



Guide d'hébergement

Amazon GameLift Servers



Amazon GameLift Servers: Guide d'hébergement

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que Amazon GameLift Servers ?	1
Ce que vous pouvez faire avec Amazon GameLift Servers	1
Comment utiliser Amazon GameLift Servers	2
Amazon GameLift Servers solutions	2
Options d'hébergement	3
FlexMatch pour le matchmaking	7
FleetIQ pour un hébergement Amazon EC2 autogéré	8
Realtime serveurs dotés d'une logique de serveur personnalisable	8
Architecture de solution d'hébergement géré	9
Composants de jeu avec hébergement	9
Ressources relatives aux solutions d'hébergement	11
Fonctionnement d'Amazon GameLift Servers	12
Hébergement de serveurs de jeux	12
Sessions de jeu en cours	13
Dimensionnement de la capacité d'une flotte	14
Surveillance des Amazon GameLift Servers	15
Utilisation d'autres AWS ressources	15
Comment fonctionnent les conteneurs	16
Composants du parc de conteneurs	16
Architectures communes	18
Fonctions de base	19
Comment les joueurs se connectent aux jeux	24
Points de service	26
AWS Emplacements pris en charge	26
Emplacements pour l'hébergement géré	31
Des emplacements pour Amazon GameLift Servers n'importe où	32
Emplacements pour Amazon GameLift Servers FlexMatch	32
Amazon GameLift Servers en Chine	33
Tarification	33
Générer des estimations de prix	34
Gérez les coûts d'hébergement des jeux	38
Premiers pas	40
Avant de commencer	40
Options d'intégration rapide	40

Options de développement personnalisées	41
Exemples Amazon GameLift Servers	41
Exemple de serveur de jeu personnalisé	42
Amazon GameLift ServersRealtimeexemple	42
Configurez un Compte AWS	43
Inscrivez-vous pour un Compte AWS	43
Création d'un utilisateur doté d'un accès administratif	44
Définissez les autorisations utilisateur pour Amazon GameLift Servers	45
Configurer l'accès programmatique pour les utilisateurs	46
Configurez l'accès programmatique à votre jeu	48
Exemples d'autorisations IAM	49
Configuration d'un rôle de service IAM	54
Obtenez des outils de développement	58
Pour les serveurs de jeux	59
Pour les services clients liés aux jeux	62
Pour la gestion des ressources	62
Pour Amazon GameLift ServersRealtime	63
Tutoriel : intégration rapide avec le wrapper Amazon GameLift Servers	64
Prérequis	65
Présentation	65
Étape 1 : Obtenir et créer le wrapper du serveur de jeu	66
Étape 2 : Préparez la version de votre serveur de jeu	66
Étape 3 : configurer le wrapper pour votre flotte	67
Étape 4 : Téléchargez le build du serveur de jeu	69
Étape 5 : Création de la EC2 flotte gérée	69
Étape 6 : créer une session de jeu et s'y connecter	71
Étape 7 : Gérez et surveillez votre flotte	72
Étape 8 : redimensionnez vos serveurs de jeu	72
Résoudre les problèmes courants	74
Étapes suivantes	75
Explorez avec un plugin de moteur de jeu	76
Flux de travail du plugin	77
Plugin pour Unreal Engine	77
Plug-in pour Unity (SDK de serveur 5.x)	112
Plug-in pour Unity (SDK de serveur 4.x)	135
Feuille de route EC2 pour le développement géré	160

Feuille de route pour le développement de conteneurs gérés	166
Feuille de route de développement pour Anywhere	174
Feuille de route de développement de l'hébergement hybride	181
Préparation de jeux pour Amazon GameLift Servers	190
Intégrez des jeux à des serveurs de jeux personnalisés	190
Amazon GameLift Servers interactions	191
Intégrer un serveur de jeu	195
Intégrer un client de jeu	206
Moteurs de jeu et Amazon GameLift Servers	213
Concevez votre service principal	239
Authentification de vos joueurs	239
Backend sans serveur	239
WebSocketbackend basé	241
Configuration pour le développement itératif	242
Créez un environnement de test basé sur le cloud	244
Configurer des tests locaux	248
Collaborez avec l'agent	253
Configuration des tests locaux (ancienne version)	255
Configuration Amazon GameLift Servers local	257
Tester un serveur de jeu	257
Tester un serveur et un client de jeu	260
Variations avec les produits locaux	263
Ajouter le FlexMatch matchmaking	264
Obtenez les données de votre flotte	264
Intégrer des jeux avec Amazon GameLift Servers Realtime	265
Que sont les Realtime serveurs ?	266
Gestion des sessions de jeu	266
Interaction client-serveur	266
Personnalisation d'un serveur	268
Déploiement et mise à jour	268
Intégration d'un client de jeu	269
Personnalisation d'un script Realtime	275
Déploiement du logiciel de serveur de jeu	281
Déployer une version de serveur personnalisée	281
Déploiement pour un hébergement géré	282
Déploiement pour un hébergement en tout lieu	282

Conditionner les fichiers de version de jeu	282
Ajouter un script d'installation de build	283
Création d'une version pour l'hébergement géré	286
Mettre à jour la version d'un serveur de jeu pour l'hébergement géré	291
Création d'une image de conteneur	292
Création d'une image de conteneur de serveur de jeu	293
Transférer une image de conteneur vers Amazon ECR	295
Déployer un Amazon GameLift Servers Realtime script	297
Fichiers de script du package	297
Télécharger des fichiers de script depuis un répertoire local	297
Charger des fichiers de script depuis Amazon S3	299
Mettre à jour les fichiers de script	301
Configuration d'une flotte d'hébergement	302
Caractéristiques de la flotte	302
Fonctionnement de la création d'une flotte	303
EC2 Flottes gérées	303
Flux de travail EC2 de création de flotte géré	305
Création d'une EC2 flotte gérée	306
Personnalisez vos flottes EC2 gérées	316
Mettre à jour la configuration d'une flotte	322
Déboguer les incidents de flotte	326
Connectez-vous à distance aux instances de flotte	330
Appairage de VPC	339
Flottes de conteneurs gérées	346
Création d'une flotte de conteneurs	348
Mettre à jour une flotte de conteneurs	353
Personnalisez une flotte de conteneurs	355
Création d'une définition de groupe de conteneurs	365
Mettre à jour la définition d'un groupe de conteneurs	370
Supprimer une définition de groupe de conteneurs	374
Des flottes partout	375
Workflow de création de flotte en tout lieu	376
Créez une flotte Anywhere	377
Abstraire une flotte avec un alias	387
Créer un alias	388
Modifier un alias	389

Gestion du placement des sessions de jeu avec des files d'attente	391
Caractéristiques des files d'attente	391
Créer une file d'attente	392
Personnaliser une file d'attente	396
Bonnes pratiques	397
Définition de l'étendue d'une file d'attente	398
Créez une file d'attente multi-sites	399
Prioriser le placement des sessions de jeu	400
Évaluer les métriques relatives aux files	406
Conception pour les instances Spot	407
Configurer la notification des événements	411
Configurer une rubrique SNS	411
Configuration d'une rubrique SNS avec chiffrement côté serveur	413
Configurez EventBridge	414
Tutoriel : Création d'une file d'attente avec des instances Spot	415
Étape 1 : Définissez l'étendue de votre file d'attente	416
Étape 2 : Création d'une infrastructure de flotte Spot	416
Étape 3 : Attribuer des alias à chaque flotte	417
Étape 4 : Création d'une file d'attente avec des destinations	417
Étape 5 : ajouter des limites de latence à la file d'attente	418
Récapitulatif	419
Élargir la capacité d'hébergement	420
Pour gérer la capacité du parc dans la console	421
Définissez les limites de capacité d'hébergement	421
Pour définir les limites de capacité	422
Définir manuellement la capacité de la flotte	423
Suspendre le dimensionnement automatique	423
Pour définir manuellement la capacité de la flotte	424
Adaptation automatique de la capacité du parc	425
Mise à l'échelle automatique basée sur les cibles	426
Mise à l'échelle automatique basée sur des règles	428
Échelle des flottes de conteneurs	433
Préparation du lancement du jeu	435
Se préparer	435
Préparez-vous pour les tests	436
Préparer le lancement	436

Planifier après le lancement	437
Gérer les ressources d'hébergement avec AWS CloudFormation	438
Bonnes pratiques	438
Utiliser des AWS CloudFormation piles	439
Des piles pour un seul emplacement	439
Des piles pour plusieurs régions	440
Mettre à jour les versions	441
Déployez les mises à jour de build	441
Déployer les mises à jour de build	442
Comment fonctionnent les annulations	443
Surveillance des Amazon GameLift Servers	444
Comparaison des métriques	444
Disponibilité des métriques par source	445
Choisir la bonne source de surveillance	447
Rubriques	448
Suivez l'hébergement des jeux sur la console	448
Tableau de bord d'hébergement	449
Constructions de serveurs de jeu	450
Amazon GameLift ServersscriptsRealtime	451
Flottes	452
Détails de la flotte	454
Sessions de jeu et sessions avec les joueurs	458
Alias	463
File d'attente des sessions de jeu	464
Moniteur avec CloudWatch	467
Dimensions des métriques	468
Métriques de la flotte	469
Mesures relatives à la flotte de conteneurs	482
Métriques de file d'attente	486
FlexMatch métriques	490
FleetIQ métriques	494
Journalisation des appels d'API	496
Amazon GameLift Servers informations dans CloudTrail	497
Compréhension Amazon GameLift Servers entrées du fichier journal	498
Journalisation des messages du serveur	500
Journalisation pour les serveurs personnalisés	500

Logging pour Amazon GameLift Servers Realtime	503
Sécurité	509
Protection des données	510
Chiffrement au repos	512
Chiffrement en transit	512
Confidentialité du trafic inter-réseau	513
Gestion des identités et des accès	513
Public ciblé	513
Authentification par des identités	514
Gestion des accès à l'aide de politiques	518
Fonctionnement de Amazon GameLift Servers avec IAM	521
Exemples de politiques basées sur l'identité	529
Résolution des problèmes	535
AWS politiques gérées	537
Journalisation et surveillance avec Amazon GameLift Servers	540
Validation de conformité	540
Résilience	542
Sécurité de l'infrastructure	543
Analyse de la configuration et des vulnérabilités	544
Bonnes pratiques de sécurité	545
Maintenir à jour les environnements d'exécution des flottes	545
Sécurisez vos configurations de ports	546
Ressources de sécurité supplémentaires	547
Guides de référence	548
API de service (AWS SDK)	548
Gérer les ressources Amazon GameLift Servers d'hébergement	549
Démarez des sessions de jeu et rejoignez des joueurs	553
SDK pour serveurs 5.x	554
Mises à jour 5.x du SDK pour serveurs	555
Migrer vers le SDK 5.x du serveur	556
SDK du serveur C++	558
SDK du serveur C#	609
SDK du serveur Go	652
SDK du serveur C++ (Unreal) : Actions	680
Server SDK 4 et versions antérieures	728
SDK de serveur pour C++ : actions	728

SDK de serveur pour C# : Actions	753
SDK de serveur pour Unreal : Actions	775
Amazon GameLift ServersréférenceRealtime	791
Référence de l'API client en temps réel (C#)	791
Realtimeréférence de script	806
Événements de placement lors des sessions de jeu	814
Syntaxe des événements de placement	814
PlacementFulfilled	815
PlacementCancelled	817
PlacementTimedOut	818
PlacementFailed	819
Versions d'AMI	820
Points de terminaison et quotas de service	822
Limites de l'API Amazon GameLift Servers	822
Référence des limites de l'API	822
Régulation au niveau des ressources	832
Limites de l'API du SDK Amazon GameLift Servers	832
Balises ping UDP	837
Comment fonctionnent les balises ping UDP	837
Cas d'utilisation courants des balises ping UDP	838
Obtenir les points de terminaison des balises	838
Implémentation de mesures de	839
Exemples de code	841
Notes de mise à jour et versions du SDK	849
Versions SDK	849
Notes de mise à jour	865
.....	cmxii

Qu'est-ce que Amazon GameLift Servers ?

Amazon GameLift Servers À utiliser pour déployer, exploiter et faire évoluer des serveurs dédiés à faible coût dans le cloud pour les jeux multijoueurs basés sur des sessions. Construit sur une infrastructure informatique AWS mondiale, il Amazon GameLift Servers permet de fournir des serveurs de jeu performants et fiables tout en adaptant dynamiquement votre utilisation des ressources pour répondre à la demande des joueurs du monde entier.

Ce que vous pouvez faire avec Amazon GameLift Servers

Amazon GameLift Servers prend en charge ces cas d'utilisation et bien d'autres encore :

- Hébergez vos propres serveurs de jeux multijoueurs personnalisés dans le cloud grâce Amazon GameLift Servers à l'EC2 hébergement géré.
- Gérez des ressources d'hébergement gérées à faible coût à l'aide des instances Spot [Amazon Elastic Compute Cloud \(Amazon EC2\)](#).
- Hébergez vos serveurs de jeu conteneurisés pour bénéficier de la flexibilité entre les plateformes et pour prendre en charge les migrations avec des conteneurs Amazon GameLift Servers gérés.
- Créez une solution d'hébergement hybride pour prendre en charge l'hébergement and/or multicloud sur site tout en gérant les sessions de jeu en un seul endroit avec Amazon GameLift Servers Anywhere.
- Créez un système de matchmaking robuste pour vos jeux multijoueurs avec Amazon GameLift Servers FlexMatch.
- Augmentez automatiquement votre capacité d'hébergement géré pour répondre aux besoins de votre jeu en fonction de l'utilisation réelle des joueurs.
- Gérez les ressources EC2 informatiques de votre Amazon pour les jeux en un seul endroit en utilisant Amazon GameLift Servers FleetIQ.
- Créez un environnement de test itératif pour les versions de votre serveur de jeu et de votre client avec Amazon GameLift Servers Anywhere et EC2.
- Pour les jeux qui ne nécessitent pas de serveur de jeu personnalisé, configurez une solution de serveur légère avec Amazon GameLift Servers Realtime.



Tip

[Commencez dès maintenant avec Amazon GameLift Servers](#)

Comment utiliser Amazon GameLift Servers

Utilisez ces outils pour gérer Amazon GameLift Servers.

AWS CLI

Utilisez le AWS Command Line Interface (AWS CLI) pour appeler le AWS SDK, y compris l'API de service pour Amazon GameLift Servers. Voir [Commencer à utiliser le AWS CLI](#) dans le guide de AWS Command Line Interface l'utilisateur.

Console Amazon GameLift Servers

Utilisez le [AWS Management Console for Amazon GameLift Servers](#) pour configurer les ressources, gérer les déploiements de vos serveurs de jeu et suivre les indicateurs de performance et d'utilisation. La Amazon GameLift Servers console est une alternative graphique à la gestion des ressources par programmation ou à l'aide du AWS CLI

Amazon GameLift Servers SDKs

Ils Amazon GameLift Servers SDKs contiennent les bibliothèques nécessaires pour établir la communication entre vos clients de jeu, vos serveurs de jeux, les services de jeux et le Amazon GameLift Servers service. Pour de plus amples informations, veuillez consulter [Obtenez des outils Amazon GameLift Servers de développement](#).

SDK client pour Amazon GameLift Servers Realtime

Le SDK client pour vous Amazon GameLift Servers Realtime permet de connecter votre client de jeu à un Realtime serveur fourni par Amazon GameLift Servers, de rejoindre des sessions de jeu et de rester en phase avec les autres joueurs. Téléchargez le [SDK](#) et découvrez comment effectuer des appels d'API avec l'[API Amazon GameLift Servers Realtime client \(C#\)](#).

Amazon GameLift Servers solutions

Amazon GameLift Servers propose une gamme de solutions pour les développeurs qui créent des jeux multijoueurs basés sur des sessions.

Solutions pour les développeurs de jeux

- [Amazon GameLift Serversoptions d'hébergement](#)
- [Amazon GameLift ServersFlexMatchpour le matchmaking](#)
- [Amazon GameLift ServersFleetIQpour un hébergement Amazon EC2 autogéré](#)
- [Amazon GameLift ServersRealtimeavec logique de serveur personnalisable](#)

Amazon GameLift Serversoptions d'hébergement

Lorsque vous travaillez avec vos serveurs de jeu Amazon GameLift Servers pour faire fonctionner vos serveurs de jeu, vous disposez de plusieurs options pour savoir où et comment ils sont hébergés. Que vous souhaitiez utiliser les ressources d'hébergement dont vous disposez déjà ou configurer un hébergement basé sur le cloud géré par Amazon GameLift Servers celui-ci, vous pouvez créer une expérience d'hébergement fluide pour vos joueurs.

[Géré EC2](#)

[Conteneurs gérés](#)

[Hébergement hybride](#)

[Hébergement n'importe où](#)

Géré EC2

Avec l' EC2 hébergement Amazon GameLift Servers géré, vous pouvez vous décharger de la majeure partie du travail de gestion de vos serveurs de jeux. Choisissez des ressources de calcul parmi une large sélection de types d' EC2instances Amazon. Intégrez vos projets de jeu et laissez-vous Amazon GameLift Servers guider par les détails. Pour en savoir plus sur l'hébergement géré, consultez[Fonctionnement d'Amazon GameLift Servers](#).

[Commencez à développer une solution d'hébergement Amazon GameLift Servers géré pour votre jeu.](#)

Fonctions principales

- Hébergez des jeux multijoueurs qui s'exécutent sur les systèmes d'exploitation Amazon Linux ou Windows Server.
- Offrez des expériences de jeu à faible latence à vos joueurs, où qu'ils se trouvent. Déployez des serveurs de jeu dans le monde entier dans toutes Régions AWS les Zones Amazon GameLift

Servers Locales compatibles. Pour obtenir la liste complète, consultez [Amazon GameLift Serverspoints de service](#).

- Utilisez le placement Amazon GameLift Servers intelligent des sessions de jeu afin que les joueurs bénéficient toujours de la meilleure expérience de jeu hébergée possible. Vous pouvez vous fier à la Amazon GameLift Servers prise de décision, ou vous pouvez personnaliser en fonction de critères de placement tels que le coût, le temps de latence des joueurs et les emplacements géographiques.
- Choisissez comment adapter vos ressources d'hébergement pour répondre à la demande des joueurs. Gérez la capacité manuellement ou configurez le dimensionnement automatique. Grâce à la mise à l'échelle automatique basée sur les objectifs, vous pouvez conserver une mémoire tampon de capacité inutilisée dimensionnée de manière dynamique, ce qui vous permet de contrôler les coûts tout en garantissant que les nouveaux joueurs peuvent accéder aux parties en un minimum d'attente.
- Laissez Amazon GameLift Servers déployer et gérer vos serveurs de jeu basés sur le cloud. Amazon GameLift Servers crée les ressources dont vous avez besoin, installe le logiciel de votre serveur de jeu et lance automatiquement les processus d'hébergement des sessions de jeu pour les joueurs. Configurez un suivi personnalisé de l'état de santé et Amazon GameLift Servers permettez de détecter et de résoudre les problèmes liés aux ressources peu performantes.
- Tirez parti des fonctionnalités de Amazon GameLift Servers surveillance pour évaluer les performances et l'utilisation. Vous pouvez suivre des indicateurs sur des facteurs tels que les performances matérielles, l'efficacité du placement des sessions de jeu et le cycle de vie des processus du serveur. Vous pouvez suivre les sessions de jeu actives et les sessions des joueurs pour observer l'utilisation au fil du temps. Vous pouvez également télécharger et stocker les journaux des sessions de jeu.
- Pour l'hébergement de production, automatisez la gestion et les déploiements des ressources d'hébergement de jeux à l'aide AWS CloudFormation de modèles pour Amazon GameLift Servers et de AWS Cloud Development Kit (AWS CDK). Tirez parti des outils et services d'intégration continue et de livraison continue (CI/CD) tels que. AWS CodePipeline

Conteneurs gérés

Amazon GameLift Servers fournit une solution complète d'hébergement cloud pour les serveurs de jeux conteneurisés. Avec les conteneurs Amazon GameLift Servers gérés, vous pouvez tirer parti des principaux avantages de l'utilisation des conteneurs, tels que la portabilité, l'agilité et la tolérance aux pannes. Les fonctionnalités suivantes sont disponibles avec les flottes de conteneurs gérées.

