untuk mengelompokkan permintaan pertandingan.
- “DistanceSort” - Menyortir menggunakan metode penyortiran eksplisit yang memesan tiket dalam batch berdasarkan bagaimana atribut pemain tertentu dengan nilai numerik dibandingkan dengan tiket tertua dalam batch.
- "latency" – Mengevaluasi data latensi regional yang dilaporkan untuk permintaan pencocokan.

Wajib? Ya

`expansions`

Aturan untuk melonggarkan persyaratan kecocokan dari waktu ke waktu ketika kecocokan tidak dapat diselesaikan. Siapkan perluasan sebagai serangkaian langkah yang diterapkan secara bertahap agar kecocokan lebih mudah ditemukan. Secara default, FlexMatch menghitung waktu tunggu berdasarkan usia tiket terbaru yang ditambahkan ke pertandingan. Anda dapat mengubah cara menghitung waktu tunggu perluasan menggunakan properti algoritme `expansionAgeSelection`.

Waktu tunggu perluasan adalah nilai absolut, jadi setiap langkah harus memiliki waktu tunggu lebih lama dari langkah sebelumnya. Misalnya, untuk menjadwalkan rangkaian perluasan bertahap, Anda dapat menggunakan waktu tunggu 30 detik, 40 detik, dan 50 detik. Waktu tunggu tidak boleh melebihi waktu maksimum yang diizinkan untuk permintaan kecocokan, yang diatur dalam konfigurasi pencocokan.

Wajib? Tidak

`target`

Aturan menyetel elemen agar rileks. Anda dapat mengendurkan properti ukuran tim atau properti pernyataan aturan apa pun. Sintaksnya adalah "<component name>[<rule/team name>].<property name>". Misalnya, untuk mengubah ukuran minimum tim: `teams[Red, Yellow].minPlayers`. Untuk mengubah persyaratan keterampilan minimum dalam pernyataan aturan perbandingan bernama "minSkill": `rules[minSkill].referenceValue`.

Wajib? Ya

`steps`

`waitTimeSeconds`

Lama waktu, dalam detik, untuk menunggu sebelum menerapkan nilai baru untuk elemen kumpulan aturan target.

Wajib? Ya

`value`

Nilai baru untuk elemen kumpulan aturan target.

FlexMatchjenis aturan

Aturan jarak batch

```
batchDistance
```

Batch aturan jarak mengukur perbedaan antara dua nilai atribut. Anda dapat menggunakan jenis aturan jarak batch dengan kecocokan besar dan kecil. Ada dua jenis aturan jarak batch:

- Bandingkan nilai atribut numerik. Misalnya, aturan jarak batch jenis ini mungkin mengharuskan semua pemain dalam pertandingan berada dalam dua tingkat keterampilan satu sama lain. Untuk jenis ini, tentukan jarak `batchAttribute` maksimum antara semua tiket.
- Bandingkan nilai atribut string. Misalnya, aturan jarak batch jenis ini mungkin mengharuskan semua pemain dalam pertandingan meminta mode permainan yang sama. Untuk jenis ini, tentukan `batchAttribute` nilai yang FlexMatch digunakan untuk membentuk batch.

Sifat aturan jarak batch

- **`batchAttribute`**- Nilai atribut pemain yang digunakan untuk membentuk batch.
- **`maxDistance`**- Nilai jarak maksimum untuk pertandingan yang sukses. Digunakan untuk membandingkan atribut numerik.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk nilai minimum (`min`), maksimum (`max`), dan rata-rata (`avg`) untuk pemain tiket. Defaultnya adalah `avg`.

Example

Contoh

```
{
  "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"
}
```

Aturan perbandingan

```
comparison
```

Aturan perbandingan membandingkan nilai atribut pemain dengan nilai lain. Ada dua jenis aturan perbandingan:

- Bandingkan dengan nilai referensi. Misalnya, aturan perbandingan jenis ini mungkin mengharuskan pemain yang cocok memiliki tingkat keterampilan tertentu atau lebih besar. Untuk jenis ini, tentukan atribut pemain, nilai referensi, dan operasi perbandingan.
- Bandingkan di seluruh pemain. Misalnya, aturan perbandingan jenis ini mungkin mengharuskan semua pemain dalam pertandingan menggunakan karakter yang berbeda. Untuk tipe ini, tentukan

atribut player dan operasi perbandingan equal (=) atau not-equal (!=). Jangan tentukan nilai referensi.

> **Note**
>
> Aturan jarak batch lebih efisien untuk membandingkan atribut pemain. Untuk mengurangi latensi perjodohan, gunakan aturan jarak batch bila memungkinkan.

Properti aturan perbandingan

- **`measurements`**- Nilai atribut pemain untuk membandingkan.
- **`referenceValue`**— Nilai untuk membandingkan pengukuran untuk untuk pertandingan prospektif.
- **`operation`**- Nilai yang menentukan bagaimana membandingkan pengukuran dengan nilai referensi. Operasi yang valid meliputi:<,<=,=,!=,>,>=.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk nilai minimum (`min`), maksimum (`max`), dan rata-rata (`avg`) untuk pemain tiket. Defaultnya adalah `avg`.

Aturan jarak

```
distance
```

Aturan jarak mengukur perbedaan antara dua nilai angka, seperti jarak antara tingkat keterampilan pemain. Misalnya, aturan jarak mungkin mengharuskan semua pemain memainkan permainan setidaknya selama 30 jam.

> **Note**
>
> Aturan jarak batch lebih efisien untuk membandingkan atribut pemain. Untuk mengurangi latensi perjodohan, gunakan aturan jarak batch bila memungkinkan.

Properti aturan jarak

- **`measurements`**- Nilai atribut pemain untuk mengukur jarak untuk. Ini harus berupa atribut dengan nilai numerik.
- **`referenceValue`**- Nilai numerik untuk mengukur jarak terhadap untuk pertandingan prospektif.
- **`minDistance`/`maxDistance`**— Nilai jarak minimum atau maksimum untuk pertandingan yang sukses.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk nilai minimum (`min`), maksimum (`max`), dan rata-rata (`avg`) untuk pemain tiket. Defaultnya adalah `avg`.

Aturan pengumpulan

```
collection
```

Aturan koleksi membandingkan sekelompok nilai atribut pemain dengan pemain lain dalam batch atau dengan nilai referensi. Koleksi dapat berisi nilai atribut untuk beberapa pemain, atribut pemain sebagai daftar string, atau keduanya. Misalnya, aturan pengumpulan mungkin melihat karakter yang dipilih pemain dalam tim. Aturan kemudian mungkin mengharuskan tim untuk memiliki setidaknya satu dari karakter tertentu.

Properti aturan koleksi

- **`measurements`**- Koleksi nilai atribut pemain untuk membandingkan. Nilai atribut harus daftar string.
- **`referenceValue`**- Nilai (atau koleksi nilai) yang digunakan untuk membandingkan pengukuran untuk pertandingan prospektif.
- **`operation`**— Nilai yang menentukan bagaimana membandingkan koleksi pengukuran. Operasi yang valid meliputi:
  - `intersection`- Operasi ini mengukur jumlah nilai yang sama di koleksi semua pemain. Untuk contoh aturan yang menggunakan operasi persimpangan, lihatReferensi: Gunakan pengurutan eksplisit untuk menemukan kecocokan terbaik.
  - `contains`- Operasi ini mengukur jumlah koleksi atribut pemain yang berisi nilai referensi yang ditentukan. Untuk contoh aturan yang menggunakan operasi contains, lihatReferensi: Tetapkan persyaratan tingkat tim dan batas latensi.
  - `reference_intersection_count`- Operasi ini mengukur jumlah item dalam koleksi atribut pemain yang cocok dengan item dalam koleksi nilai referensi. Anda dapat menggunakan operasi

ini untuk membandingkan beberapa atribut pemain yang berbeda. Untuk contoh aturan yang membandingkan beberapa koleksi atribut pemain, lihatReferensi: Temukan persimpangan di beberapa atribut pemain.

- **`minCount`/`maxCount`**— Nilai hitungan minimum atau maksimum untuk pertandingan yang berhasil.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Untuk nilai ini, Anda dapat menggunakan `union` untuk menggabungkan atribut pemain dari semua pemain di pesta. Atau, Anda dapat menggunakan `intersection` untuk menggunakan atribut pemain yang memiliki kesamaan partai. Defaultnya adalah `union`.

Aturan majemuk

```
compound
```

Aturan majemuk menggunakan pernyataan logis untuk membentuk pertandingan 40 atau lebih sedikit pemain. Anda dapat menggunakan beberapa aturan gabungan dalam satu set aturan. Saat menggunakan beberapa aturan majemuk, semua aturan majemuk harus benar untuk membentuk kecocokan.