Commencez à développer une solution d'hébergement Amazon GameLift Servers géré pour votre serveur de jeu conteneurisé.

Fonctions principales

- Développez une architecture personnalisée avec des conteneurs légers pour exécuter le logiciel de votre serveur de jeu sur des ressources d'hébergement Amazon GameLift Servers basées sur Linux.
- Utilisez les outils Docker pour créer une image de conteneur basée sur Linux. Stockez les images à déployer dans un référentiel Amazon Elastic Container Registry (Amazon ECR).
- Offrez des expériences aux joueurs à faible latence en déployant les ressources de la flotte de conteneurs dans n'importe quelle zone Région AWS ou zone locale qui Amazon GameLift Servers les prend en charge. Consultez [Amazon GameLift Servers points de service](#).
- Gérez le cycle de vie de votre flotte à l'aide d'outils permettant de modéliser les versions des serveurs de jeu et de déployer des mises à jour de flotte
- Utilisez les fonctionnalités de placement des sessions de Amazon GameLift Servers jeu, notamment les files d'attente et le FlexMatch matchmaking, pour trouver les meilleurs matchs de session de jeu possibles pour vos joueurs.
- Testez l'architecture de votre serveur de jeu et de conteneur avec le Amazon GameLift Servers service à l'aide d'une flotte Anywhere. Testez votre jeu localement ou dans un environnement de test basé sur le cloud.
- Suivez les performances d'hébergement de jeux grâce à des indicateurs de performance spécifiques au conteneur. Surveillez l'état des ressources de votre flotte à l'aide de métriques matérielles.
- Gérez les ressources du parc de conteneurs à l'aide AWS CloudFormation de modèles pour Amazon GameLift Servers.

Hébergement hybride

Utilisez le Amazon GameLift Servers service avec une combinaison d'hébergement Amazon GameLift Servers géré et d'hébergement autogéré Anywhere. Une approche hybride vous permet de créer la solution dont vous avez besoin dès maintenant tout en vous préparant à ce que vous devrez être dans le futur. Les scénarios courants dans lesquels une solution hybride a du sens sont les suivants :

- Étendez votre solution d'hébergement au AWS Cloud. Complétez les capacités de votre solution d'hébergement existante (matériel sur site ou autre hébergement basé sur le cloud) en ajoutant un hébergement Amazon GameLift Servers géré. Avec l'hébergement géré, vous pouvez augmenter votre capacité d'hébergement ou ajouter une capacité « en rafale » pour augmenter rapidement et ne payer que pour les ressources lorsque vous en avez besoin. Vous pouvez également profiter de la présence mondiale du Amazon GameLift Servers service pour toucher un plus grand nombre de joueurs dans le monde entier et leur offrir l'expérience multijoueur à faible latence qu'ils attendent.
- Préparez-vous à la migration vers un hébergement basé sur le cloud. Si vous envisagez ou planifiez de migrer vers le AWS Cloud (au lieu de mettre à niveau votre propre matériel), une solution d'hébergement hybride est un moyen viable d'effectuer la transition aussi progressivement que nécessaire.
- Augmentez la latence pour les joueurs dans des lieux autres que ceux desservis par Amazon GameLift Servers. Si vous utilisez déjà un hébergement Amazon GameLift Servers géré, vous devrez peut-être aider les joueurs dans certaines situations. Par exemple, vous souhaitez peut-être atteindre des joueurs situés dans des endroits particulièrement éloignés ou réduire considérablement le temps de latence dans ces zones. Ajoutez des emplacements d'hébergement personnalisés et utilisez Amazon GameLift Servers Anywhere pour gérer ces sites ainsi que vos ressources d'hébergement gérées.

[Commencez à développer une solution d'hébergement Amazon GameLift Servers hybride pour votre jeu.](#)

Fonctions principales

- Utilisez les mêmes composants client et serveur de jeu avec des ressources d'hébergement gérées et autogérées. Offrez une expérience de jeu unifiée sur toutes les ressources d'hébergement.
- Utilisez les mêmes FlexMatch matchmakers pour placer des correspondances sur toutes les ressources d'hébergement.
- Gérez ensemble de manière centralisée vos ressources d'hébergement hybride pendant que vous les déployez dans le monde entier.
- À mesure que la demande des joueurs fluctue, gérez le chargement des sessions de jeu de manière fluide entre les ressources gérées et autogérées.
- Avec l'Amazon GameLift Serversagent, vous pouvez utiliser les mêmes outils pour gérer le cycle de vie des serveurs de jeu sur tous les types de ressources d'hébergement.

- Rassemblez les statistiques et les journaux du jeu et des joueurs sur toutes les ressources d'hébergement. Tirez parti des Amazon GameLift Servers fonctionnalités et autres AWS services pour combiner les données et développer des solutions d'observabilité cohérentes.

Hébergement n'importe où

Utilisez les flottes Amazon GameLift Servers Anywhere avec gestion des sessions de Amazon GameLift Servers jeu, y compris le matchmaking, pour héberger vos serveurs de jeu personnalisés où vous le souhaitez. Anywhere les flottes sont particulièrement utiles comme environnements de test pour le développement rapide et itératif de jeux. Configurez un Anywhere parc pour votre propre poste de travail local ou un ensemble de ressources d'hébergement basées sur le cloud. Pour l'hébergement de production, vous pouvez utiliser une approche hybride avec des Anywhere flottes pour votre matériel sur site complétées par des flottes Amazon GameLift Servers gérées.

Pour plus d'informations sur les tests avec Anywhere, consultez [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#). Pour plus d'informations sur la configuration d'une flotte Anywhere, consultez [Configuration d'un parc d'hébergement avec Amazon GameLift Servers](#).

[Commencez à développer une solution d'hébergement Amazon GameLift Servers Anywhere pour votre jeu.](#)

Fonctions principales

- Effectuez des tests rapides et itératifs pendant que vous développez vos jeux multijoueurs.
- Utilisez Amazon GameLift Servers des outils pour gérer les serveurs de jeux hébergés sur votre propre matériel.
- Profitez du matériel disponible le plus proche de vos joueurs, où que vous soyez.

Amazon GameLift Servers FlexMatch pour le matchmaking

FlexMatch À utiliser pour créer des ensembles de règles personnalisés afin de définir les matchs multijoueurs de votre jeu. FlexMatch utilise des ensembles de règles pour comparer les joueurs compatibles pour chaque match et offrir aux joueurs l'expérience multijoueur idéale.

Pour plus d'informations FlexMatch, voir [Qu'est-ce que c'est Amazon GameLift Servers FlexMatch ?](#)

Fonctions principales

- Équilibrez la vitesse de création des matchs et la qualité des matchs.

- Associez des joueurs ou des équipes en fonction de caractéristiques définies.
- Définissez des règles pour placer les joueurs dans les matchs en fonction de la latence.

Amazon GameLift Servers FleetIQ pour un hébergement Amazon EC2 autogéré

Utilisez des groupes de serveurs de FleetIQ jeu pour travailler directement avec vos ressources d'hébergement dans Amazon EC2 et Amazon EC2 Auto Scaling. Cela offre l'avantage d'Amazon GameLift Servers optimisations pour un hébergement de jeux peu coûteux et résilient. Cette solution s'adresse aux développeurs de jeux qui ont besoin de plus de flexibilité que ce que proposent les Amazon GameLift Servers solutions entièrement gérées.

Pour plus d'informations sur le FleetIQ fonctionnement avec Amazon EC2 et EC2 Auto Scaling pour l'hébergement de jeux, consultez le [guide du Amazon GameLift Servers FleetIQ développeur](#).

Fonctions principales

- Obtenez un équilibrage optimisé des instances Spot à l'aide de l'FleetIQ algorithme.
- Utilisez les fonctionnalités de routage des joueurs pour gérer efficacement les ressources de votre serveur de jeu et offrir une meilleure expérience aux joueurs lorsqu'ils rejoignent des parties.
- Adaptez automatiquement la capacité d'hébergement en fonction de l'utilisation des joueurs.
- Gérez directement EC2 les instances Amazon vous-même Compte AWS.
- Utilisez l'un des formats exécutables de serveur de jeu pris en charge, notamment Windows, Linux, les conteneurs et Kubernetes.

Amazon GameLift Servers Realtime avec logique de serveur personnalisable

Utilisez Realtime des serveurs pour héberger des jeux qui n'ont pas besoin d'un serveur de jeu personnalisé. Cette solution de serveur légère fournit des serveurs de jeu que vous pouvez configurer en fonction de votre jeu. Vous pouvez héberger Realtime des serveurs à l'aide d'une solution d'hébergement Amazon GameLift Servers géré.

Pour plus d'informations sur l'hébergement avec Amazon GameLift Servers Realtime, consultez [Intégrer des jeux avec Amazon GameLift Servers Realtime](#).

Fonctions principales

- Utilisez les fonctionnalités Amazon GameLift Servers de gestion, notamment le dimensionnement automatique, les files d'attente multi-sites et le placement des sessions de jeu.
- Utilisez les ressources Amazon GameLift Servers d'hébergement et choisissez le type de matériel AWS informatique pour vos flottes.
- Tirez parti d'un réseau complet pour l'interaction entre le client et le serveur du jeu.
- Bénéficiez des fonctionnalités intrinsèques du serveur de jeux avec une logique de serveur personnalisable.
- Effectuez des mises à jour en direct des configurations Realtime et de la logique du serveur.

Architecture de Amazon GameLift Servers solution gérée

Les diagrammes présentés dans cette rubrique décrivent la structure d'une solution Amazon GameLift Servers d'hébergement complète.

Composants de jeu avec hébergement

Le schéma suivant montre comment les composants clés d'une solution d'Amazon GameLift Servers hébergement géré fonctionnent ensemble pour faire fonctionner des serveurs de jeu dédiés et aider les joueurs à trouver des sessions de jeu hébergées et à s'y connecter. La solution d'hébergement que vous développerez pour votre jeu inclura la plupart ou la totalité de ces composants.

Les principaux composants de cette architecture sont les suivants :

Clients du jeu

Un client de jeu est un logiciel qui s'exécute sur l'appareil d'un joueur. Le joueur joue à votre jeu en rejoignant une session de jeu sur un serveur de jeu hébergé. Un client de jeu demande à rejoindre une session de jeu via un service principal, reçoit des informations de connexion pour une session de jeu et les utilise pour se connecter directement à la session de jeu. Pour de plus amples informations, veuillez consulter [Préparation de jeux pour Amazon GameLift Servers](#). Lors de la connexion à un Realtime serveur, un client de jeu A utilise le SDK client pour Amazon GameLift ServersRealtime.

Services de backend

Un service principal est un service personnalisé que vous créez pour gérer la communication avec le Amazon GameLift Servers service pour le compte d'un client de jeu. Vous pouvez également utiliser les services de backend pour des tâches spécifiques au jeu, telles que l'authentification et l'autorisation des joueurs, l'inventaire ou le contrôle des devises. Un service principal communique avec le Amazon GameLift Servers service à l'aide des opérations d'API du AWS SDK.

Un service principal fait des demandes pour obtenir des informations sur les sessions de jeu existantes et pour démarrer des sessions de jeu. Les demandes de nouvelles sessions de jeu définissent certaines caractéristiques, telles que le nombre maximum de joueurs. Ces demandes invitent Amazon GameLift Servers à démarrer le processus de placement de session de jeu. Lorsqu'une session de jeu est prête à accepter des joueurs, le service principal récupère les informations de connexion et les fournit au client du jeu.

Services externes

Votre jeu peut s'appuyer sur des services externes, tels que la validation d'un abonnement. Un service externe peut transmettre des informations à vos serveurs de jeu via un service principal et Amazon GameLift Servers.

Serveurs de jeux

Un serveur de jeu est le logiciel serveur de votre jeu qui fonctionne sur un ensemble de ressources d'hébergement. Vous téléchargez le logiciel de votre serveur de jeu Amazon GameLift Servers, qui le déploie sur les ressources d'hébergement et lance les processus du serveur. Chaque processus du serveur de jeu se connecte au Amazon GameLift Servers service pour signaler qu'il est prêt à héberger des sessions de jeu. Il interagit avec le service pour démarrer des sessions de jeu, valider les joueurs nouvellement connectés et signaler l'état des sessions de jeu et des connexions des joueurs.

Les serveurs de jeu personnalisés communiquent avec eux Amazon GameLift Servers à l'aide du SDK du serveur pour Amazon GameLift Servers. Pour plus d'informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Realtime les serveurs sont des serveurs de jeux fournis par Amazon GameLift Servers. Vous pouvez personnaliser la logique du serveur en fournissant un script personnalisé. Pour de plus amples informations, veuillez consulter [Intégrer des jeux avec Amazon GameLift Servers Realtime](#).

Outils de gestion des hôtes

Lors de la configuration et de la gestion des ressources d'hébergement, les propriétaires de jeux utilisent des outils de gestion de l'hébergement pour gérer les versions ou les scripts des serveurs

de jeu, les flottes, le matchmaking et les files d'attente. L'Amazon GameLift Servers outil défini dans le AWS SDK et dans la console vous permet de gérer vos ressources d'hébergement de différentes manières. Vous pouvez accéder à distance à n'importe quel serveur de jeu individuel pour résoudre les problèmes.

Ressources relatives aux solutions d'hébergement

Le schéma suivant illustre Amazon GameLift Servers les ressources qui constituent une solution d'hébergement géré. Fournissez une version de serveur personnalisée ou un Amazon GameLift Servers Realtime script, déployez une flotte d'ordinateurs pour héberger des serveurs de jeu, puis configurez une file d'attente de sessions de jeu pour trouver les ressources d'hébergement disponibles et démarrer de nouvelles sessions de jeu. Pour les jeux qui utilisent le FlexMatch matchmaking, ajoutez une configuration de matchmaking et un ensemble de règles de matchmaking pour générer des matchs entre joueurs.

Code du serveur de jeu

- **Build** : votre logiciel de serveur de jeu personnalisé qui s'exécute sur Amazon GameLift Servers et héberge des sessions de jeu pour vos joueurs. Une version de jeu représente l'ensemble des fichiers qui exécutent votre serveur de jeu sur un système d'exploitation spécifique et auxquels vous devez procéder à l'intégration Amazon GameLift Servers. Téléchargez les fichiers de build du jeu Amazon GameLift Servers dans l' Région AWS endroit où vous prévoyez de configurer des flottes. Pour de plus amples informations, veuillez consulter [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#).
- **Script** — Votre configuration et votre logique de jeu personnalisée à utiliser avec Amazon GameLift ServersRealtime. Configurez Amazon GameLift Servers Realtime pour vos clients de jeu en créant un script à l'aide JavaScript de et ajoutez-y une logique de jeu personnalisée pour héberger des sessions de jeu pour vos joueurs. Pour de plus amples informations, veuillez consulter [Déployez un script pour Amazon GameLift ServersRealtime](#).

Parc

Ensemble de ressources informatiques qui font fonctionner vos serveurs de jeu et hébergent des sessions de jeu pour vos joueurs. Pour plus d'informations sur les endroits où vous pouvez déployer des flottes, consultez [Amazon GameLift Serverspoints de service](#). Pour plus d'informations sur la création de flottes, consultez [Configuration d'un parc d'hébergement avec Amazon GameLift Servers](#).

Alias

Identifiant abstrait d'une flotte que vous pouvez utiliser pour modifier la flotte à laquelle vos joueurs sont connectés à tout moment. Pour de plus amples informations, veuillez consulter [Créez un Amazon GameLift Servers alias](#).

File d'attente des sessions de jeu

Un mécanisme de placement de sessions de jeu qui reçoit les demandes de nouvelles sessions de jeu et recherche les serveurs de jeu disponibles pour héberger les nouvelles sessions. Pour plus d'informations sur les files d'attente des sessions de jeu, consultez [Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente](#).

Fonctionnement d'Amazon GameLift Servers

Cette rubrique décrit comment Amazon GameLift Servers gérer l'hébergement dédié de vos serveurs de jeux multijoueurs et comment les mettre à la disposition des joueurs. Il décrit le fonctionnement des fonctionnalités de base.

Hébergement de serveurs de jeux

Avec Amazon GameLift Servers, vous pouvez héberger vos serveurs de jeu de différentes manières : gérés Amazon GameLift Servers et Amazon GameLift Servers n'importe où. Amazon GameLift Servers FleetIQ Pour plus d'informations Amazon GameLift Servers FleetIQ, voir [Qu'est-ce que c'est Amazon GameLift Servers FleetIQ ?](#)

Vous pouvez concevoir une flotte en fonction des besoins de votre jeu. Pour plus d'informations sur la conception d'une flotte, consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).

Amazon GameLift Servers géré

Avec Managed Amazon GameLift Servers, vous pouvez héberger vos serveurs de jeu sur des ressources informatiques Amazon GameLift Servers virtuelles, appelées instances. Configurez vos ressources d'hébergement en créant une flotte d'instances et en les déployant pour faire fonctionner vos serveurs de jeu.

Amazon GameLift Servers N'importe où

Avec Amazon GameLift Servers Anywhere, vous pouvez héberger vos serveurs de jeu sur des ordinateurs que vous gérez. Configurez vos ressources d'hébergement en créant une flotte Anywhere qui fait référence à vos ressources informatiques.

Alias de flotte

Un alias est une désignation que vous pouvez transférer entre flottes, ce qui en fait un moyen pratique d'avoir une localisation générique de flotte. Vous pouvez utiliser un alias pour faire passer les clients du jeu d'une flotte à une autre sans changer de client de jeu. Vous pouvez également créer un alias de terminal pointant vers du contenu.

Sessions de jeu en cours

Une fois que vous avez déployé la version de votre serveur de jeu sur une flotte et que vous avez Amazon GameLift Servers lancé les processus du serveur de jeu sur chaque instance, la flotte peut héberger des sessions de jeu. Amazon GameLift Servers démarre de nouvelles sessions de jeu lorsque le service client de votre jeu envoie une demande de placement au service principal ou à Amazon GameLift Servers.

Placement des sessions de jeu et FleetIQ algorithm

Les files d'attente utilisent l'FleetIQ algorithm pour sélectionner un serveur de jeu disponible pour héberger une nouvelle session de jeu. L'élément clé pour le placement des sessions de jeu est la file d'attente des sessions de Amazon GameLift Servers jeu. Vous attribuez à une file d'attente de session de jeu une liste de flottes, qui détermine où la file d'attente peut placer les sessions de jeu. Pour plus d'informations sur les files d'attente des sessions de jeu et sur la manière de les concevoir pour votre jeu, consultez [Personnaliser une file d'attente de session de jeu](#).

Optimisation du placement des sessions de jeu avec des balises ping UDP

À l'aide de balises ping Amazon GameLift Servers UDP, vous pouvez calculer la latence aller-retour des paquets UDP entre les joueurs et les serveurs de jeu situés à différents endroits afin de choisir l'emplacement optimal pour une session de jeu. Pour plus d'informations sur les balises ping UDP et sur la façon dont vous pouvez les utiliser pour mesurer la latence, consultez. [Balises ping UDP](#)

- Pour les jeux qui utilisent des files d'attente de session de jeu pour le placement, la demande de placement peut inclure des données de latence, que la file d'attente utilise automatiquement pour hiérarchiser les emplacements de placement. Vous pouvez personnaliser davantage la priorisation et définir des politiques, notamment des limites de latence. Consultez [Prioriser le placement des sessions de jeu](#).
- Pour les jeux qui n'utilisent pas de files d'attente pour les sessions de jeu, mais dont les flottes comportent plusieurs emplacements, vous pouvez évaluer les données de latence et choisir le meilleur emplacement disponible avant de faire une demande de session de jeu à Amazon

GameLift Servers Consultez les sections Obtenir et créer des sessions de jeu dans [Ajoutez Amazon GameLift Servers à votre client de jeu](#).