Anda tidak dapat memperluas aturan gabungan menggunakan aturan ekspansi, tetapi Anda dapat memperluas aturan yang mendasari atau mendukung.

Sifat aturan majemuk

- **`statement`**- Logika yang digunakan untuk menggabungkan aturan individu untuk membentuk aturan majemuk. Aturan yang Anda tentukan di properti ini harus telah ditetapkan sebelumnya dalam kumpulan aturan Anda. Anda tidak dapat menggunakan `batchDistance` aturan dalam aturan gabungan.

  Properti ini mendukung operator logis berikut:

  - `and`- Ekspresi benar jika dua argumen yang diberikan benar.
  - `or`- Ekspresi benar jika salah satu dari dua argumen yang disediakan benar.
  - `not`- Membalikkan hasil argumen dalam ekspresi.
  - `xor`- Ekspresi benar jika hanya satu argumen yang benar.

Example Contoh

Contoh berikut cocok dengan pemain dari berbagai tingkat keterampilan berdasarkan mode permainan yang mereka pilih.

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

Latensi aturan

```
latency
```

Aturan latensi mengukur latensi pemain per lokasi. Aturan latensi mengabaikan lokasi mana pun dengan latensi lebih tinggi dari maksimum. Seorang pemain harus memiliki nilai latensi di bawah maksimum di setidaknya satu lokasi untuk aturan latensi untuk menerimanya. Anda dapat menggunakan jenis aturan ini dengan kecocokan besar dengan menentukan `maxLatency` properti.

Sifat aturan latensi

- **`maxLatency`**- Nilai latensi maksimum yang dapat diterima untuk suatu lokasi. Jika tiket tidak memiliki lokasi dengan latensi di bawah maksimum, maka tiket tidak cocok dengan aturan latensi.
- **`maxDistance`**- Nilai maksimum antara latensi setiap tiket dan nilai referensi jarak.
- **`distanceReference`**- Nilai latensi untuk membandingkan latensi tiket dengan. Tiket dalam jarak maksimum nilai referensi jarak menghasilkan pertandingan yang sukses. Opsi yang valid mencakup nilai latensi pemain minimum (`minavg`) dan rata-rata ().
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk nilai minimum (`min`), maksimum (`max`), dan rata-rata (`avg`) untuk pemain tiket. Defaultnya adalah `avg`.

> **Note**
>
> Antrian dapat menempatkan sesi permainan di Wilayah yang tidak cocok dengan aturan latensi. Untuk informasi selengkapnya tentang kebijakan latensi antrean, lihat Membuat kebijakan latensi pemain.

Aturan semacam absolut

```
absoluteSort
```

Aturan sortir absolut mengurutkan kumpulan tiket perjodohan berdasarkan atribut pemain tertentu dibandingkan dengan tiket pertama yang ditambahkan ke batch.

Sifat aturan semacam absolut

- **`sortDirection`**- Perintah untuk menyortir tiket perjodohan masuk Pilihan yang valid termasuk `ascending` dan`descending`.
- **`sortAttribute`**- Atribut pemain untuk mengurutkan tiket berdasarkan.
- **`mapKey`**- Opsi untuk mengurutkan atribut pemain jika itu adalah peta. Pilihan yang valid meliputi:
  - `minValue`- Kunci dengan nilai terendah adalah yang pertama.
  - `maxValue`- Kunci dengan nilai tertinggi adalah yang pertama.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk atribut pemain minimum (`min`), atribut pemain maksimum (`max`), dan rata-rata (`avg`) dari semua atribut pemain untuk pemain dalam partai. Defaultnya adalah `avg`.

Example

Contoh

Aturan contoh berikut mengurutkan pemain berdasarkan tingkat keterampilan dan rata-rata tingkat keterampilan pihak.

```
{
    "name":"AbsoluteSortExample",
```

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

Aturan semacam jarak

```
distanceSort
```

Aturan pengurutan jarak mengurutkan kumpulan tiket perjodohan berdasarkan jarak atribut pemain tertentu dari tiket pertama yang ditambahkan ke batch.