- Si vous utilisez FlexMatch pour le matchmaking, vous pouvez définir des règles de match pour utiliser les données de latence. Consultez la section [Demander le matchmaking pour les joueurs](#) et la section sur les [règles de latence](#) sur les types de FlexMatch règles dans le guide du Amazon GameLift Servers FlexMatch développeur.

Connexions entre les joueurs et les jeux

Dans le cadre du processus de placement de session de jeu, la file d'attente ou la session de jeu invite le serveur de jeu sélectionné à démarrer une nouvelle session de jeu. Le serveur de jeu répond à l'invite et indique Amazon GameLift Servers quand il est prêt à accepter les connexions des joueurs. Amazon GameLift Servers fournit ensuite les informations de connexion au service principal ou au service client du jeu. Vos clients de jeu utilisent ces informations pour se connecter directement à la session de jeu et commencer à jouer.

Dimensionnement de la capacité d'une flotte

Lorsqu'une flotte est active et prête à accueillir des sessions de jeu, vous pouvez ajuster la capacité de votre flotte pour répondre à la demande des joueurs. Nous vous recommandons de trouver un équilibre entre le fait que tous les nouveaux joueurs trouvent un jeu rapidement et les dépenses excessives consacrées à des ressources inutilisées.

Amazon GameLift Servers fournit un outil de mise à l'échelle automatique très efficace, ou vous pouvez définir manuellement la capacité du parc. Pour de plus amples informations, veuillez consulter [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).

Auto scaling (Mise à l'échelle automatique)

Amazon GameLift Servers propose deux méthodes de mise à l'échelle automatique :

- [Mise à l'échelle automatique basée sur les cibles](#)
- [Mise à l'échelle automatique avec des politiques basées sur des règles](#)

Fonctionnalités de mise à l'échelle supplémentaires

- Protection des sessions de jeu — Empêchez Amazon GameLift Servers de mettre fin aux sessions de jeu qui accueillent des joueurs actifs lors d'un événement de réduction.

- Limites de dimensionnement : contrôlez l'utilisation globale des instances en définissant des limites minimales et maximales quant au nombre d'instances d'un parc.
- Suspension du dimensionnement automatique : suspendez le dimensionnement automatique au niveau de l'emplacement du parc sans modifier ni supprimer vos politiques de dimensionnement automatique.
- Mesures de mise à l'échelle : suivez l'historique de capacité et les événements de mise à l'échelle d'une flotte.

Surveillance des Amazon GameLift Servers

Lorsque vos flottes sont opérationnelles, Amazon GameLift Servers collecte diverses informations pour vous aider à surveiller les performances de vos serveurs de jeu déployés. Vous pouvez utiliser ces informations pour optimiser votre utilisation des ressources, résoudre les problèmes et mieux comprendre comment les joueurs sont actifs dans vos jeux. Amazon GameLift Servers collecte les informations suivantes :

- Flotte, emplacement, session de jeu et détails de la session du joueur
- Métriques d'utilisation
- État des processus du serveur
- Journaux des sessions de jeu

Pour plus d'informations sur la surveillance dans Amazon GameLift Servers, consultez [Surveillance des Amazon GameLift Servers](#).

Utilisation d'autres AWS ressources

Vos serveurs de jeu et vos applications peuvent communiquer avec d'autres AWS ressources. Par exemple, vous pouvez utiliser un ensemble de services Web pour l'authentification des joueurs ou les réseaux sociaux. Pour que vos serveurs de jeu puissent accéder aux AWS ressources que vous Compte AWS gérez, autorisez explicitement Amazon GameLift Servers l'accès à vos AWS ressources.

Amazon GameLift Servers fournit plusieurs options pour la gestion de ce type d'accès. Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

Comment fonctionnent les conteneurs dans Amazon GameLift Servers

Amazon GameLift Servers les flottes de conteneurs sont conçues pour vous apporter de la flexibilité dans la manière dont vous déployez et dimensionnez vos applications conteneurisées. Il utilise Amazon Elastic Container Service (Amazon ECS) pour gérer le déploiement et l'exécution des tâches pour Amazon GameLift Servers vos flottes. Cette rubrique décrit les éléments structurels de base pour faire fonctionner des conteneurs sur un parc Amazon GameLift Servers géré, illustre les architectures courantes et décrit certains concepts de base.

 Accélérez l'intégration grâce à ces outils pour les conteneurs gérés :

- Le [kit de démarrage pour conteneurs](#) rationalise l'intégration et la configuration de la flotte. Il ajoute des fonctionnalités essentielles de gestion des sessions de jeu à votre serveur de jeu et utilise des modèles préconfigurés pour créer une flotte de conteneurs et un pipeline de déploiement automatisé pour votre serveur de jeu. Après le déploiement, utilisez la Amazon GameLift Servers console et les outils d'API pour surveiller les performances de la flotte, gérer les sessions de jeu et analyser les indicateurs.
- Pour les développeurs Unreal Engine ou Unity, utilisez les [Amazon GameLift Serversplugins](#) pour intégrer votre serveur de jeu et créer une flotte de conteneurs à partir de l'environnement de développement de votre moteur de jeu. Les flux de travail guidés du plugin vous aident à créer une solution simple et rapide avec un hébergement basé sur le cloud à l'aide de conteneurs gérés. Puis, sur cette base, créez une solution d'hébergement personnalisée pour votre jeu.

Composants du parc de conteneurs

Flotte

Une flotte de conteneurs est un ensemble d' EC2 instances Amazon destinées à héberger vos serveurs de jeux conteneurisés. Ces instances sont gérées Amazon GameLift Servers en votre nom. Lorsque vous créez une flotte, vous configurez la manière dont l'architecture du conteneur est déployée avec le logiciel de votre serveur de jeu sur chaque instance de flotte. Vous pouvez créer un parc de conteneurs avec des instances situées dans un ou plusieurs emplacements géographiques. Vous pouvez utiliser des outils de Amazon GameLift Servers dimensionnement

pour adapter automatiquement la capacité d'une flotte de conteneurs afin d'accueillir des sessions de jeu et des joueurs.

Instance

Une EC2 instance Amazon est le serveur virtuel qui fournit la capacité de calcul pour l'hébergement de jeux. Avec Amazon GameLift Servers, vous pouvez choisir parmi une gamme de types d'instances. Chaque type d'instance offre une combinaison différente de CPU, de mémoire, de stockage et de capacité réseau.

Lorsque vous créez un parc de conteneurs, Amazon GameLift Servers déployez vos conteneurs en fonction du type d'instance que vous avez choisi et de la configuration de votre flotte. Chaque instance de flotte déployée est identique et exécute le logiciel de votre serveur de jeu conteneurisé de la même manière. Le nombre d'instances d'une flotte détermine la taille de la flotte et sa capacité d'hébergement de jeux.

Groupe de conteneurs

Amazon GameLift Servers utilise le concept de groupe de conteneurs pour décrire et gérer un ensemble de conteneurs. Un groupe de conteneurs est similaire à une « tâche » ou à un « pod » de conteneurs. Au sein de chaque groupe de conteneurs, vous pouvez définir le comportement des conteneurs, configurer les dépendances et partager les ressources de processeur et de mémoire disponibles.

Chaque instance de flotte peut comporter les types de groupes de conteneurs suivants :

- Un groupe de conteneurs de serveurs de jeux gère les conteneurs qui exécutent votre application de serveur de jeu et les logiciels associés. Une flotte de conteneurs doit disposer d'un groupe de conteneurs de ce type pour accueillir les sessions de jeu et les joueurs. Un groupe de conteneurs de serveurs de jeu peut être répliqué sur une instance de flotte. Le nombre de répliques de groupes de serveurs de jeu par instance de flotte dépend des exigences de calcul de votre logiciel et des ressources de calcul disponibles sur l'instance.
- Un groupe de conteneurs par instance, facultatif, vous permet d'exécuter des logiciels supplémentaires sur chaque instance de flotte. Ils sont utiles pour exécuter des services d'arrière-plan ou des programmes utilitaires, par exemple pour la surveillance. Le logiciel de votre serveur de jeu ne dépend pas directement des processus d'un groupe par instance. Une seule copie d'un groupe de conteneurs par instance est déployée sur chaque instance de flotte.

Chaque groupe de conteneurs d'une flotte de conteneurs possède un conteneur désigné « essentiel ». Un conteneur essentiel conditionne le cycle de vie d'un groupe de conteneurs. Si le conteneur essentiel tombe en panne, l'ensemble du groupe de conteneurs redémarre.

Conteneur

Le conteneur est l'élément le plus fondamental d'une architecture basée sur un conteneur. Il inclut une image de conteneur avec les exécutables du logiciel et les fichiers dépendants. Définissez un conteneur pour configurer le mode d'exécution et d'interaction du logiciel. Amazon GameLift Servers

Amazon GameLift Servers définit deux types de conteneurs :

- Un conteneur de serveur de jeu inclut tout ce dont vous avez besoin pour exécuter les processus de votre serveur de jeu et héberger des sessions de jeu pour les joueurs. Cela inclut la version de votre serveur de jeu et les logiciels qui en dépendent. Définissez un conteneur de serveur de jeu pour le groupe de conteneurs de serveurs de jeu d'une flotte. Le conteneur du serveur de jeu est automatiquement considéré comme essentiel pour le groupe de conteneurs.
- Un conteneur de support exécute des logiciels supplémentaires pour prendre en charge votre serveur de jeu. Il est similaire au concept d'un conteneur « sidecar ». Il vous donne la possibilité d'exécuter et de faire évoluer les logiciels de support parallèlement à vos serveurs de jeu, mais de les gérer en tant que conteneurs distincts. Dans un groupe de conteneurs de serveurs de jeu, vous pouvez définir zéro ou plusieurs conteneurs de support. Dans un groupe de conteneurs par instance, tous les conteneurs sont des conteneurs de support. Tout contenant de support peut être désigné comme essentiel.

Calcul

Un calcul représente une copie d'un groupe de conteneurs de serveurs de jeu sur une instance de flotte.

Architectures communes

Le schéma suivant illustre la structure de flotte de conteneurs la plus simple. Dans cette structure, chaque instance de la flotte conserve une copie du groupe de conteneurs du serveur de jeu. Le groupe de conteneurs possède un conteneur de serveur de jeu unique qui exécute un processus de serveur de jeu. Dans cet exemple, le parc de conteneurs est configuré pour placer une copie du groupe de conteneurs du serveur de jeu par instance. Avec cette architecture, chaque instance exécute un processus de serveur de jeu.

Ce deuxième schéma illustre une architecture de flotte de conteneurs plus complexe. Dans cette structure, la flotte possède à la fois un groupe de conteneurs de serveurs de jeu et un groupe de conteneurs par instance. Le groupe de conteneurs de serveurs de jeu possède des conteneurs

distincts pour le processus du serveur de jeu et un processus de support. La flotte est configurée pour placer trois copies du groupe de conteneurs du serveur de jeu sur chaque instance de flotte. Le groupe de conteneurs par instance n'est jamais répliqué. Dans cet exemple, le parc de conteneurs est configuré pour placer trois copies du groupe de conteneurs du serveur de jeu par instance. Avec cette architecture, chaque instance exécute trois processus de serveur de jeu.

Fonctions de base

Cette section résume comment Amazon GameLift Servers implémenter certains concepts de base des conteneurs. Pour obtenir des instructions sur la manière de travailler avec des flottes de conteneurs, consultez les rubriques pertinentes de ce guide.

Mises à jour actives du parc

Les conteneurs gérés fournissent un support avancé pour vous aider à gérer le cycle de vie de votre logiciel hébergé et de votre architecture de conteneur. Vous pouvez mettre à jour les définitions de vos conteneurs, y compris les images des conteneurs, et apporter des modifications à vos flottes existantes. Cette fonctionnalité accélère et facilite l'itération des modifications apportées à vos conteneurs pendant le développement. Il fournit également des fonctionnalités qui vous aident à créer, déployer et suivre les mises à jour de vos versions logicielles au fil du temps. Ces fonctionnalités incluent :

- Gérez les mises à jour et le versionnement des définitions de groupes de conteneurs. Vous pouvez mettre à jour presque toutes les propriétés d'une définition de groupe de conteneurs, y compris les images de conteneurs et les paramètres de configuration. Chaque fois que vous mettez à jour un conteneur, attribue Amazon GameLift Servers automatiquement un numéro de version à la mise à jour et conserve toutes les versions par défaut. Vous pouvez accéder à n'importe quelle version spécifique, et vous pouvez supprimer des versions comme vous le souhaitez. Lorsque vous créez un parc de conteneurs, vous pouvez spécifier la définition du groupe de conteneurs et la version à déployer.
- Mettez à jour les flottes de conteneurs existantes avec de nouvelles définitions de groupes de conteneurs et de nouveaux paramètres de configuration. Vous pouvez déployer des mises à jour de conteneurs sur des flottes déjà déployées sur des instances de flotte. Vous pouvez suivre l'état des déploiements de mises à jour sur chaque site du parc à l'aide du AWS SDK AWS Management Console ou de la CLI.
- Configurez la manière dont vous souhaitez que les mises à jour de flotte soient déployées sur une flotte active.

- **Protection des sessions de jeu.** Choisissez de protéger les instances de flotte avec des sessions de jeu actives jusqu'à la fin des sessions de jeu (déploiement sécurisé). Vous pouvez également choisir de remplacer les instances de flotte quelle que soit l'activité de la session de jeu (déploiement non sécurisé). Utilisez des déploiements non sécurisés pendant les phases de développement et de test afin de réduire le temps de déploiement.
- **Pourcentage de santé minimum.** Spécifiez le pourcentage de tâches saines que vous souhaitez conserver pendant le déploiement. Cette fonctionnalité vous permet de décider du nombre d'instances de flotte concernées lors d'un déploiement. Une valeur faible donne la priorité à la vitesse de déploiement, tandis qu'une valeur élevée garantit que la disponibilité du serveur de jeu reste élevée tout au long du déploiement.
- **Stratégie en cas d'échec du déploiement.** Décidez des mesures à prendre en cas d'échec d'un déploiement. Un échec de déploiement signifie que certains conteneurs mis à jour ont échoué aux vérifications d'état et sont considérés comme défectueux. Vous pouvez configurer des déploiements pour rétablir automatiquement l'état précédemment déployé de toutes les instances de flotte. Vous pouvez également choisir de conserver certaines des instances de flotte altérées pour les utiliser dans le cadre du débogage.

La possibilité de mettre à jour les flottes actives est très utile lorsque vous souhaitez déployer une mise à jour du logiciel de votre serveur de jeu. Une fois que vous avez créé une nouvelle image de conteneur pour votre serveur de jeu, son déploiement s'effectue en deux étapes : tout d'abord, mettez à jour la définition du groupe de conteneurs avec la nouvelle image, puis mettez à jour le parc de conteneurs. Amazon GameLift Servers gère toutes les autres tâches selon les besoins.

Emballage de conteneurs

Lorsque vous développez votre structure de conteneurs en vue d'un déploiement dans une flotte de conteneurs, un objectif commun est d'optimiser l'utilisation de la puissance de calcul disponible. Pour atteindre cet objectif, vous devez intégrer autant de groupes de conteneurs de serveurs de jeu que possible dans chaque instance de flotte.

Amazon GameLift Servers vous y aide en calculant le nombre maximal de groupes de conteneurs de serveurs de jeu par instance, sur la base des informations suivantes :

- Le type d'instance du parc, ses vCPU et ses ressources de mémoire.
- Les exigences en matière de vCPU et de mémoire pour tous les conteneurs du groupe de conteneurs du serveur de jeu.

Les exigences en matière de vCPU et de mémoire pour tous les conteneurs d'un groupe de conteneurs par instance, le cas échéant.

Lorsque vous créez une flotte de conteneurs, vous pouvez utiliser le maximum calculé ou vous pouvez spécifier le nombre souhaité. Il est recommandé de faire des essais avec le logiciel de votre serveur de jeu conteneurisé afin de déterminer les ressources nécessaires pour des performances optimales du serveur de jeu.

Mise à l'échelle de la capacité

La capacité de la flotte mesure le nombre de sessions de jeu qu'une flotte peut héberger simultanément. Vous pouvez également mesurer la capacité en fonction du nombre de joueurs simultanés qu'une flotte peut supporter. Pour augmenter ou diminuer la capacité d'hébergement d'une flotte, vous ajoutez ou supprimez des instances de flotte.

Une flotte de conteneurs est configurée pour exécuter un nombre spécifique de processus de serveur de jeu simultanés sur chaque instance de flotte. (Vous pouvez le calculer en fonction (1) des groupes de conteneurs du serveur de jeu par instance et (2) du nombre de processus du serveur de jeu exécutés dans chaque groupe de conteneurs.) Le nombre de serveurs de jeu actuels par instance vous indique quel est l'impact de l'ajout ou de la suppression de chaque instance de flotte. Par exemple, si votre flotte de conteneurs exécute un processus de serveur de jeu dans chaque groupe de conteneurs de serveurs de jeu, et que chaque instance de flotte contient 100 groupes de conteneurs de serveurs de jeu, vous augmentez ou diminuez la capacité de votre flotte à héberger des sessions de jeu simultanées par incréments de 100. Si chaque session de jeu comporte 10 emplacements pour joueurs, vous augmentez ou diminuez la capacité de votre flotte à accueillir des joueurs par incréments de 1 000.

Avec les flottes de conteneurs, vous pouvez utiliser n'importe laquelle des méthodes de mise à l'échelle des capacités fournies par Amazon GameLift Servers. Il s'agit des licences suivantes :

- Définissez la capacité du parc manuellement en définissant le nombre d'instances de flotte souhaité.
- Configurez le dimensionnement automatique en ciblant la mémoire tampon souhaitée parmi les instances disponibles (suivi des cibles). Cette méthode maintient automatiquement un certain nombre de ressources d'hébergement inactives afin que les joueurs entrants puissent accéder rapidement aux jeux. Au fur et à mesure que la demande des joueurs augmente ou diminue, la taille de cette zone tampon est continuellement ajustée.

- Configurez le dimensionnement automatique avec des règles de dimensionnement personnalisées (fonctionnalité avancée). Cette méthode vous permet d'évoluer en fonction des indicateurs de flotte que vous choisissez.

client/server Connexions au jeu

Avec les flottes Amazon GameLift Servers gérées, les clients de jeu se connectent directement à vos serveurs de jeu hébergés dans le cloud. Lorsqu'un client de jeu demande à rejoindre un jeu, Amazon GameLift Servers trouve une session de jeu et fournit des informations de connexion (IP et port) au client du jeu. Vous pouvez contrôler l'accès externe aux instances de flotte en ouvrant certaines plages de ports (autorisations entrantes) pour la flotte. Les autorisations entrantes déterminent les ports ouverts au trafic entrant. Vous pouvez rapidement fermer tous les ports, en limiter à quelques-uns ou ouvrir tous les ports.

Les flottes de conteneurs gérées nécessitent un paramètre supplémentaire permettant d'accéder aux processus exécutés dans un conteneur. Lorsque vous créez une définition de conteneur, vous spécifiez un ensemble de ports, un pour chaque processus qui établit une connexion. Cela consiste notamment à :

- Tous les processus du serveur de jeu qui s'exécuteront simultanément dans le conteneur du serveur de jeu. Tous les processus du serveur de jeu doivent autoriser les clients du jeu à se connecter pour rejoindre une session de jeu.
- Tout processus d'un conteneur de support auquel une source externe doit se connecter. Par exemple, vous pouvez vous connecter à distance à une application de test.

Lorsque vous définissez les paramètres du port de conteneur interne, Amazon GameLift Servers utilise-les pour calculer les autorisations entrantes externes auxquelles les clients du jeu et les autres applications peuvent se connecter. Amazon GameLift Servers gère également le mappage entre les autorisations entrantes et les ports de conteneur individuels, ce qui permet aux joueurs d'accéder à une session de jeu dans un conteneur. Ce mappage interne fournit une couche de sécurité en protégeant vos serveurs de jeu d'un accès direct aux ports à conteneurs. Vous avez la possibilité de personnaliser les paramètres des ports orientés vers l'extérieur d'une flotte selon vos besoins. Pour plus d'informations sur la configuration manuelle des ports de flotte de conteneurs, consultez [Configuration des connexions réseau](#).

Vous pouvez modifier les paramètres portuaires d'une flotte de conteneurs à tout moment. Cette modification nécessite le déploiement d'une mise à jour de la flotte.

Le schéma suivant illustre le rôle des connexions portuaires au sein d'une flotte de conteneurs. Comme indiqué, vous définissez des ports sur des conteneurs individuels et utilisez ces Amazon GameLift Servers informations pour configurer suffisamment de ports sur l'instance de flotte pour qu'ils soient mappés à chaque port de conteneur. Les autorisations entrantes des instances externes et les ports de connexion internes sont calculés Amazon GameLift Servers pour votre parc, sauf si vous choisissez de les définir manuellement.

Enregistrement des conteneurs

Dans les flottes de conteneurs gérées, les flux de sortie standard (et d'erreur standard) sont capturés pour tous les conteneurs. Cela inclut les journaux de session de jeu de votre serveur de jeu. Vous pouvez configurer un parc de conteneurs pour utiliser l'une des options suivantes pour gérer les flux de sortie :

- Enregistrez la sortie du conteneur sous forme de flux de CloudWatch journal Amazon. Chaque flux de journal fait référence à l'identifiant de la flotte et au conteneur. Si vous choisissez cette option de journalisation pour la flotte, vous spécifiez un groupe de CloudWatch journaux qui organise tous les flux de journaux de la flotte. Vous pouvez ensuite utiliser les CloudWatch fonctionnalités pour rechercher et analyser les données du journal selon vos besoins.
- Enregistrez la sortie du conteneur dans un bucket de stockage Amazon Simple Storage Service (Amazon S3). Vous pouvez consulter, partager ou télécharger le contenu selon vos besoins.
- Désactivez la journalisation. Dans ce scénario, la sortie du conteneur n'est pas enregistrée.

Amazon GameLift Servers envoie les données de journal des flottes de conteneurs gérées vers les CloudWatch services Amazon S3 de votre AWS compte. Pour consulter vos données, utilisez ces outils AWS Management Console ou d'autres outils en vous connectant à votre AWS compte et en utilisant les différents services. Vous étendez l'accès limité Amazon GameLift Servers pour effectuer ces actions en créant un rôle de service pour vos flottes de conteneurs.

Vous pouvez modifier la configuration de journalisation d'une flotte de conteneurs à tout moment. Cette modification nécessite le déploiement d'une mise à jour de la flotte.

Les flottes de conteneurs et l'agent Amazon GameLift Servers

Une architecture de conteneur couramment utilisée exécute un seul processus dans chaque conteneur. Dans une flotte de Amazon GameLift Servers conteneurs, le groupe de conteneurs de

serveurs de jeu possède un conteneur de serveur de jeu, qui exécute un processus de serveur de jeu. Cette architecture permet de Amazon GameLift Servers gérer le cycle de vie du processus de serveur de jeu unique dans chaque groupe de conteneurs de serveurs de jeu d'une instance de flotte.

Si vous choisissez de créer une architecture de conteneur qui exécute plusieurs processus de serveur de jeu dans chaque groupe de conteneurs de serveurs de jeu, vous devez trouver un moyen de gérer le cycle de vie de tous les processus. Cela inclut des tâches telles que le démarrage, l'arrêt et le remplacement de processus selon les besoins, la gestion du nombre souhaité de processus à exécuter simultanément et la gestion des états de défaillance.

Vous pouvez choisir d'utiliser l'Amazon GameLift Serversagent pour ces tâches. Pour un parc de conteneurs, l'agent implémente des instructions d'exécution qui spécifient les exécutable à exécuter (et leur nombre), fournissent les paramètres de lancement et définissent les règles relatives à l'activation du serveur de jeu. Par exemple, les instructions d'exécution peuvent indiquer à l'agent de gérer dix processus de serveur de jeu pour une utilisation en production, et un processus de serveur de jeu avec des paramètres de lancement spéciaux pour les tests.

Pour utiliser l'agent avec vos flottes de conteneurs, ajoutez-le à l'image de votre conteneur et incluez un ensemble d'instructions d'exécution. Pour plus d'informations sur l'agent, consultez [Collaborez avec l'Amazon GameLift Serversagent](#).

Comment les joueurs se connectent aux jeux

Une session de jeu est une instance de votre jeu qui s'exécute sur Amazon GameLift Servers. Pour jouer à votre jeu, un joueur peut soit rechercher et rejoindre une session de jeu existante, soit créer une nouvelle session de jeu et la rejoindre. Un joueur rejoint le jeu en créant une session de joueur pour la session de jeu. Si la session de jeu est ouverte aux joueurs, elle Amazon GameLift Servers réserve un emplacement au joueur et fournit des informations de connexion. Le joueur peut alors se connecter à la session de jeu et profiter de l'emplacement réservé.

Pour obtenir des informations détaillées sur la création et la gestion de sessions de jeu et de sessions de joueurs avec des serveurs de jeu personnalisés, consultez [Ajoutez Amazon GameLift Servers à votre client de jeu](#). Pour plus d'informations sur la connexion des joueurs à Amazon GameLift ServersRealtime, consultez [Intégration d'un client de jeu pour Amazon GameLift ServersRealtime](#).

Amazon GameLift Serversfournit plusieurs fonctionnalités liées aux sessions de jeu et aux sessions des joueurs.

Organisez des sessions de jeu sur les meilleures ressources disponibles sur plusieurs sites

Choisissez entre plusieurs options lors de la configuration de la façon dont Amazon GameLift Servers sélectionne les ressources pour héberger de nouvelles sessions de jeu. Si vous gérez des flottes sur plusieurs sites, vous pouvez créer des files d'attente de sessions de jeu qui placeront de nouvelles sessions de jeu sur n'importe quelle flotte, quel que soit l'endroit où vous vous trouvez. Utilisez des balises ping UDP pour calculer les données de latence, que la file d'attente utilise automatiquement pour hiérarchiser les emplacements de placement.

Contrôlez l'accès des joueurs aux sessions de jeu

Configurez les sessions de jeu pour autoriser ou refuser les demandes d'inscription des nouveaux joueurs, quel que soit le nombre de joueurs connectés.

Utiliser des données de jeu et de joueur personnalisées

Ajoutez des données personnalisées aux objets de session de jeu et aux objets de session de joueur. Amazon GameLift Servertransmet les données de session de jeu à un serveur de jeu lors du démarrage d'une nouvelle session de jeu. Amazon GameLift Servertransmet les données de session du joueur au serveur de jeu lorsqu'un joueur se connecte à la session de jeu.

Filtrer et trier les sessions de jeu disponibles

Utilisez la recherche et le tri par session pour trouver la meilleure correspondance possible pour un joueur potentiel, ou laissez le joueur choisir parmi une liste de sessions de jeu disponibles. Utilisez la recherche et le tri des sessions pour trouver des sessions de jeu en fonction de leurs caractéristiques. Vous pouvez également rechercher et trier selon vos propres données de jeu personnalisées.

Suivez les données d'utilisation du jeu et des joueurs

Enregistrez automatiquement les journaux des sessions de jeu terminées. Vous pouvez configurer le stockage des journaux lors de l'Amazon GameLift Serversintégration à vos serveurs de jeu. Pour de plus amples informations, veuillez consulter [Enregistrement des messages du serveur Amazon GameLift Servers](#).

Utilisez la Amazon GameLift Servers console pour afficher des informations détaillées sur les sessions de jeu, notamment les métadonnées de session, les paramètres et les données de session des joueurs. Pour plus d'informations, consultez [Sessions de jeu et de jeu dans la Amazon GameLift Servers console](#) et [Métriques](#).

Amazon GameLift Serverspoints de service

Amazon GameLift Serversles fonctionnalités sont disponibles dans plusieurs Régions AWS Zones Locales. Vous pouvez concevoir une solution d'hébergement qui place vos serveurs de jeu là où se trouvent vos joueurs.

AWS Emplacements pris en charge

Le tableau suivant répertorie les Zones Régions AWS Locales qui prennent en charge Amazon GameLift Servers les ressources. Il indique les types de ressources que vous pouvez créer dans chaque emplacement.

Emplacement géographique	Code de localisation	Région d'origine pour les flottes gérées (site unique)	Région d'origine pour les flottes gérées (sites multiples)	Emplacement distant pour les flottes gérées (multi-sites)	Flotte Anywhere	File d'attente des sessions de jeu	FlexMatch entremetteur et ensemble de règles
USA Est (Virginie du Nord)	us-east-1	Oui	Oui	Oui	Oui	Oui	Oui
USA Est (Ohio)	us-east-2	Oui		Oui	Oui	Oui	
USA Ouest (Californie du Nord)	us-west-1	Oui		Oui	Oui	Oui	
USA Ouest (Oregon)	us-west-2	Oui	Oui	Oui	Oui	Oui	Oui
Afrique (Le Cap)	af-south-1			Oui			

Emplacement géographique	Code de localisation	Région d'origine pour les flottes gérées (site unique)	Région d'origine pour les flottes gérées (sites multiples)	Emplacement distant pour les flottes gérées (multi-sites)	Flotte Anywhere	File d'attente des sessions de jeu	FlexMatch entremetteur et ensemble de règles
Asie-Pacifique (Thaïlande)	ap-southeast-7			Oui			
Asie-Pacifique (Hong Kong)	ap-east-1			Oui			
Asie-Pacifique (Malaisie)	ap-southeast-5			Oui			
Asie-Pacifique (Mumbai)	ap-south-1	Oui		Oui	Oui	Oui	
Asie-Pacifique (Osaka)	ap-northeast-3			Oui			
Asie-Pacifique (Séoul)	ap-northeast-2	Oui	Oui	Oui	Oui	Oui	Oui
Asie-Pacifique (Singapour)	ap-southeast-1	Oui		Oui	Oui	Oui	
Asie-Pacifique (Sydney)	ap-southeast-2	Oui	Oui	Oui	Oui	Oui	Oui

Emplacement géographique	Code de localisation	Région d'origine pour les flottes gérées (site unique)	Région d'origine pour les flottes gérées (sites multiples)	Emplacement distant pour les flottes gérées (multi-sites)	Flotte Anywhere	File d'attente des sessions de jeu	FlexMatch entremetteur et ensemble de règles
Asie-Pacifique (Tokyo)	ap-northeast-1	Oui	Oui	Oui	Oui	Oui	Oui
Canada (Centre)	ca-central-1	Oui		Oui	Oui	Oui	
Europe (Francfort)	eu-central-1	Oui	Oui	Oui	Oui	Oui	Oui
Europe (Irlande)	eu-west-1	Oui	Oui	Oui	Oui	Oui	Oui
Europe (Londres)	eu-west-2	Oui		Oui	Oui	Oui	
Europe (Milan)	eu-south-1			Oui			
Europe (Paris)	eu-west-3			Oui			
Europe (Stockholm)	eu-north-1			Oui			
Moyen-Orient (Bahreïn)	me-south-1			Oui			

Emplacement géographique	Code de localisation	Région d'origine pour les flottes gérées (site unique)	Région d'origine pour les flottes gérées (sites multiples)	Emplacement distant pour les flottes gérées (multi-sites)	Flotte Anywhere	File d'attente des sessions de jeu	FlexMatch entremetteur et ensemble de règles
Amérique du Sud (São Paulo)	sa-east-1	Oui		Oui	Oui	Oui	
Zone locale d'Atlanta	us-east-1-atl-1			Oui			
Zone locale de Chicago	us-east-1-chi-1			Oui			
Zone locale de Dallas	us-east-1-dfw-1			Oui			
Zone locale de Denver	us-ouest-2-den-1			Oui			
Zone locale de Houston	us-east-1-iah-1			Oui			
Zone locale de Kansas City	us-east-1-mci-1			Oui			

Emplacement géographique	Code de localisation	Région d'origine pour les flottes gérées (site unique)	Région d'origine pour les flottes gérées (sites multiples)	Emplacement distant pour les flottes gérées (multi-sites)	Flotte Anywhere	File d'attente des sessions de jeu	FlexMatch entremetteur et ensemble de règles
Zone locale de Los Angeles	us-west-2-lax-1			Oui			
Zone locale de Phoenix	us-west-2-phx-1			Oui			
Lagos, zone locale de Nigéria	af-sud-1-los-1			Oui			

Note

Tous ne Régions AWS sont pas activés par défaut pour un Compte AWS. Si vous souhaitez disposer d'une flotte multi-sites avec des instances dans ces régions, vous devez les activer. Pour plus d'informations sur les régions qui ne sont pas activées par défaut et sur la manière de les activer, consultez la section [Gestion Régions AWS](#) dans le Références générales AWS. Les flottes que vous avez créées avant le 28 février 2022 ne sont pas concernées. En outre, vous devez mettre à jour votre politique d'Amazon GameLift Servers administrateur pour autoriser l'`ec2:DescribeRegions` action. Pour un exemple de politique avec des régions qui ne sont pas activées par défaut, consultez [Exemples d'autorisations d'administration](#).

Emplacements pour l'hébergement géré

Amazon GameLift Servers l'hébergement géré déploie des flottes de ressources de serveurs de jeux. Chaque flotte est créée dans une Région AWS, qui est la région d'origine de la flotte. La région d'origine d'une flotte est référencée dans le numéro de ressource Amazon (ARN) de la flotte.

Vous pouvez déployer une flotte d'une seule région, avec des ressources d'hébergement uniquement dans la région d'origine. Vous pouvez également déployer une flotte sur plusieurs sites, avec des ressources d'hébergement sur plusieurs sites géographiques. Une flotte multisite comprend une région d'origine et un ou plusieurs sites distants. Lorsque vous gérez la capacité d'hébergement d'un parc de plusieurs sites, vous pouvez définir la capacité de chaque site individuellement.

Les sites distants d'une flotte multi-sites peuvent être des Zones Régions AWS Locales ou autres. Une zone locale est une extension d'une Région AWS. Il vous permet de placer les ressources de calcul au plus près des utilisateurs afin de réduire la latence du jeu. Pour plus d'informations, consultez [Local Zones AWS](#). Le code de localisation d'une zone locale est son code de région parent suivi d'un identifiant de localisation physique. Par exemple, le code de la zone locale de Los Angeles est `us-west-2-lax-1`.

Le schéma suivant illustre une flotte à sites multiples avec des ressources réparties dans deux Régions AWS et une zone locale. La région d'origine de la flotte est `us-west-2`, et elle compte deux sites éloignés : `us-east-2` région et zone `us-west-2-lax-1` locale.

Outre les ressources de flotte, l'hébergement géré avec Amazon GameLift Servers peut également utiliser les types de ressources suivants. Vous créez ces ressources dans un fichier spécifique Région AWS qui prend en charge le type de ressource.

- **Construction** — Il s'agit d'une version de serveur de jeu destinée à être hébergée avec une EC2 flotte gérée. Créez une ressource de construction dans la même région que la flotte dans laquelle elle sera déployée.
- **Script** — Il s'agit d'un script de configuration pour héberger un jeu avec Amazon GameLift Servers Realtime. Créez une ressource de script dans la même région que la flotte dans laquelle elle sera déployée.
- **Définition du groupe de conteneurs et image du conteneur** — Il s'agit d'une configuration permettant d'exécuter des conteneurs sur un parc de conteneurs géré. Il identifie une ou plusieurs images de conteneurs à l'aide d'un logiciel à déployer sur le parc de conteneurs. Créez une définition de groupe de conteneurs et toutes les images de conteneurs (qui sont stockées dans

un référentiel Amazon Elastic Container Registry) dans la même région que le parc dans lequel ils seront déployés.

- File d'attente des sessions de jeu : cette ressource traite les demandes de sessions de jeu et lance de nouvelles sessions de jeu. Le traitement a lieu à l'endroit Région AWS où se trouve la file d'attente. Pour réduire la latence lors du processus de placement des sessions de jeu, créez une file d'attente géographiquement proche des joueurs qui l'utiliseront.

Des emplacements pour Amazon GameLift Servers n'importe où

Une flotte Amazon GameLift Servers Anywhere est un ensemble de matériel d'hébergement que vous fournissez. Vous gérez toutes les activités sur vos ressources d'hébergement, y compris le déploiement du logiciel du serveur de jeu, sa mise à jour et le démarrage des processus du serveur. Vous créez une flotte Anywhere pour connecter le Amazon GameLift Servers service à vos ressources d'hébergement autogérées. Amazon GameLift Servers gère le placement des sessions de jeu : traitement des demandes d'inscription des joueurs, localisation des ressources d'hébergement disponibles, lancement de nouvelles sessions de jeu et fourniture d'informations de connexion aux clients du jeu. Vous pouvez créer une flotte Anywhere dans n'importe quel parc Régions AWS qui les prend en charge.

Vous ajoutez des instances de matériel d'hébergement à un parc Anywhere en l'enregistrant. Chaque instance enregistrée doit être associée à un emplacement personnalisé. Les emplacements personnalisés ne sont pas liés aux Régions AWS Zones Locales. Ils sont utilisés pour représenter l'emplacement physique du matériel.

Pour plus d'informations sur la création d'une flotte Anywhere et le test de l'intégration de votre serveur de jeu, consultez [Créez un Amazon GameLift Servers Flotte Anywhere](#) et [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#).

Emplacements pour Amazon GameLift Servers FlexMatch

FlexMatch les ressources sont utilisées pour traiter les demandes de matchmaking des joueurs. Ils incluent une ressource de configuration de matchmaking et une ressource d'ensemble de règles. Le traitement a lieu Région AWS là où se trouvent les FlexMatch ressources. Pour réduire la latence dans le processus de matchmaking, créez les ressources géographiquement à proximité des joueurs qui les utiliseront. Une configuration de matchmaking et l'ensemble de règles qu'elle utilise doivent se trouver dans la même configuration Région AWS. Vous pouvez créer des FlexMatch ressources dans n'importe lequel d'entre eux Régions AWS qui les soutiennent.

Pour plus d'informations sur la configuration FlexMatch de votre solution d'hébergement, consultez le [guide du Amazon GameLift Servers FlexMatch développeur](#).

Amazon GameLift Servers en Chine

Lorsque vous utilisez Amazon GameLift Servers des ressources dans la région de Chine (Pékin), gérée par Sinnet, ou dans la région de Chine (Ningxia), gérée par la NWCD, vous devez disposer d'un compte distinct (AWS Chine). Sachez que certaines fonctionnalités ne sont pas disponibles dans les régions de Chine. Pour plus d'informations sur l'utilisation Amazon GameLift Servers dans ces régions, consultez les ressources suivantes :

- [Amazon Web Services en Chine](#)
- [Amazon GameLift Servers](#) (Commencer à utiliser Amazon Web Services en Chine)

Tarification pour Amazon GameLift Servers

Tarification pour Amazon GameLift Servers varie en fonction du type de service et des fonctionnalités que vous utilisez. Pour une discussion complète sur la tarification, y compris des exemples de scénarios, voir [Amazon GameLift Servers Tarification](#).

Vous pouvez utiliser ... Amazon GameLift Servers sans encourir de frais avec le niveau AWS gratuit. Vous pouvez utiliser les avantages du niveau gratuit si vous êtes AWS client depuis moins de 12 mois et si vous respectez les limites d'utilisation du niveau gratuit. Pour plus d'informations, consultez la page sur l'[offre gratuite AWS](#).