Properti aturan sortir jarak

- **`sortDirection`**- Arah untuk menyortir tiket perjodohan. Pilihan yang valid termasuk `ascending` dan`descending`.
- **`sortAttribute`**- Atribut pemain untuk mengurutkan tiket berdasarkan.
- **`mapKey`**- Opsi untuk mengurutkan atribut pemain jika itu adalah peta. Pilihan yang valid meliputi:
  - `minValue`- Untuk tiket pertama yang ditambahkan ke batch, temukan kunci dengan nilai terendah.
  - `maxValue`- Untuk tiket pertama yang ditambahkan ke batch, temukan kunci dengan nilai tertinggi.
- **`partyAggregation`**— Nilai yang menentukan bagaimana FlexMatch menangani tiket dengan beberapa pemain (pihak). Opsi yang valid termasuk nilai minimum (`min`), maksimum (`max`), dan rata-rata (`avg`) untuk pemain tiket. Defaultnya adalah `avg`.

FlexMatchekspresi properti

Ekspresi properti dapat digunakan untuk menentukan properti yang berhubungan dengan pencocokan tertentu. Mereka memungkinkan Anda untuk menggunakan perhitungan dan logika ketika mendefinisikan nilai properti. Ekspresi properti umumnya menghasilkan salah satu dari dua bentuk:

- Data pemain individu.
- Koleksi terhitung data pemain individu.

Ekspresi properti perjodohan umum

Ekspresi properti mengidentifikasi nilai tertentu untuk pemain, tim, atau pertandingan. Ekspresi parsial berikut menggambarkan cara mengidentifikasi tim dan pemain:

Tujuan	Input	Arti	Output
Untuk mengidentifikasi tim tertentu dalam sebuah pertandingan:	`teams[red]`	Tim merah	Tim
Untuk mengidentifikasi satu set tim tertentu dalam pertandingan:	`teams[red,blue]`	Tim Merah dan Tim Biru	Daftar<Team>
Untuk mengidentifikasi semua tim dalam sebuah pertandingan:	`teams[*]`	Semua tim	Daftar<Team>
Untuk mengidentifikasi pemain dalam tim tertentu:	`team[red] .players`	Pemain dalam tim Merah	Daftar<Player>
Untuk mengidentifikasi pemain dalam satu set tim tertentu dalam pertandingan:	`team[red, blue].players`	Pemain dalam pertandingan, dikelompokkan menurut tim	Daftar<List<Player>>
Untuk mengidentifikasi pemain dalam sebuah pertandingan:	`team[*].players`	Pemain dalam pertandingan, dikelompokkan menurut tim	Daftar<List<Player>>

Contoh ekspresi properti

Tabel berikut mengilustrasikan beberapa ekspresi properti yang dibangun pada contoh sebelumnya:

Ekspresi	Arti	Jenis Hasil
`teams[red].players[playerId]`	ID pemain dari semua pemain di tim merah	Daftar<string>
`teams[red].players.attributes[skill]`	Atribut "keterampilan" dari semua pemain di tim merah	Daftar<number>
`teams[red,blue].players.attributes[skill]`	Atribut “keterampilan” dari semua pemain di tim Merah dan tim Biru, dikelompokkan berdasarkan tim	Daftar<List<number>>
`teams[*].players.attributes[skill]`	Atribut "keterampilan" dari semua pemain dalam pertandingan, dikelompokkan berdasarkan tim	Daftar<List<number>>

Agregasi ekspresi properti

Ekspresi properti dapat digunakan untuk menggabungkan data tim dengan menggunakan fungsi atau kombinasi fungsi berikut ini:

Agregasi	Input	Arti	Output
`min`	Daftar<number>	Dapatkan minimum semua nomor dalam daftar.	jumlah
`max`	Daftar<number>	Dapatkan maksimal semua nomor dalam daftar.	jumlah

Agregasi	Input	Arti	Output
`avg`	Daftar<number>	Dapatkan rata-rata semua nomor dalam daftar.	jumlah
`median`	Daftar<number>	Dapatkan median dari semua nomor dalam daftar.	jumlah
`sum`	Daftar<number>	Dapatkan jumlah semua nomor dalam daftar.	jumlah
`count`	Daftar<?>	Dapatkan jumlah elemen dalam daftar.	jumlah
`stddev`	Daftar<number>	Dapatkan standar deviasi dari semua nomor dalam daftar.	jumlah
`flatten`	Daftar<List<?>>	Ubah koleksi daftar yang di-nest menjadi satu daftar yang berisi semua elemen.	Daftar<?>
`set_intersection`	Daftar<Daftar<string>>	Dapatkan daftar string yang ditemukan di semua daftar string dalam koleksi.	Daftar<string>
Semua di atas	Daftar<List<?>>	Semua operasi pada daftar bersarang beroperasi pada setiap subdaftar secara individual untuk menghasilkan daftar hasil.	Daftar<?>