Les bases de coûts pour chacun Amazon GameLift Servers les services sont les suivants :

- Amazon GameLift Servers hébergement géré : vous payez en fonction (1) de l'utilisation horaire de l'instance pour l'hébergement de sessions de jeu et (2) du transfert de données pour le trafic entre les clients du jeu et les serveurs de jeu hébergés. Les coûts d'utilisation des instances varient en fonction du type/de la taille de l'instance Région AWS, du système d'exploitation et du fait qu'il s'agit d'une instance ponctuelle ou à la demande.
- Amazon GameLift Servers FlexMatch autonome — Vous payez en fonction (1) du nombre de demandes de matchmaking que vous faites, et (2) du temps de traitement nécessaire pour évaluer les demandes de match et proposer des matchs.
- Amazon GameLift Servers Hébergement n'importe où — Vous payez en fonction (1) du nombre de sessions de jeu qui Amazon GameLift Servers place sur Anywhere les calculs et (2) le nombre de minutes de connexion au processus serveur.

- **Amazon GameLift Servers FleetIQ** — Vous payez des frais pour EC2 chaque instance d'un groupe de serveurs de jeu. Les frais sont dérivés du prix horaire de l'instance et s'ajoutent au coût EC2 d'utilisation. Les coûts d'utilisation des instances varient en fonction du type/de la taille de l'instance Région AWS, du système d'exploitation et du fait qu'il s'agit d'une instance ponctuelle ou à la demande.

Rubriques

- [Generate Amazon GameLift Servers estimations de prix](#)
- [Gérez votre Amazon GameLift Servers frais d'hébergement](#)

Generate Amazon GameLift Servers estimations de prix

Avec Calculateur de tarification AWS, vous pouvez [créer une estimation de prix pour Amazon GameLift Servers](#). Vous n'avez pas besoin de connaissances approfondies AWS pour utiliser la calculatrice. Compte AWS

Calculateur de tarification AWS le calculateur vous guide à travers les décisions qui influent sur les coûts des services pour vous donner une idée du montant Amazon GameLift Servers peut coûter cher pour votre projet de jeu. Si vous ne savez pas encore comment vous comptez utiliser Amazon GameLift Servers, puis utilisez les valeurs par défaut pour générer une estimation. Lors de la planification de l'utilisation en production, le calculateur peut vous aider à tester des scénarios potentiels et à générer des estimations plus précises.

Vous pouvez l'utiliser Calculateur de tarification AWS pour générer des estimations pour les éléments suivants Amazon GameLift Servers options d'hébergement :

- [Estimation Amazon GameLift Servers hébergement géré](#)

Estimation Amazon GameLift Servers hébergement géré

Cette option fournit une estimation des coûts d'hébergement de vos jeux sur Amazon GameLift Servers serveurs gérés, y compris les coûts d'utilisation des instances de serveur et de transfert de données. Avec Amazon GameLift Servers hébergement géré, il n'y a aucun coût supplémentaire pour FlexMatch matchmaking.

Si vous hébergez ou prévoyez d'héberger des serveurs de jeu dans plusieurs AWS régions ou sur plusieurs types d'instances, créez une estimation pour chaque région et type d'instance.

Amazon GameLift Servers Instances

Cette section vous aide à estimer le type et le nombre de ressources informatiques dont vous avez besoin pour héberger des sessions de jeu pour vos joueurs. Amazon GameLift Servers utilise les [instances Amazon Elastic Compute Cloud \(Amazon EC2\)](#) pour gérer les serveurs de jeux. Entrée Amazon GameLift Servers, vous déployez un parc d'instances avec un type d'instance et un système d'exploitation spécifiques. Si vous possédez ou prévoyez d'avoir plusieurs flottes, créez une estimation pour chaque flotte.

Pour commencer, ouvrez le menu [Configurer Amazon GameLift Servers page](#) de Calculateur de tarification AWS. Ajoutez une description, choisissez une région, puis choisissez Estimer Amazon GameLift Servers hébergement (instance + transfert de données sortant). Sous Amazon GameLift Servers instances, renseignez les champs suivants :

- Nombre maximal de joueurs simultanés (maximum de CCU)

Il s'agit du nombre maximum de joueurs pouvant se connecter à vos serveurs de jeu en même temps. Ce champ indique la capacité d'hébergement Amazon GameLift Servers doit répondre à la demande maximale des joueurs. Entrez le nombre maximal quotidien de joueurs que vous comptez accueillir en utilisant des instances dans AWS la région de votre choix.

Par exemple, si vous souhaitez autoriser 1 000 joueurs à se connecter à votre jeu à tout moment, conservez la valeur par défaut de **1000**.

- CCU moyen par heure en pourcentage du CCU quotidien de pointe

Il s'agit du nombre moyen de joueurs simultanés par heure sur une période de 24 heures. Nous utilisons cette valeur pour estimer la quantité de capacité d'hébergement durable qui Amazon GameLift Servers doit être maintenu pour vos joueurs. Si vous ne savez pas quelle valeur de pourcentage utiliser, conservez la valeur par défaut de **50** pourcentage. Pour les jeux où la demande de joueurs est stable, nous vous recommandons de saisir une valeur en **70** pourcentage.

Par exemple, si votre jeu a une CCU horaire moyenne de 6 000 et une CCU maximale de 10 000, entrez la valeur du **60** pourcentage.

- Sessions de jeu par instance

Il s'agit du nombre de sessions de jeu que chacune de vos instances de serveur de jeu peut héberger simultanément. Les facteurs qui peuvent influencer ce nombre incluent les besoins en ressources de votre serveur de jeu, le nombre de joueurs à héberger lors de chaque session de jeu et les attentes des joueurs en matière de performance. Si vous connaissez le nombre de sessions

de jeu simultanées pour votre jeu, entrez cette valeur. Vous pouvez également conserver la valeur par défaut de **20**.

- Nombre de joueurs par session de jeu

Il s'agit du nombre moyen de joueurs qui se connectent à une session de jeu, tel que défini dans votre conception de jeu. Si vous avez des modes de jeu avec un nombre de joueurs différent, estimez le nombre moyen de joueurs par session de jeu sur l'ensemble de votre jeu. La valeur par défaut est **8**.

- % de la mémoire tampon inactive de l'instance

Il s'agit du pourcentage de capacité d'hébergement inutilisée à maintenir en réserve pour faire face à des pics soudains de demande de joueurs. La taille de la mémoire tampon est un pourcentage du nombre total d'instances d'un parc. La valeur par défaut est le **10** pourcentage.

Par exemple, avec un tampon d'inactivité de 20 %, un parc de joueurs comptant 100 instances actives conserve 20 instances inactives.

- Instance ponctuelle %

Amazon GameLift Servers les flottes peuvent utiliser une combinaison d'instances à la demande et d'instances ponctuelles. Alors que les instances à la demande offrent une disponibilité plus fiable, les instances Spot constituent une alternative très rentable. Nous vous recommandons d'utiliser une combinaison pour optimiser à la fois les économies de coûts et la disponibilité. Pour plus d'informations sur la façon dont Amazon GameLift Servers utilise des instances Spot, voir [Instances à la demande et instances ponctuelles](#).

Dans ce champ, entrez le pourcentage d'instances ponctuelles à gérer dans un parc. Nous recommandons un pourcentage d'instance Spot compris entre 50 et 85 %. La valeur par défaut est le **50** pourcentage.

Par exemple, si vous déployez un parc de 100 instances et que vous spécifiez un **40** pourcentage, Amazon GameLift Servers travaille à la maintenance de 60 instances à la demande et de 40 instances ponctuelles.

- Type d'instance

Amazon GameLift Servers les flottes peuvent utiliser différents types d' EC2 instances Amazon dont la puissance de calcul, la mémoire, le stockage et les capacités réseau varient. Lorsque vous configurez un Amazon GameLift Servers fleet, choisissez le type d'instance qui correspond le

mieux aux besoins de votre jeu. Pour plus d'informations sur la sélection d'un type d'instance avec Amazon GameLift Servers, voir [Choisissez les ressources de calcul pour un parc géré](#).

Si vous connaissez le type d'instance que vous utilisez ou que vous comptez utiliser dans votre Amazon GameLift Servers flotte, choisissez ce type. Si vous n'êtes pas sûr du type à choisir, pensez à choisir c5.large. Il s'agit d'un type à haute disponibilité dont la taille et les capacités sont moyennes.

- Système d'exploitation

Ce champ indique le système d'exploitation sur lequel s'exécutent vos serveurs de jeu, qu'il s'agisse de Linux ou de Windows. La valeur par défaut est Linux.

Transfert de données sortantes (DTO)

Cette section vous aide à estimer le coût du trafic entre vos clients de jeu et les serveurs de jeu. Les frais de transfert de données s'appliquent uniquement au trafic sortant. Le transfert de données entrantes est gratuit.

Sur le [Configure Amazon GameLift Servers page](#) de Calculateur de tarification AWS, développez Data transfer out (DTO), puis complétez les champs suivants :

- Type d'estimation DTO

Vous pouvez choisir d'estimer le DTO de l'une des deux manières suivantes, en fonction de la manière dont vous suivez le transfert de données pour votre jeu.

- Par mois (en Go) — Si vous suivez le trafic mensuel de vos serveurs de jeux, choisissez ce type.
- Par joueur — Si vous suivez le transfert de données par joueur, choisissez ce type. Il s'agit du type par défaut.

Dans le champ suivant, vous estimez le DTO par joueur en fonction du nombre d'heures de jeu que vous avez calculé dans la section précédente.

- DTO par mois (en Go)

Si vous avez choisi le type d'estimation de DTO par mois (en Go), entrez votre consommation mensuelle estimée de DTO en Go pour chaque instance, par région.

- DTO par joueur

Si vous avez choisi le type d'estimation du DTO par joueur, saisissez l'estimation de l'utilisation du DTO par joueur de votre jeu en Ko/s. La valeur par défaut est 4.

Lorsque vous avez terminé de configurer votre Amazon GameLift Servers estimation du prix, choisissez Ajouter à mon estimation. Pour plus d'informations sur la création et la gestion des estimations dans Calculateur de tarification AWS, voir [Créer une estimation, configurer un service et ajouter d'autres services](#) dans le Guide de Calculateur de tarification AWS l'utilisateur.

Gérez votre Amazon GameLift Servers frais d'hébergement

Votre AWS facture reflète les frais d'hébergement de vos jeux. Vous pouvez consulter les frais estimés pour le mois en cours et les frais finaux des mois précédents sur la console de facturation à l'adresse <https://console.aws.amazon.com/costmanagement/>. Pour plus d'informations sur les outils et les ressources destinés à vous aider à gérer vos AWS coûts, consultez le [guide de AWS Billing l'utilisateur](#). Ce guide peut vous aider à évaluer votre consommation de ressources, à déterminer votre utilisation future et à déterminer vos besoins en matière de mise à l'échelle.

Tenez compte de ces conseils pour vous aider à gérer le coût de Amazon GameLift Servers services.

Créez des alertes de facturation pour surveiller l'utilisation

Configurez une alerte d'utilisation du niveau AWS gratuit pour vous avertir lorsque votre utilisation approche ou dépasse les limites du niveau gratuit pour Amazon GameLift Servers et autres Services AWS. Vous pouvez configurer les alertes pour qu'elles prennent des mesures en fonction de vos niveaux d'utilisation. Par exemple, vous pouvez définir automatiquement votre budget à zéro lorsque vous atteignez la limite du niveau gratuit.

Vous pouvez également configurer des alertes CloudWatch de facturation Amazon pour recevoir des notifications lorsque l'utilisation atteint des seuils personnalisés.

Pour plus d'informations, consultez les rubriques suivantes dans le guide de AWS Billing l'utilisateur :

- [Suivi de votre utilisation du niveau AWS gratuit](#)
- [Préférences relatives aux alertes de facturation](#)

Suivez les coûts par Amazon GameLift Servers flotte

Utilisez des balises de répartition des AWS coûts pour organiser et suivre les coûts d'hébergement de vos jeux en fonction de Amazon GameLift Servers EC2 Flottes Amazon et autres ressources. En étiquetant vos flottes, individuellement ou par groupes, vous pouvez créer des rapports de répartition des coûts qui catégorisent les coûts en fonction de l'étiquette attribuée. Vous pouvez utiliser ce type de rapport pour identifier la manière dont les flottes contribuent à vos coûts d'hébergement. Vous pouvez également utiliser les balises pour filtrer les vues dans AWS Cost Explorer.

Pour en savoir plus, consultez les rubriques suivantes :

- [Utilisation des balises de répartition des AWS coûts](#), guide de AWS Billing l'utilisateur
- [Analyse de vos AWS coûts avec Cost Explorer](#), guide de AWS Cost Management l'utilisateur

Réduisez à zéro la capacité du parc géré

Les flottes gérées peuvent continuer à entraîner des coûts même lorsqu'elles ne sont pas utilisées pour héberger des sessions de jeu. Pour éviter d'encourir des frais inutiles, [réduisez votre parc](#) à zéro lorsqu'il n'est pas utilisé. Si vous utilisez le dimensionnement automatique, suspendez cette activité et définissez manuellement la capacité du parc.

Démarrer avec Amazon GameLift Servers

Profitez de ces ressources de démarrage pour en savoir plus sur le Amazon GameLift Servers service et sur la façon dont vous pouvez commencer à développer une solution d'hébergement personnalisée pour vos jeux multijoueurs basés sur des sessions.

Avant de commencer

- Créez un Compte AWS (ou désignez un existant) à utiliser avec Amazon GameLift Servers.
- Configurez les utilisateurs avec des autorisations pour les AWS services associés Amazon GameLift Servers et les services connexes.
- Sélectionnez l'homme dans lequel Région AWS vous souhaitez travailler. Pour le développement, choisissez une région proche de chez vous. Vous pouvez changer de région à tout moment.

[Configurez un Compte AWS](#)

Options d'intégration rapide

Essayez ces outils de démarrage rapide pour mettre rapidement en place une solution d'hébergement de base avec un développement rationalisé. Ces outils sont idéaux pour la validation de concept et le prototypage, ou pour créer des environnements de test pour le développement itératif rapide de jeux. Après avoir utilisé ces outils pour déployer un serveur de jeu à des fins d'hébergement, vous pouvez utiliser la Amazon GameLift Servers console et les outils d'API pour surveiller les performances de la flotte, gérer les sessions de jeu et analyser les indicateurs.

- [Enveloppe de serveur de jeu pour Amazon GameLift Servers](#) — Cet outil est le moyen le plus rapide et le plus simple d'héberger votre serveur de jeu et d'exécuter des sessions de jeu avec Amazon GameLift Servers, sans aucune modification du code du jeu. Le wrapper du serveur de jeu offre des fonctionnalités de base de gestion des sessions de jeu et un déploiement rationalisé du serveur de jeu. C'est l'outil idéal pour effectuer une évaluation pratique de Amazon GameLift Servers l'utilisation de votre propre projet de jeu ou d'un exemple de projet. Lorsque vous êtes prêt à créer une solution d'hébergement de jeux personnalisée, passez à l'une des options de développement personnalisées avec intégration complète avec le SDK du serveur pour Amazon GameLift Servers. Si votre jeu n'a pas besoin d'une solution d'hébergement personnalisée, vous pouvez continuer à utiliser le wrapper de serveur de jeu pour déployer et héberger vos serveurs de jeu en production.

- [Amazon GameLift Serversplugin pour Unreal Engine ou Unity](#) — Les plugins vous proposent des flux de travail graphiques et des exemples de ressources pour vous guider dans les étapes initiales et déployer votre serveur de jeu avec une solution d'hébergement de base. Utilisez le plugin pour configurer l'hébergement avec des flottes Anywhere autogérées, ou pour déployer des flottes gérées ou des EC2 flottes de conteneurs basées sur le cloud. Lorsque vous êtes prêt à développer une solution d'hébergement personnalisée, vous pouvez vous appuyer sur vos solutions intégrées au plugin.
- [Kit de démarrage pour les conteneurs Amazon GameLift Servers gérés](#) : ce kit rationalise les tâches d'intégration d'un serveur de jeu, de préparation d'une image de conteneur de serveur de jeu et de déploiement d'une flotte de conteneurs à des fins d'hébergement. Pour l'intégration, le kit ajoute des fonctionnalités essentielles de gestion de session de jeu à votre serveur de jeu. Le kit utilise des modèles préconfigurés pour créer une flotte de conteneurs et un pipeline de déploiement automatisé pour le serveur de jeu. Lorsque vous serez prêt à ajouter des fonctionnalités complètes de gestion des sessions de jeu, suivez l'une des feuilles de route de développement personnalisées pour intégrer le SDK du serveur pour. Amazon GameLift Servers

Options de développement personnalisées

Suivez l'une de ces feuilles de route de développement pour commencer à créer une solution d'hébergement personnalisée complète pour votre jeu. Les feuilles de route fournissent des conseils détaillés sur la façon de créer, de tester et de personnaliser chaque composant de votre solution d'hébergement.

- [Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Managed EC2](#)
- [Feuille de route de développement pour l'hébergement avec des conteneurs Amazon GameLift Servers gérés](#)
- [Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Anywhere](#)
- [Feuille de route de développement pour l'hébergement hybride avec Amazon GameLift Servers](#)

Exemples Amazon GameLift Servers

Si vous envisagez de l'utiliser Amazon GameLift Servers pour gérer votre serveur de jeu personnalisé, ou si vous souhaitez en tirer parti Amazon GameLift ServersRealtime, nous vous recommandons d'essayer les exemples suivants avant d'utiliser le service pour votre propre jeu.

L'exemple de serveur de jeu personnalisé vous donne de l'expérience en matière d'hébergement de jeux sur la Amazon GameLift Servers console. L'Amazon GameLift ServersRealtimeexemple montre comment préparer un jeu pour l'hébergement à l'aide de Realtime serveurs.

Exemple de serveur de jeu personnalisé

Cet exemple illustre le processus de déploiement d'un exemple de serveur de jeu sur une EC2 flotte Amazon GameLift Servers gérée à des fins d'hébergement. Utilisez l'exemple de client de jeu pour vous connecter à une session de jeu en direct. Vous pouvez découvrir comment utiliser Amazon GameLift Servers .tools, y compris la console et la AWS CLI, pour surveiller les performances d'hébergement et l'utilisation de la flotte.

L'exemple vous guide à travers les étapes suivantes :

- Téléchargez l'exemple de build du serveur de jeu.
- Créez une flotte pour exécuter le build du serveur de jeu.
- Procurez-vous l'exemple de client de jeu et utilisez-le pour vous connecter à un serveur de jeu et rejoindre une session de jeu.
- Passez en revue les statistiques de la flotte et des sessions de jeu.

Démarrez plusieurs clients de jeu et jouez au jeu pour générer des données d'hébergement. Utilisez la Amazon GameLift Servers console pour consulter les ressources d'hébergement, suivre les indicateurs et explorer les options permettant d'augmenter la capacité d'hébergement de la flotte.

Pour commencer, connectez-vous à la [Amazon GameLift Serversconsole](#). Dans la barre de navigation de gauche, allez dans Ressources, essayez un exemple de jeu.

Amazon GameLift ServersRealtimeexemple

Cet exemple est un didacticiel complet qui vous explique comment déployer un exemple de jeu multijoueur, Mega Frog Race, avec Amazon GameLift ServersRealtime. Le didacticiel explique comment intégrer votre client de jeu au Realtime SDK et déployer une solution d'hébergement complète avec des Realtime serveurs sur des EC2 flottes gérées.

Pour un didacticiel pratique, consultez [la section Création de serveurs pour les jeux mobiles multijoueurs en quelques lignes JavaScript](#) sur le blog AWS dédié aux jeux. Pour le code source de Mega Frog Race, consultez le [GitHub dépôt](#).

Le code source inclut les éléments suivants :

- Client de jeu : code source du client de jeu C++, créé dans Unity. Le client du jeu obtient les informations de connexion à la session de jeu, se connecte au serveur et échange des mises à jour avec les autres joueurs.
- Service principal : code source d'une AWS Lambda fonction qui gère les appels directs à l'API du service pour Amazon GameLift Servers.
- Realtiescript : fichier de script source qui configure un parc de Realtime serveurs pour le jeu. Ce script inclut la configuration minimale requise pour que chaque Realtime serveur puisse communiquer avec Amazon GameLift Servers et héberger des sessions de jeu.

Après avoir configuré l'exemple de jeu pour l'hébergement, utilisez-le comme point de départ pour expérimenter d'autres Amazon GameLift Servers fonctionnalités telles que FlexMatch.

Configurez un Compte AWS

Pour commencer à utiliser Amazon GameLift Servers, créez et configurez votre Compte AWS. La création d'un Compte AWS. Cette section explique comment créer votre compte, configurer vos utilisateurs et configurer les autorisations.

Rubriques

- [Inscrivez-vous pour un Compte AWS](#)
- [Création d'un utilisateur doté d'un accès administratif](#)
- [Définissez les autorisations utilisateur pour Amazon GameLift Servers](#)
- [Configurer l'accès programmatique pour les utilisateurs](#)
- [Configurez l'accès programmatique à votre jeu](#)
- [Exemples d'autorisations IAM pour Amazon GameLift Servers](#)
- [Configurer un rôle de service IAM pour Amazon GameLift Servers](#)

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'<https://portal.aws.amazon.com/billing/inscription>.
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisissez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les [tâches nécessitant un accès utilisateur racine](#).

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à <https://aws.amazon.com/> et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez l'utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en [AWS Management Console](#) tant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez [Connexion en tant qu'utilisateur racine](#) dans le Guide de l'utilisateur Connexion à AWS .

2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

Pour obtenir des instructions, voir [Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root \(console\)](#) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

Pour obtenir des instructions, consultez [Activation d' AWS IAM Identity Center](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir [Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center](#) dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu'utilisateur doté d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section [Connexion au portail AWS d'accès](#) dans le guide de l'Connexion à AWS utilisateur.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

Pour obtenir des instructions, consultez [Création d'un ensemble d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

Pour obtenir des instructions, consultez [Ajout de groupes](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

Définissez les autorisations utilisateur pour Amazon GameLift Servers

Créez des utilisateurs supplémentaires ou étendez les autorisations d'accès aux utilisateurs existants selon les besoins de vos Amazon GameLift Servers ressources. La meilleure pratique ([meilleures pratiques de sécurité dans IAM](#)) consiste à appliquer des autorisations de moindre privilège à tous les utilisateurs. Pour obtenir des conseils sur la syntaxe des autorisations, voir [Exemples d'autorisations IAM pour Amazon GameLift Servers](#).

Suivez les instructions ci-dessous pour définir les autorisations utilisateur en fonction de la façon dont vous gérez les utilisateurs de votre AWS compte.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

Créez un jeu d'autorisations. Suivez les instructions de la rubrique [Création d'un jeu d'autorisations](#) du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :

- Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique [Création d'un rôle pour un utilisateur IAM](#) dans le Guide de l'utilisateur IAM.
- (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique [Ajout d'autorisations à un utilisateur \(console\)](#) du Guide de l'utilisateur IAM.

Lorsque vous travaillez avec des utilisateurs IAM, il est recommandé de toujours associer des autorisations à des rôles ou à des groupes d'utilisateurs, et non à des utilisateurs individuels.

Configurer l'accès programmatique pour les utilisateurs

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identification à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Si vous utilisez des clés d'accès, consultez la section [Meilleures pratiques de gestion des clés AWS d'accès](#).

Configurez l'accès programmatique à votre jeu

La plupart des jeux utilisent des services principaux pour communiquer à Amazon GameLift Servers l'aide du AWS SDKs. Utilisez un service principal (agissant pour le compte d'un client de jeu) pour demander des sessions de jeu, placer des joueurs dans des jeux et effectuer d'autres tâches. Ces services ont besoin d'un accès programmatique et d'informations d'identification de sécurité pour authentifier les appels à l'API du service pour. Amazon GameLift Servers

En Amazon GameLift Servers effet, vous gérez cet accès en créant un utilisateur joueur dans AWS Identity and Access Management (IAM). Gérez les autorisations des utilisateurs des joueurs via l'une des options suivantes :

- Créez un rôle IAM avec les autorisations d'utilisateur du joueur et autorisez celui-ci à assumer le rôle en cas de besoin. Le service principal doit inclure le code permettant d'assumer ce rôle avant d'envoyer des demandes à Amazon GameLift Servers. Conformément aux meilleures pratiques de sécurité, les rôles fournissent un accès temporaire limité. Vous pouvez utiliser des rôles pour les charges de travail exécutées sur AWS des ressources ([rôles IAM](#)) ou en dehors de AWS ([IAM Roles Anywhere](#)).
- Créez un groupe d'utilisateurs IAM avec des autorisations d'utilisateur joueur et ajoutez votre utilisateur joueur au groupe. Cette option fournit à l'utilisateur de votre joueur des informations d'identification à long terme, que le service principal doit stocker et utiliser lors de ses communications. Amazon GameLift Servers

Pour la syntaxe de la politique d'autorisation, voir [Exemples d'autorisations utilisateur pour les joueurs](#).

Pour plus d'informations sur la gestion des autorisations destinées à être utilisées par une charge de travail, voir [Identités IAM : informations d'identification temporaires dans IAM](#).

Exemples d'autorisations IAM pour Amazon GameLift Servers

Utilisez la syntaxe de ces exemples pour définir des autorisations AWS Identity and Access Management (IAM) pour les utilisateurs qui ont besoin d'accéder aux Amazon GameLift Servers ressources. Pour plus d'informations sur la gestion des autorisations des utilisateurs, consultez [Définissez les autorisations utilisateur pour Amazon GameLift Servers](#). Lorsque vous gérez des autorisations pour des utilisateurs extérieurs à l'IAM Identity Center, il est recommandé de toujours associer les autorisations aux rôles ou aux groupes d'utilisateurs IAM, et non à des utilisateurs individuels.

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que solution autonome, consultez [Configurer votre Compte AWS pour Amazon GameLift Servers FleetIQ](#).

Exemples d'autorisations d'administration

Ces exemples donnent à un administrateur d'hébergement ou à un développeur un accès ciblé pour gérer les ressources d'hébergement de Amazon GameLift Servers jeux.

Exemple Syntaxe pour les autorisations d'accès Amazon GameLift Servers complet aux ressources

L'exemple suivant étend l'accès complet à toutes les Amazon GameLift Servers ressources.

JSON

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

Exemple Syntaxe pour les autorisations relatives aux Amazon GameLift Servers ressources avec prise en charge des régions qui ne sont pas activées par défaut

L'exemple suivant étend l'accès à toutes les Amazon GameLift Servers ressources et AWS régions qui ne sont pas activées par défaut. Pour plus d'informations sur les régions qui ne sont pas activées par défaut et sur la manière de les activer, consultez la section [Gestion Régions AWS](#) dans le Références générales AWS.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "gamelift:*"
    ],
    "Resource": "*"
  }
}
```

Exemple Syntaxe de la Amazon GameLift Servers ressource permettant d'accéder aux images de conteneurs dans Amazon ECR

L'exemple suivant étend l'accès aux actions Amazon GameLift Servers Amazon Elastic Container Registry (Amazon ECR) dont les utilisateurs ont besoin lorsqu'ils travaillent avec des flottes de conteneurs gérées.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ecr:DescribeImages",
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer"
    ],
    "Resource": "*"
  }
}
```

Exemple Syntaxe pour les Amazon GameLift Servers ressources et **PassRole** les autorisations

L'exemple suivant étend l'accès à toutes les Amazon GameLift Servers ressources et permet à un utilisateur de transmettre un rôle de service IAM à Amazon GameLift Servers. Un rôle de service Amazon GameLift Servers limite la capacité d'accéder à d'autres ressources et services en votre nom, comme décrit dans [Configurer un rôle de service IAM pour Amazon GameLift Servers](#). Par exemple, lorsque vous répondez à une CreateBuild demande, Amazon GameLift Servers vous devez accéder à vos fichiers de compilation dans un compartiment Amazon S3. Pour plus d'informations sur cette PassRole action, voir [IAM : transmettre un rôle IAM à un AWS service spécifique](#) dans le Guide de l'utilisateur IAM.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
    "Effect": "Allow",
    "Action": "gamelift:*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "gamelift.amazonaws.com"
      }
    }
  }
]
```

Exemples d'autorisations utilisateur pour les joueurs

Ces exemples permettent à un service principal ou à une autre entité d'effectuer des appels d'API vers l'Amazon GameLift Servers API. Ils couvrent les scénarios courants de gestion des sessions de jeu, des sessions des joueurs et du matchmaking. Pour en savoir plus, consultez [Configurez l'accès programmatique à votre jeu](#).

Exemple Syntaxe pour les autorisations de placement dans les sessions de jeu

L'exemple suivant étend l'Amazon GameLift Servers API à l'accès aux files d'attente de placement des sessions de jeu pour créer des sessions de jeu et gérer les sessions des joueurs.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlayerPermissionsForGameSessionPlacements",
      "Effect": "Allow",
      "Action": [
        "gamelift:StartGameSessionPlacement",
        "gamelift:DescribeGameSessionPlacement",
        "gamelift:StopGameSessionPlacement",
        "gamelift>CreatePlayerSession",

```

```
    "gamelift:CreatePlayerSessions",
    "gamelift:DescribeGameSessions"
  ],
  "Resource": "*"
}
```

Exemple Syntaxe pour les autorisations de matchmaking

L'exemple suivant étend l'accès à ceux Amazon GameLift Servers APIs qui gèrent les activités FlexMatch de matchmaking. FlexMatch associe les joueurs à des sessions de jeu nouvelles ou existantes et lance le placement de sessions de jeu pour les jeux hébergés sur Amazon GameLift Servers. Pour plus d'informations FlexMatch, voir [Qu'est-ce que c'est Amazon GameLift Servers FlexMatch ?](#)

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForGameSessionMatchmaking",
    "Effect": "Allow",
    "Action": [
      "gamelift:StartMatchmaking",
      "gamelift:DescribeMatchmaking",
      "gamelift:StopMatchmaking",
      "gamelift:AcceptMatch",
      "gamelift:StartMatchBackfill",
      "gamelift:DescribeGameSessions"
    ],
    "Resource": "*"
  }
}
```

Exemple Syntaxe pour les autorisations de placement manuel dans les sessions de jeu

L'exemple suivant étend l'accès aux sessions de jeu Amazon GameLift Servers APIs qui créent manuellement des sessions de jeu et des sessions de joueurs sur des flottes spécifiques. Ce scénario prend en charge les jeux qui n'utilisent pas de files d'attente de placement, tels que les jeux

qui permettent aux joueurs de s'inscrire en choisissant parmi une liste de sessions de jeu disponibles (la méthode list-and-pick « »).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "PlayerPermissionsForManualGameSessions",
    "Effect": "Allow",
    "Action": [
      "gamelift:CreateGameSession",
      "gamelift:DescribeGameSessions",
      "gamelift:SearchGameSessions",
      "gamelift:CreatePlayerSession",
      "gamelift:CreatePlayerSessions",
      "gamelift:DescribePlayerSessions"
    ],
    "Resource": "*"
  }
}
```

Configurer un rôle de service IAM pour Amazon GameLift Servers

Certaines Amazon GameLift Servers fonctionnalités nécessitent que vous étendiez un accès limité à d'autres AWS ressources dont vous êtes propriétaire. Vous pouvez le faire en créant un rôle AWS Identity and Access Management (IAM). Un [rôle](#) IAM est une identité IAM que vous pouvez créer dans votre compte et qui dispose d'autorisations spécifiques. Un rôle IAM est similaire à un utilisateur IAM dans la mesure où il s'agit d'une AWS identité dotée de politiques d'autorisation qui déterminent ce que l'identité peut et ne peut pas faire. AWS En revanche, au lieu d'être associé de manière unique à une personne, un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. En outre, un rôle ne dispose pas d'informations d'identification standard à long terme comme un mot de passe ou des clés d'accès associées. Au lieu de cela, lorsque vous adoptez un rôle, il vous fournit des informations d'identification de sécurité temporaires pour votre session de rôle.

Cette rubrique explique comment créer un rôle que vous pouvez utiliser avec vos flottes Amazon GameLift Servers gérées. Si vous utilisez Amazon GameLift Servers FleetIQ pour optimiser l'hébergement de jeux sur vos instances Amazon Elastic Compute Cloud (Amazon EC2), consultez [Configurer votre Compte AWS for Amazon GameLift ServersFleetIQ](#).

Dans la procédure suivante, créez un rôle avec une politique d'autorisation personnalisée et une politique de confiance qui permet Amazon GameLift Servers d'assumer le rôle.

Création d'un rôle de service IAM pour une flotte Amazon GameLift Servers gérée EC2

Étape 1 : créer une politique d'autorisation.

Utilisez les instructions et les exemples de cette page pour créer une politique d'autorisation personnalisée pour le type de Amazon GameLift Servers flotte avec lequel vous travaillez.

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Politiques (Politiques).

Si vous sélectionnez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.
3. En haut de la page, sélectionnez Créer une politique.
4. Dans la section Éditeur de politique, choisissez l'option JSON.
5. Saisissez ou collez un document de politique JSON. Pour de plus amples informations sur le langage de la stratégie IAM, consultez la référence de [politique JSON IAM](#).
6. Résolez les avertissements de sécurité, les erreurs ou les avertissements généraux générés durant la [validation de la politique](#), puis choisissez Suivant.

Note

Vous pouvez basculer à tout moment entre les options des éditeurs visuel et JSON. Toutefois, si vous apportez des modifications ou si vous choisissez Suivant dans l'éditeur visuel, IAM peut restructurer votre politique afin de l'optimiser pour l'éditeur visuel. Pour plus d'informations, consultez la page [Restructuration de politique](#) dans le Guide de l'utilisateur IAM.

7. (Facultatif) Lorsque vous créez ou modifiez une politique dans le AWS Management Console, vous pouvez générer un modèle de stratégie JSON ou YAML que vous pouvez utiliser dans les AWS CloudFormation modèles.

Pour ce faire, dans l'éditeur de politiques, sélectionnez Actions, puis sélectionnez Générer CloudFormation un modèle. Pour en savoir plus AWS CloudFormation, consultez la [référence](#)

[aux types de AWS Identity and Access Management ressources](#) dans le Guide de AWS CloudFormation l'utilisateur.

8. Lorsque vous avez fini d'ajouter des autorisations à la politique, choisissez Suivant.
9. Sur la page Vérifier et créer, tapez un Nom de politique et une Description (facultative) pour la politique que vous créez. Vérifiez les Autorisations définies dans cette politique pour voir les autorisations accordées par votre politique.
10. (Facultatif) Ajoutez des métadonnées à la politique en associant les balises sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la section [Balises pour les AWS Identity and Access Management ressources](#) dans le Guide de l'utilisateur d'IAM.
11. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.

Étape 2 : Créez un rôle qui Amazon GameLift Servers peut assumer.

Pour créer un rôle IAM

1. Dans le volet de navigation de la console IAM, sélectionnez Roles (Rôles), puis Create role (Créer un rôle).
2. Sur la page Sélectionner une entité de confiance, choisissez l'option Politique de confiance personnalisée. Cette sélection ouvre l'éditeur de politique de confiance personnalisée.
3. Remplacez la syntaxe JSON par défaut par la suivante, puis choisissez Next pour continuer.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Sur la page Ajouter des autorisations, recherchez et sélectionnez la politique d'autorisation que vous avez créée à l'étape 1. Choisissez Next (Suivant) pour continuer.
5. Sur la page Nom, révision et création, entrez un nom de rôle et une description (facultatif) pour le rôle que vous créez. Passez en revue les entités de confiance et les autorisations ajoutées.
6. Choisissez Créer un rôle pour enregistrer votre nouveau rôle.

Création d'un rôle IAM pour les conteneurs Amazon GameLift Servers gérés

Si vous utilisez des conteneurs Amazon GameLift Servers gérés, vous devez créer un rôle de service IAM à utiliser avec un parc de conteneurs. Ce rôle accorde des autorisations limitées Amazon GameLift Servers nécessaires pour gérer les ressources de votre flotte de conteneurs et prendre des mesures en votre nom.

Pour créer un rôle IAM pour une flotte de conteneurs

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le volet de navigation de la console IAM, sélectionnez Roles (Rôles), puis Create role (Créer un rôle).
3. Sur la page Sélectionner une entité de confiance, choisissez le AWS service et sélectionnez le cas d'utilisation « GameLift ». Choisissez Next (Suivant)
4. Dans Ajouter des autorisations, choisissez la politique gérée `GameLiftContainerFleetPolicy`. Choisissez Suivant. Voir [AWS politiques gérées pour Amazon GameLift Servers](#) pour plus d'informations sur cette politique.
5. Dans Nom, révision et création, entrez un nom de rôle et choisissez Créer un rôle pour enregistrer le nouveau rôle.