Tabel berikut menggambarkan beberapa ekspresi properti yang valid yang menggunakan fungsi agregasi:

Ekspresi	Arti	Jenis Hasil
flatten(teams[*].players.attributes[skill])	Atribut "keterampilan" semua pemain dalam pertandingan (tidak dikelompokkan)	Daftar<number>
avg(teams[red].players.attributes[skill])	Keterampilan rata-rata pemain tim merah	jumlah
avg(teams[*].players.attributes[skill])	Keterampilan rata-rata masing-masing tim dalam pertandingan	Daftar<number>
avg(flatten(teams[*].players.attributes[skill]))	Tingkat keterampilan rata-rata semua pemain dalam pertandingan. Ekspresi ini mendapatkan daftar keterampilan pemain yang diratakan dan kemudian dirata-ratakan.	jumlah
count(teams[red].players)	Jumlah pemain di tim merah	jumlah
count (teams[*].players)	Jumlah pemain di setiap tim dalam pertandingan	Daftar<number>
max(avg(teams[*].players.attributes[skill]))	Tingkat keterampilan tim tertinggi dalam pertandingan	jumlah

FlexMatch acara perjodohan

Amazon GameLift FlexMatch memancarkan acara untuk setiap tiket perjodohan saat diproses. Anda dapat mempublikasikan acara ini ke SNS topik Amazon, seperti yang dijelaskan dalamSiapkan pemberitahuan FlexMatch acara. Peristiwa ini juga dipancarkan ke Amazon CloudWatch Events dalam waktu dekat dan dengan upaya terbaik.

Topik ini menjelaskan struktur FlexMatch peristiwa dan memberikan contoh untuk setiap jenis acara. Untuk informasi selengkapnya tentang status tiket perjodohan, lihat MatchmakingTicketdi Referensi Amazon. GameLift API

MatchmakingSearching

Tiket telah dimasukkan ke dalam matchmaking. Ini termasuk permintaan baru dan permintaan yang merupakan bagian dari match usulan yang gagal.

Sumber daya: ConfigurationArn

Detail: jenis, tiket, estimatedWaitMillis, gameSessionInfo

Contoh

```
{
  "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

Match potensial telah dibuat. Ini dikeluarkan untuk semua potensi match baru, terlepas dari apakah penerimaan diperlukan.

Sumber daya: ConfigurationArn

Detail: jenis, tiket,acceptanceTimeout,acceptanceRequired, ruleEvaluationMetrics, gameSessionInfo, matchId

Contoh

```
{
  "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

Pemain telah menerima match potensial. Acara ini berisi status penerimaan setiap pemain dalam match. Data yang hilang berarti AcceptMatch belum dipanggil untuk pemain itu.

Sumber daya: ConfigurationArn

Detail: jenis, tiket,matchId, gameSessionInfo

Contoh

```
{
  "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

Penerimaan match selesai karena penerimaan pemain, penolakan pemain, atau batas waktu penerimaan.

Sumber daya: ConfigurationArn

Detail: jenis, tiket, penerimaan,matchId, gameSessionInfo

Contoh

```
{
  "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

Matchmaking telah berhasil diselesaikan dan sesi game telah dibuat.

Sumber daya: ConfigurationArn

Detail: jenis, tiket,matchId, gameSessionInfo

Contoh

```
{
  "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

Tiket matchmaking telah gagal karena batas waktu.

Sumber daya: ConfigurationArn

Detail: jenis, tiket, ruleEvaluationMetrics, pesan,matchId, gameSessionInfo

Contoh


```
{
  "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

Tiket matchmaking telah dibatalkan.

Sumber daya: ConfigurationArn

Detail: jenis, tiket, ruleEvaluationMetrics, pesan,matchId, gameSessionInfo

Contoh

```
{
  "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

Tiket matchmaking mengalami kesalahan. Ini mungkin karena antrean sesi game tidak dapat diakses atau kesalahan internal.

Sumber daya: ConfigurationArn

Detail: jenis, tiket, ruleEvaluationMetrics, pesan,matchId, gameSessionInfo

Contoh

```
{
  "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"
  }
}
```

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.