Syntaxe de la politique d'autorisation

- Autorisations Amazon GameLift Servers pour assumer le rôle de service

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
"Effect": "Allow",
"Principal": {
  "Service": "gamelift.amazonaws.com"
},
"Action": "sts:AssumeRole"
}
]
}
```

- Autorisations d'accès aux AWS régions qui ne sont pas activées par défaut

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gamelift.amazonaws.com",
          "gamelift.ap-east-1.amazonaws.com",
          "gamelift.me-south-1.amazonaws.com",
          "gamelift.af-south-1.amazonaws.com",
          "gamelift.eu-south-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Obtenez des outils Amazon GameLift Servers de développement

Amazon GameLift Servers fournit un ensemble d'outils SDKs et d'autres pour vous aider à créer des solutions d'hébergement de jeux pour vos jeux. Ils SDKs ajoutent des fonctionnalités aux serveurs de jeux, aux clients de jeu et aux services principaux qui leur permettent d'interagir avec le Amazon GameLift Servers service. Pour obtenir les dernières informations sur les versions du Amazon GameLift Servers SDK et leur compatibilité, consultez [Notes de mise à jour pour Amazon GameLift Servers](#).

Pour les serveurs de jeux

Intégrez et créez vos serveurs de jeu 64 bits avec le SDK pour serveurs pour Amazon GameLift Servers. Le serveur de jeu utilise le SDK du serveur pour communiquer avec le Amazon GameLift Servers service pour la gestion des sessions de jeu, y compris le démarrage, la mise à jour et l'arrêt des sessions de jeu. Pour obtenir de l'aide sur l'intégration du SDK du serveur dans vos projets de jeu, consultez [Préparation de jeux pour Amazon GameLift Servers](#).

Accompagnement au développement

- Système d'exploitation de développement
 - Windows
 - Linux
- Langages de programmation

[Procurez-vous le Amazon GameLift Servers SDK](#). Pour obtenir des informations spécifiques à la version et des instructions d'installation, consultez les fichiers readme inclus dans chaque package.

- [SDK du serveur C++](#)
 - [Référence du SDK du serveur](#)
 - [Comment intégrer](#)
- [SDK du serveur C#](#) (le support pour .NET 4, .NET 6, .NET 8 varie selon la version, voir) [Versions SDK](#)
 - [Référence du SDK du serveur](#)
 - [Comment intégrer](#)
- [SDK du serveur Go](#)
 - [Référence du SDK du serveur](#)
 - [Comment intégrer](#)
- Support du moteur de jeu

Le plugin complet Amazon GameLift Servers inclut des flux de travail d'interface utilisateur et des exemples de ressources, ainsi que des versions intégrées du SDK et du AWS SDK du serveur. Les flux de travail vous indiquent comment configurer et déployer votre serveur de jeu pour l'héberger avec des EC2 flottes gérées, des flottes de conteneurs gérées ou des flottes Anywhere autogérées. Si vous n'avez pas besoin des flux de travail guidés, vous pouvez uniquement obtenir le SDK du serveur pour votre moteur de jeu à partir des mêmes GitHub référentiels.

Si vous utilisez un autre moteur de jeu ou un autre environnement de développement que le plugin ne prend pas en charge, procurez-vous le SDK du serveur pour votre langage de programmation et ajoutez-le à votre projet de jeu.

Pour obtenir des informations spécifiques à la version et des instructions d'installation, consultez les fichiers readme inclus dans chaque package.

- [Plugin pour Unreal Engine](#) : conçu pour être utilisé avec les versions 5.1, 5.2, 5.3, 5.4, 5.5 et 5.6. Vérifiez les fichiers readme spécifiques à la version pour le support d'Unreal.
 - [Guide des plugins pour Unreal Engine](#)
 - [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
- [Plugin pour Unity](#) : conçu pour être utilisé avec les versions LTS de Unity Editor 6.0, 2022.3 ou 2021.3. Il prend en charge les profils .NET Framework et .NET Standard de Unity, avec .NET Standard 2.1 et .NET 4.x. Consultez les fichiers readme spécifiques à chaque version pour le support Unity.
 - [Guide des plugins pour Unity](#)
 - [???Référence du SDK du serveur C#](#)
- [SDK de serveur pour Unreal](#)
 - [Référence du SDK du serveur](#)
 - [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#)
- [SDK de serveur pour Unity](#)
 - [Référence du SDK du serveur](#)
 - [Intégrer Amazon GameLift Servers dans un projet Unity](#)

Support d'exécution

Pour une solution d'hébergement géré, configurez votre serveur de jeu pour qu'il fonctionne sur l'une des images de machine Amazon suivantes (AMIs). Consultez [Amazon GameLift Servers Versions de l'AMI](#) Amazon GameLift Servers pour plus de détails.

- [Windows Server 2016](#)
- [Amazon Linux 2023](#)
- [Amazon Linux 2](#)

 Note

Le support d'Amazon Linux 2 (AL2) prendra fin le 30 juin 2025. Pour en savoir plus, consultez [Amazon Linux 2 FAQs](#). Pour les serveurs de jeu hébergés AL2 et utilisant le SDK 4.x Amazon GameLift Servers du serveur, mettez d'abord à jour la version du serveur de jeu vers le SDK 5.x, puis déployez-la sur les instances 023. AL2 Consultez [Migrer vers le SDK 5.x du serveur pour Amazon GameLift Servers](#).

Outils supplémentaires

[Wrapper de serveur de jeu pour Amazon GameLift Servers](#)

Cet outil vous aide à déployer un serveur de jeu à héberger avec un ensemble de fonctionnalités de base de gestion des sessions de jeu. Avec cet outil, vous n'avez pas besoin de modifier le code de votre jeu ni d'intégrer le SDK du serveur pour Amazon GameLift Servers. Utilisez le wrapper du serveur de jeu pour emballer votre serveur de jeu et le déployer pour l'hébergement de jeux à l'aide de l'une des trois solutions Amazon GameLift Servers d'hébergement (Anywhere EC2, managed ou managed containers). Cet outil convient parfaitement à l'évaluation précoce ou au prototypage avec votre propre jeu ou un exemple de jeu, car il ne prend pas en charge la personnalisation du serveur de jeu. Si votre jeu n'a pas besoin de fonctionnalités personnalisées, vous pouvez déployer votre serveur de jeu avec le wrapper de serveur de jeu pour l'hébergement de production.

[Amazon GameLift Servers Boîte à outils](#)

Le Amazon GameLift Servers kit d'outils est un ensemble de scripts et d'autres outils que nous avons développés pour aider les développeurs à résoudre des scénarios et des problèmes courants. Le matériel de la boîte à outils comprend des scripts, des exemples de code et des fichiers readme.

- [Kit de démarrage sur les conteneurs](#) : utilisez cet outil pour rationaliser les tâches de configuration des versions de serveurs de jeu destinées à l'hébergement avec des conteneurs Amazon GameLift Servers gérés. Le kit intègre des fonctionnalités essentielles de gestion des sessions de jeu dans un serveur de jeu et utilise des modèles préconfigurés pour créer une flotte de conteneurs et configurer un pipeline de déploiement automatique pour la construction de votre serveur de jeu. Après le déploiement, vous pouvez surveiller les performances de la flotte, gérer les sessions de jeu et analyser les indicateurs à l'aide de la Amazon GameLift Servers console et des outils d'API. Le kit s'intègre à Amazon Simple Storage Service AWS CodeBuild pour le stockage et pour le déploiement de l'infrastructure AWS CloudFormation pour l'automatisation des builds.

- [Outil de mise à jour rapide des builds](#) : utilisez cet outil pour modifier une version de serveur de jeu déjà déployée sur une EC2 flotte gérée. L'outil est conçu pour vous aider à échanger rapidement les fichiers de compilation du jeu sans avoir à configurer et à créer de nouvelles EC2 flottes à chaque modification. Vous pouvez mettre à jour des instances individuelles ou toutes les instances du parc. Les options vous permettent de remplacer des fichiers de build spécifiques ou une version complète, et vous permettent de gérer le redémarrage des serveurs de jeu après les mises à jour.

Pour les services clients liés aux jeux

Créez un service principal 64 bits pour vos clients de jeu à l'aide du AWS SDK, qui inclut l'API de service pour. Amazon GameLift Servers Le service principal de votre jeu gère les interactions côté client avec le Amazon GameLift Servers service, notamment le lancement de nouvelles sessions de jeu et l'inscription des joueurs aux parties.

[Téléchargez le AWS SDK](#)

Pour plus d'informations sur l'utilisation du AWS SDK avec Amazon GameLift Servers, consultez les ressources suivantes :

- [Référence d'API Amazon GameLift Servers](#)
- [Intégration du service client](#)
- [Concevoir un service client principal](#)

Pour la gestion Amazon GameLift Servers des ressources

Utilisez les outils suivants pour créer, mettre à jour et surveiller vos ressources d'hébergement Amazon GameLift Servers géré.

- [AWS Management Console](#) — La AWS console est une application Web qui fournit un accès centralisé à toutes les consoles de AWS service individuelles, y compris Amazon GameLift Servers. Utilisez la console pour créer ou vous connecter à un AWS compte, puis ouvrez la Amazon GameLift Servers console pour utiliser les ressources d'hébergement de vos jeux. Vous pouvez configurer et déployer des flottes d'hébergement et d'autres ressources, consulter les indicateurs d'utilisation et de performance, suivre les ressources dans le tableau de bord et effectuer de nombreuses autres tâches. [Accédez à la Amazon GameLift Servers console.](#)
- [API de service pour Amazon GameLift Servers](#) — Cette API vous donne un accès programmatique à toutes vos Amazon GameLift Servers ressources. Il fait partie du AWS SDK, que vous pouvez

télécharger pour l'utiliser avec les langages de programmation les plus courants. [Procurez-vous le AWS SDK.](#)

- [AWS interface de ligne de commande \(CLI\)](#) — La AWS CLI vous permet d'interagir avec les AWS services à l'aide d'un shell de ligne de commande. Les outils fournissent un accès direct au public APIs pour les AWS services ainsi que des commandes personnalisées disponibles pour un service. [Obtenez la AWS CLI.](#)
- [AWS CloudFormation](#) pour Amazon GameLift Servers — Le AWS CloudFormation service vous aide à modéliser et à configurer les AWS ressources afin de rationaliser le déploiement et la gestion de l'infrastructure. Créez un AWS CloudFormation modèle pour décrire les Amazon GameLift Servers ressources de votre solution d'hébergement, puis utilisez-le pour créer des ressources supplémentaires ou mettre à jour les configurations. Consultez la [référence du type de Amazon GameLift Servers ressource.](#)

Pour Amazon GameLift ServersRealtime

Configurez et déployez Realtime des serveurs pour héberger vos parties multijoueurs. Pour permettre à vos clients de jeu de se connecter aux Realtime serveurs, utilisez le SDK Amazon GameLift Servers Realtime client. Pour commencer, [téléchargez le SDK Realtime client](#). Pour plus d'informations sur la configuration, consultez [Intégration d'un client de jeu pour Amazon GameLift ServersRealtime](#).

Prise en charge de SDK

Le client SDK Realtime contient la source pour les langages suivants :

- C# (.NET)

Environnements de développement

Créez le SDK à partir des sources en fonction des besoins des systèmes d'exploitation de développement et moteurs de jeu pris en charge suivants :

- Systèmes d'exploitation : Windows, Linux, Android, iOS
- Moteurs de jeu — Unity, moteurs compatibles avec les bibliothèques C#

Systèmes d'exploitation de serveur de jeux

Vous pouvez déployer des Realtime serveurs sur des ressources d'hébergement qui s'exécutent sur les plateformes suivantes :

- [Amazon Linux](#)
- [Amazon Linux 2](#)

 Note

AL2 est proche de la fin du support. Pour en savoir plus, consultez [Amazon Linux 2 FAQs](#).

Tutoriel : intégration rapide avec le wrapper Amazon GameLift Servers

Bienvenue dans le didacticiel d'intégration pour Amazon GameLift Servers. Dans ce didacticiel, vous déployez rapidement votre serveur de jeu pour qu'il soit hébergé sur un parc de ressources informatiques basées sur le cloud. Utilisez ce didacticiel pour éviter d'intégrer le SDK du serveur Amazon GameLift Servers dans le code de votre jeu, et déployez plutôt votre jeu avec les fonctionnalités minimales nécessaires pour communiquer avec le Amazon GameLift Servers service et exécuter des sessions de jeu. Vous allez configurer une solution d'hébergement de base et l'utiliser pour profiter de la gamme complète de fonctionnalités telles que le dimensionnement automatique et le matchmaking. C'est également un excellent moyen d'héberger un prototype de votre jeu dans le cadre d'une démo en direct ou à des fins de test.

Principaux avantages de cette méthode d'intégration :

- Déployez rapidement votre serveur de jeu pour un hébergement rapide.
- Aucun changement de code de jeu et aucune modification requise.
- Utilisez cette méthode avec n'importe quel exécutable de jeu, quel que soit le moteur de jeu.
- Explorez tous les outils Amazon GameLift Servers de gestion, y compris la surveillance de l'activité des sessions de jeu et de l'état de santé de l'hébergement.

 Note

L'emballage est destiné à l'évaluation et à une utilisation de base en production. Les fonctionnalités avancées telles que la gestion détaillée des sessions des joueurs nécessitent une intégration complète du SDK au serveur.

Prérequis

Avant de commencer, assurez-vous d'avoir :

- Et Compte AWS avec les autorisations appropriées
- AWS CLI installé
- Go 1,18 et plus
- Un serveur de jeu multijoueur exécutable
- Marque (Linux/Mac)
- Git installé avec un compte actif

Présentation

Dans ce didacticiel, vous allez :

1. Obtenez et créez le wrapper
2. Préparez le build du jeu
3. Configurer le wrapper
4. Téléchargez le build du serveur de jeu
5. Création de la EC2 flotte gérée
6. Créer une session de jeu et s'y connecter
7. Surveillez et gérez vos serveurs de jeu
8. Faites évoluer vos serveurs de jeu

Étape 1 : Obtenir et créer le wrapper du serveur de jeu

Utilisez les commandes suivantes pour obtenir la source du wrapper du serveur de jeu et créer le wrapper. Ces commandes utilisent SSH, mais vous pouvez également accéder directement au dépôt Github.

Windows

```
> git clone git@github.com:amazon-gamelift/amazon-gamelift-servers-game-server-wrapper.git
> cd amazon-gamelift-servers-game-server-wrapper
> powershell -file .\build.ps1
```

Mac et Linux

```
$ git clone git@github.com:amazon-gamelift/amazon-gamelift-servers-game-server-wrapper.git
$ cd amazon-gamelift-servers-game-server-wrapper
$ make
```

En cas de compilation réussie, un répertoire « out » est ajouté à `amazon-gamelift-servers-game-server-wrapper`. Ce répertoire contient trois dossiers, un pour chaque option de parc d'hébergement prise en charge, qui contiennent un ensemble d'artefacts de build. Dans le cadre de ce didacticiel, vous déployez sur EC2 des flottes gérées. Vous allez donc utiliser le dossier `gamelift-servers-managed-ec2`.

Étape 2 : Préparez la version de votre serveur de jeu

Au cours de cette étape, vous préparez les fichiers de build de votre serveur de jeu pour les télécharger sur le Amazon GameLift Servers.

Créez le répertoire du jeu

Maintenant, préparez un répertoire de jeu sur votre machine locale. Ce répertoire doit contenir tous les fichiers dont vous avez besoin pour faire fonctionner votre serveur de jeu Amazon GameLift

Servers. Cela inclut le wrapper du serveur de jeu, le build de votre serveur de jeu et le fichier `config.yaml` qui permet au wrapper de fonctionner avec votre serveur de jeu.

Procédez comme suit :

1. EC2 Flotte gérée. Dans le dossier de sortie du wrapper du serveur de jeu, recherchez l'artefact de build à déployer dans une flotte gérée EC2 . Le build sera écrit dans un répertoire de sortie comme indiqué ici : `out\linux\amd64\gamelift-servers-managed-ec2`.
2. Copiez le fichier exécutable de votre serveur de jeu et tous les fichiers associés dont il a besoin pour fonctionner dans le `gamelift-servers-managed-ec2` dossier. Vous pouvez avoir des répertoires imbriqués selon vos besoins.

Voici un exemple de structure de répertoire :

```
gamelift-servers-managed-ec2
#-- config.yaml
#-- amazon-gamelift-servers-game-server-wrapper
#-- MyGame
# #-- server-executable.exe
# #-- my-game-settings
# # .....
```

Étape 3 : configurer le wrapper pour votre flotte

Amazon GameLift Servers gère le cycle de vie des instances de calcul d'un parc, en créant de nouvelles instances lorsque la version de votre serveur est installée et en recyclant les instances selon les besoins. Le service gère le cycle de vie des processus du serveur de jeu qui s'exécute sur chaque instance. Une EC2 flotte gérée peut avoir des instances situées à plusieurs endroits pour soutenir les joueurs où qu'ils se trouvent.

Modifiez le fichier `config.yaml` pour configurer le wrapper pour la journalisation, la configuration des ports et l'initialisation du serveur.

1. Configurez les paramètres de journalisation. Le wrapper du serveur de jeu génère des messages de journal pour chaque processus du serveur de jeu. Par défaut, le niveau de journalisation est défini sur `debug` pour une verbosité maximale. Cela est très utile lors de la configuration et du

- dépannage et détermine le niveau de détail des messages du journal, dans ce cas, les plus détaillés. Les options incluent debug, info, warn et error (les moins détaillées).
2. Spécifiez le chemin d'accès au répertoire des journaux du serveur de jeu. Le chemin par défaut pour les journaux du serveur de jeu est `./game-server-logs`. Ce répertoire contient tous les journaux générés par votre serveur de jeu et chaque instance les possède. Les journaux sont automatiquement téléchargés Amazon GameLift Servers là où ils sont accessibles depuis l'onglet Événements. Consultez la section Dépannage pour plus de détails.
 3. Définissez la configuration des ports réseau. Réglez le port de jeu comme vous le souhaitez. Pour ce didacticiel, spécifiez un seul port, car vous allez créer une flotte qui n'exécutera qu'un seul processus de serveur de jeu simultané par instance. Si vous décidez d'exécuter plusieurs processus à la fois, vous devez configurer suffisamment de ports pour chaque processus simultané. La valeur par défaut est 37016, comme indiqué dans le fichier de configuration, mais en général, pour les flottes utilisant des versions Linux, utilisez les ports 22 et 1026-60000. Pour les flottes utilisant des versions Windows, utilisez les ports 1026 à 60000.
 4. Configurez le chemin d'accès à l'exécutable du serveur de jeu. Pour `./MyGame/my-server-executable.exe` personnaliser le chemin d'accès à l'exécutable de votre serveur de jeu avec son nom et son emplacement réels. C'est le point d'entrée pour lancer votre serveur de jeu.
 5. Configurez les arguments du serveur de jeu. Spécifiez au minimum un argument `-port` et utilisez la même valeur de port de jeu que celle que vous avez définie auparavant. La valeur « pos » 0 indique qu'il s'agit du premier argument. Ajoutez d'autres arguments si nécessaire. Ces arguments sont transmis à votre serveur de jeu lors de son lancement, ce qui vous permet de configurer son comportement d'exécution.
 - a. Argument : `--port`
 - b. Valeur : `"{{.port number here}}"`
 - c. Position : 0 (Premier argument de la liste)

Exemple de configuration :

```
log-config:
  wrapper-log-level: debug
game-server-logs-dir: ./game-server-logs
ports:
  gamePort: 37016
game-server-details:
  executable-file-path: ./MyGame/my-server-executable
```

```
game-server-args:
  - arg: "--port"
    val: "{{.gameport}}"
    pos: 0
```

Étape 4 : Téléchargez le build du serveur de jeu

Vous avez maintenant terminé tous les éléments nécessaires à la construction de votre serveur de jeu (wrapper du serveur de jeu, config.yaml et fichiers de votre serveur de jeu), et vous êtes prêt à télécharger votre build de jeu pour l'héberger. Amazon GameLift Servers Le moyen le plus rapide de télécharger votre build de jeu est d'utiliser la AWS CLI commande `upload-build` illustrée dans l'exemple suivant.

Téléchargement d'une version de jeu sous Windows :

```
aws gamelift upload-build \
  --name gamelift-test-2025-03-11-1 \
  --build-version gamelift-test-2025-03-11-1 \
  --build-root out/windows/amd64/gamelift-servers-managed-ec2 \
  --operating-system WINDOWS_2016 \
  --server-sdk-version 5.3.0 \
  --region us-west-2
```

Note

Pour les versions Mac et Linux, utilisez `--operating-system AMAZON_LINUX_2023`

Lorsque vous créez le build, enregistrez l'ID du build issu de la réponse de l'API pour l'utiliser pour créer le parc de véhicules.

Étape 5 : Création de la EC2 flotte gérée

Les étapes suivantes décrivent une configuration de flotte minimale, afin que vous puissiez être opérationnel dès que possible.

Pour créer votre flotte :

1. Connectez-vous au AWS Management Console et naviguez vers Amazon GameLift Servers.
2. Dans la barre de menu en haut de la fenêtre de la console, vérifiez dans quelle région se trouve votre build. Prenez note de cela, car votre flotte doit se trouver dans la même région, sinon vous ne pourrez ni trouver ni choisir votre modèle.
3. Dans le volet de navigation de la EC2 section Managed, choisissez Builds.
4. Sélectionnez le build que vous avez chargé précédemment pour afficher la page des détails du build.
5. Dans la section Flottes, choisissez Créer une flotte qui affiche la page Définir les détails de la EC2 flotte gérée à partir de laquelle vous pouvez suivre l'état de votre flotte et consulter les événements de création de flotte dans l'onglet Événements.
6. Renseignez le nom et la description, puis choisissez Next.
7. Sur la page Définir les détails de l'instance, la région du build est affichée par défaut. Choisissez les régions supplémentaires que vous souhaitez ajouter.
8. Pour le type de flotte, choisissez On-Demand.
9. Sous Types d'instances, choisissez c5.large, puis Next.
10. Dans Configuration d'exécution, étant donné que la version de jeu que vous avez téléchargée utilise le wrapper, vous devez plutôt pointer vers l'exécutable du wrapper. Pour les serveurs de jeux Windows, c'est le `casC:\game\amazon-gamelift-servers-game-server-wrapper.exe`. Pour les serveurs de jeux Linux, c'est le `cas/local/game/amazon-gamelift-servers-game-server-wrapper`.

Par exemple : `LaunchPath": "/local/game/amazon-gamelift-servers-game-server-wrapper", "ConcurrentExecutions": 1, "Parameters": "--port 37016`

Configurez également les valeurs des ports de jeu avec une plage qui tient compte du port défini dans les paramètres de lancement de la configuration d'exécution `config.yaml` et de la configuration d'exécution. Le port d'entrée de `config.yaml` ne doit pas nécessairement correspondre à celui spécifié dans la configuration d'exécution de la flotte, mais au moment de l'exécution, si la configuration spécifie un port différent, cette valeur remplace celle du `config.yaml`. Les paramètres de lancement saisis dans la configuration d'exécution remplacent également ceux figurant dans le `config.yaml`.

11. Sur la page Réviser et créer, vérifiez toutes les configurations, puis choisissez Soumettre pour créer votre flotte. L'état de la flotte changera au fur et à mesure que la capacité d'hébergement du serveur de jeu augmentera, et elle affichera très bientôt le statut Actif. Une fois l'activation terminée

et la flotte déployée, le service lance le wrapper qui est prêt à recevoir une demande de session de jeu.

Étape 6 : créer une session de jeu et s'y connecter

Lorsque le statut de votre flotte indique Actif, cela signifie que les serveurs de jeu sont prêts à accueillir une session de jeu. Pour démarrer une session de jeu, faites une demande de session de jeu auprès du Amazon GameLift Servers service. Ici, vous allez utiliser le AWS CLI pour faire cette demande.

Note

N'oubliez pas que la création d'une session de jeu à l'aide du AWS CLI est utile pour tester et vous familiariser avec le processus. À un moment donné, vous ajouterez des appels de AWS SDK programmatiques au service principal de votre jeu dans le cadre du matchmaking ou du système de placement de sessions de jeu de votre jeu.

Utilisez ce qui suit pour créer une session de jeu :

```
aws gamelift create-game-session \  
--fleet-id <FLEET_ID> \  
--game-properties '[{"Key": "exampleProperty", "Value": "exampleValue"}]' \  
--maximum-player-session-count 3 \  
--region us-west-2
```

Vous pouvez également transmettre des propriétés de jeu personnalisées à l'exécutable de votre serveur. Consultez les arguments du serveur de jeu dans le fichier README pour plus de détails. Lorsqu'il reçoit l' create-game-session appel, il demande au Amazon GameLift Servers wrapper de lancer le fichier exécutable du serveur de jeu et de démarrer une session de jeu. Ce qu'il contient config.yaml affecte la configuration du serveur de jeu, et les paramètres de lancement définis dans la console déterminent la configuration de la session de jeu elle-même.

Exemple de format pour ajouter des propriétés de jeu :

```
defaultArgs:
```

```
- arg: "--port"
  val: "{{.GamePort}}"
  pos: 0
- arg: "--ipAddress"
  val: "{{.IpAddress}}"
  pos: 1
- arg: "--gameSessionId"
  val: "{{.GameSessionId}}"
  pos: 2
```

Étape 7 : Gérez et surveillez votre flotte

Maintenant que votre parc de serveurs de jeu est configuré et qu'une session de jeu est lancée, vous pouvez le gérer et le surveiller depuis la Amazon GameLift Servers console. La meilleure façon d'y parvenir est d'accéder à la page des détails de la flotte, où vous pouvez modifier les détails de la flotte ou modifier l'échelle et la capacité de la flotte dans l'onglet Mise à l'échelle. Consultez la section suivante sur le dimensionnement de vos serveurs de jeu.

Choisissez l'onglet Métriques pour afficher des graphiques illustrant la surveillance des métriques d'activité et du matériel. Pour plus de détails sur les graphiques de mesures, cliquez sur le lien Info situé à côté de l'identifiant de votre flotte. En outre, vous pouvez surveiller de près vos serveurs de jeu depuis l'onglet Mesures, mais vous pouvez également ajouter des alarmes à ces statistiques dans le CloudWatch tableau de bord.

Pour accéder au CloudWatch tableau de bord depuis la console :

1. Tapez « CloudWatch » dans la barre de recherche et choisissez-le dans la liste des résultats de recherche pour afficher l' CloudWatch aperçu.
2. Faites défiler la page vers le bas et choisissez Afficher le GameLift tableau de bord pour afficher des graphiques basés sur des indicateurs clés relatifs à vos flottes et à vos sessions de jeu.

Étape 8 : redimensionnez vos serveurs de jeu

À l'étape suivante, vous allez configurer le dimensionnement automatique. La mise à l'échelle automatique vous permet d'adapter de manière dynamique la capacité de votre flotte en fonction de l'activité du serveur de jeu. Au fur et à mesure que les joueurs arrivent et commencent des sessions de jeu, le scalage automatique ajoute de nouvelles instances, et à mesure que la demande des joueurs diminue, le scalage automatique libère les instances inutilisées. Il s'agit d'un moyen efficace

de minimiser vos ressources et vos coûts d'hébergement tout en offrant une expérience de jeu fluide et rapide.

Lorsque vous vous préparez au lancement du jeu, vous devez configurer le dimensionnement automatique de vos flottes. La mise à l'échelle automatique est recommandée comme moyen efficace de minimiser vos ressources et vos coûts d'hébergement tout en offrant une expérience de jeu fluide et rapide.

Pour configurer manuellement la capacité du parc

Pour configurer manuellement la capacité du parc

1. Accédez à l'onglet Scaling sur la page de détails de votre flotte.
2. Sélectionnez un lieu, puis cliquez sur Modifier.
3. Modifiez la valeur des instances souhaitées et modifiez les paramètres de taille minimale et maximale afin de dépasser leurs valeurs actuelles, puis choisissez Confirmer.

Note

Utilisez le paramètre maximum d'instances comme limite pour éviter une mise à l'échelle et des coûts excessifs.

Pour utiliser la mise à l'échelle automatique basée sur les cibles

Pour utiliser la mise à l'échelle automatique basée sur les cibles

La mise à l'échelle automatique basée sur les cibles (suivi des cibles) lie la mise à l'échelle de la flotte au pourcentage de sessions de jeu disponibles. Au fur et à mesure que les joueurs se pressent pour jouer à votre jeu et que le nombre de sessions de jeu disponibles diminue, celui-ci réagit en ajoutant automatiquement de nouvelles instances à la flotte.

1. Sous Politique de mise à l'échelle automatique basée sur Target, choisissez Ajouter une politique et définissez la capacité de la flotte pour qu'elle change automatiquement lorsqu'elle atteint le seuil du pourcentage de sessions de jeu disponibles que vous avez défini. Une plus grande mémoire tampon permet de mieux gérer les surtensions, ce qui permet aux nouveaux joueurs de se lancer rapidement dans les jeux, mais cela peut également entraîner une augmentation des coûts d'hébergement.

2. Choisissez Confirmer pour accepter les modifications.

La mise à l'échelle automatique basée sur des règles vous donne un contrôle plus précis, notamment la possibilité de lier la mise à l'échelle à d'autres indicateurs de flotte et de définir des seuils personnalisés et des réponses de dimensionnement. Il propose des options puissantes, mais il nécessite également l'utilisation de la CLI et des tests approfondis pour comprendre le comportement des règles personnalisées en action. Ce didacticiel se concentre d'abord sur la configuration de l'approche basée sur les cibles.

Résoudre les problèmes courants

Voici les problèmes courants que vous pouvez rencontrer lors du fonctionnement de vos serveurs de jeu et de vos sessions de jeu. Lorsque le serveur ou la session de jeu ne fonctionnent pas correctement, la première étape consiste à consulter les journaux, ce qui peut révéler l'un des problèmes décrits ci-dessous concernant les nouveaux déploiements ou les jeux en production.

Les informations suivantes sont souvent révélées dans les journaux :

- Le processus du serveur de jeu ne peut pas démarrer. Il s'agit peut-être d'une erreur dans la configuration du wrapper. Vérifiez que le fichier possède le bon chemin de lancement et que les paramètres et arguments de lancement sont corrects.
- La version du serveur de jeu n'est pas exécutable. Il s'agit probablement d'une erreur dans le code du jeu.
- Les joueurs ne peuvent pas se connecter aux sessions de jeu. Il s'agit probablement d'une erreur de configuration du port.
- Retard ou lenteur des connexions. Passez en revue les politiques et les seuils de dimensionnement.
- Aucune connexion. Vérifiez les règles et la configuration des ports pour votre flotte.

Pour consulter les journaux d'événements de votre Amazon GameLift Servers flotte

Pour consulter les journaux d'événements de votre Amazon GameLift Servers flotte

1. Ouvrez la Amazon GameLift Servers console .
2. Sur la page Détails de la flotte, choisissez l'onglet Événements et téléchargez le journal. Vous pouvez également surveiller l'activité et les statistiques matérielles relatives à l'état du serveur de jeu et aux activations de sessions de jeu depuis l'onglet Mesures.

Pour consulter les journaux des sessions de jeu

Pour consulter les journaux des sessions de jeu

1. Depuis la console, ouvrez votre flotte et ouvrez l'onglet Sessions de jeu.
2. Choisissez un identifiant de session de jeu dans la liste pour afficher la page de présentation correspondante.
3. Choisissez Télécharger les journaux pour télécharger un fichier journal localement.

Pour consulter les journaux des sessions de jeu avec la CLI, utilisez l'GetGameSessionLogURLAPI. Amazon GameLift Servers stocke automatiquement les journaux pendant 14 jours.

Vous pouvez également configurer les CloudWatch journaux Amazon pour votre flotte. Cela fournit des fonctionnalités de journalisation supplémentaires et une intégration avec d'autres services AWS de surveillance.

Pour un accès aux journaux en temps réel ou des périodes de conservation prolongées grâce à CloudWatch :

1. En haut du tableau de bord de votre Amazon GameLift Servers console, tapez « CloudWatch » dans la barre de recherche et choisissez-le dans le menu déroulant des résultats.
2. Accédez aux groupes de CloudWatch journaux et recherchez des sessions spécifiques. La méthode la plus simple consiste à cliquer sur Rechercher tout et à filtrer à l'aide gameSessionId de ClientID.

Étapes suivantes

- [Créez une flotte multi-sites pour ajouter de l'hébergement dans des sites supplémentaires](#)
- [Ajoutez une file d'attente pour les sessions de jeu afin de trouver les meilleurs emplacements possibles pour les sessions de jeu sur plusieurs sites](#)
- [Expérimentez le FlexMatch matchmaking en créant un entremetteur et un ensemble de règles pour votre jeu](#)
- [Commencez à travailler sur les fonctionnalités de votre client de jeu et des composants du service principal, afin que les joueurs puissent faire des demandes d'inscription et se connecter directement aux sessions de jeu](#)
- [Lorsque vous êtes prêt, passez à une solution entièrement intégrée](#)

Explorez avec le Amazon GameLift Servers .

Le Amazon GameLift Servers Le plugin est un module complémentaire complet pour votre moteur de jeu Unreal ou Unity. Il vous guide à travers les étapes de base du déploiement de votre jeu pour l'hébergement avec Amazon GameLift Servers. Grâce à l'ensemble d'outils et aux flux de travail du plugin, vous pouvez travailler dans votre environnement de développement de moteur de jeu pour préparer votre serveur de jeu à l'hébergement, configurer l'hébergement sur une machine locale pour les tests, créer un service principal simple et déployer votre serveur de jeu sur un hébergement géré dans le cloud.

Utilisez le plugin pour découvrir comment travailler avec Amazon GameLift Servers et mettez rapidement en place une solution d'hébergement de jeux. Vous pouvez travailler avec des exemples de ressources de jeu ou avec votre propre projet de jeu. Le plugin automatise un certain nombre d'étapes afin que vous puissiez créer rapidement une solution de travail simple. Lorsque vous aurez terminé les flux de travail guidés du plugin, vous pourrez connecter un client de jeu à des sessions de jeu hébergées en direct via Amazon GameLift Servers. Après avoir utilisé le plugin pour créer une solution d'hébergement simple, vous pouvez personnaliser votre solution pour répondre aux besoins de votre jeu.

Le plugin est disponible pour les moteurs de jeu suivants :

- Unreal Engine
- Unity

Le plugin inclut les composants suivants pour chaque moteur de jeu :

- Modules de plug-in pour l'éditeur du moteur de jeu. Lorsque le plugin est installé, un nouveau bouton du menu principal vous donne accès à Amazon GameLift Servers fonctionnalité.
- Bibliothèques pour Amazon GameLift Servers API de service avec fonctionnalité côté client.
- Bibliothèques pour Amazon GameLift Servers SDK du serveur (version 5).
- Exemples de ressources à utiliser pour tester l'intégration d'un serveur.
- Configurations modifiables, sous forme de AWS CloudFormation modèles, qui définissent la solution de votre serveur de jeu.

Rubriques

- [Flux de travail du plugin](#)

- [Amazon GameLift Serversplugin pour Unreal Engine](#)
- [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#)
- [Amazon GameLift Serversplugin pour Unity \(SDK de serveur 4.x\)](#)

Flux de travail du plugin

Les étapes suivantes décrivent un processus typique de préparation et de déploiement de votre projet de jeu sur Amazon GameLift Servers. Vous devez effectuer ces étapes en travaillant dans l'éditeur du moteur de jeu et en utilisant votre code de jeu.

1. Créez un profil utilisateur qui renvoie à l'utilisateur de votre AWS compte et fournit des informations d'accès avec des autorisations d'utilisation Amazon GameLift Servers.
2. Configurez les AWS ressources associées que le plugin utilise dans la solution d'hébergement (appelées « bootstrapping »).
3. Ajoutez du code serveur à votre projet pour établir une communication entre un serveur de jeu en cours d'exécution et le Amazon GameLift Servers service.
4. Ajoutez un code client à votre projet qui permet aux clients du jeu d'envoyer des demandes à Amazon GameLift Servers pour démarrer de nouvelles sessions de jeu, puis s'y connecter.
5. Utilisez le flux de travail Anywhere pour configurer votre station de travail locale en tant que serveur de calcul Anywhere et héberger votre serveur de jeu. Lancez votre serveur de jeu et votre client localement via le plugin, connectez-vous à une session de jeu et testez l'intégration.
6. Utilisez le EC2 flux de travail géré pour télécharger votre serveur de jeu sur Amazon GameLift Servers et déployez une solution d'hébergement cloud simple mais complète. Lancez votre client de jeu localement via le plugin, demandez une session de jeu, connectez-vous à celui-ci et jouez à votre jeu.

Lorsque vous travaillez dans le plugin, vous allez créer et utiliser AWS des ressources. Ces actions peuvent entraîner des frais pour le AWS compte utilisé. Si vous êtes nouveau dans ce AWS domaine, ces actions peuvent être couvertes par le [niveau AWS gratuit](#).

Amazon GameLift Serversplugin pour Unreal Engine

Ce plugin ajoute le SDK et les outils du serveur Amazon GameLift Servers C++ à l'éditeur UE. Utilisez les flux de travail guidés de l'interface utilisateur pour intégrer les fonctionnalités du SDK du

serveur dans votre projet de jeu et déployer une solution Amazon GameLift Servers d'hébergement pour votre serveur de jeu.

Avec le plugin, vous pouvez créer une solution d'hébergement fonctionnelle de base, puis l'optimiser et la personnaliser selon vos besoins. Configurez une flotte Amazon GameLift Servers Anywhere avec votre poste de travail local comme hôte. Pour un hébergement dans le cloud avec des flottes de conteneurs gérés EC2 ou gérés, déployez votre serveur de jeu avec une solution complète pour gérer les demandes de sessions de jeu et les connexions clients.

Rubriques

- [Installez le plugin pour votre projet de jeu Unreal](#)
- [Prochaines étapes : Personnalisez votre solution d'hébergement de jeux](#)
- [Plugin pour Unreal : configurer un profil AWS utilisateur](#)
- [Plugin pour Unreal : intégrez le code de votre jeu](#)
- [Plugin pour Unreal : hébergez votre jeu localement avec Anywhere Amazon GameLift Servers](#)
- [Plugin pour Unreal : déployez votre jeu sur une flotte gérée EC2](#)
- [Plugin pour Unreal : déployez votre jeu sur une flotte de conteneurs gérée](#)

Installez le plugin pour votre projet de jeu Unreal

[Téléchargez le Amazon GameLift Servers plugin pour Unreal Engine sur GitHub](#)

Consultez le fichier readme du GitHub dépôt pour savoir comment installer le plugin dans votre éditeur Unreal pour un projet de jeu.

Le plugin inclut les composants suivants :

- Modules de plug-in pour l'éditeur UE. Lorsque le plugin est installé, un nouveau bouton du menu principal vous donne accès aux Amazon GameLift Servers fonctionnalités.
- bibliothèques C++ pour l'API Amazon GameLift Servers de service. Utilisez les fonctionnalités de l'API dans un service principal côté client pour aider les clients de jeux à demander des sessions de jeu et des informations sur les sessions de send/retrieve jeu.
- Bibliothèques Unreal pour le SDK Amazon GameLift Servers du serveur (version 5). Utilisez le SDK du serveur dans le code de votre serveur de jeu pour gérer la communication entre les processus du serveur de jeu hébergé et le Amazon GameLift Servers service.

- Contenu destiné aux tests, notamment une carte de jeu de démarrage et deux cartes de test avec des plans de base et des éléments d'interface utilisateur à utiliser pour tester l'intégration d'un serveur.
- Configurations modifiables, sous forme de AWS CloudFormation modèles, que le plugin utilise lors du déploiement de votre serveur de jeu pour l'hébergement.

Ce plugin utilise des AWS CloudFormation modèles pour déployer des solutions d'hébergement pour des scénarios de jeu courants. Utilisez ces solutions telles que fournies ou personnalisez-les selon les besoins de vos jeux.

Prochaines étapes : Personnalisez votre solution d'hébergement de jeux

L'utilisation des flux de travail guidés du plugin est un bon moyen d'être rapidement opérationnel avec une solution Amazon GameLift Servers d'hébergement. Avec le plugin, vous pouvez configurer les versions de base de chacun des composants de votre solution.

Lorsque vous serez prêt, vous pourrez vous appuyer sur cette solution de base en personnalisant chaque composant et en peaufinant votre solution avant le lancement du jeu. Envisagez les options suivantes :

- Modifiez vos flottes et leurs configurations. Consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Personnalisez la configuration de la file d'attente de votre session de jeu. Voir [Personnaliser une file d'attente de session de jeu](#) :
- Ajoutez des fonctionnalités à votre serveur de jeu et à votre client de jeu. Consultez [Intégrez votre serveur de jeu avec Amazon GameLift Servers](#) et [Intégrez votre client de jeu à Amazon GameLift Servers](#).
- Personnalisez votre service principal. Consultez [Concevez le service client de votre jeu](#).
- Configurez le dimensionnement automatique des capacités pour répondre à la demande attendue des joueurs. Consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).
- Configurez les outils d'observabilité de l'hébergement, notamment les outils d'analyse et de journalisation. Consultez [Surveillance des Amazon GameLift Servers](#).
- Automatisez votre déploiement à l'aide de l'[infrastructure en tant que code \(IaC\)](#). Les flux de travail guidés du plugin pour les solutions gérées utilisent des AWS CloudFormation modèles.

Vous pouvez les personnaliser selon vos besoins. Consultez [Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation](#).

Rubriques

- [Plugin pour Unreal : configurer un profil AWS utilisateur](#)
- [Plugin pour Unreal : intégrez le code de votre jeu](#)
- [Plugin pour Unreal : hébergez votre jeu localement avec Anywhere Amazon GameLift Servers](#)
- [Plugin pour Unreal : déployez votre jeu sur une flotte gérée EC2](#)
- [Plugin pour Unreal : déployez votre jeu sur une flotte de conteneurs gérée](#)

Plugin pour Unreal : configurer un profil AWS utilisateur

Après avoir installé le plugin, configurez un profil utilisateur avec un AWS compte valide. Vous pouvez gérer plusieurs profils dans le plugin, mais vous ne pouvez sélectionner qu'un seul profil à la fois. Chaque fois que vous travaillez dans le plugin, sélectionnez un profil à utiliser. Chaque page de flux de travail affiche le profil actuellement sélectionné.

La gestion de plusieurs profils vous permet de passer d'un déploiement d'hébergement à un autre. Par exemple, vous pouvez configurer des profils qui utilisent le même AWS compte mais qui sont déployés AWS dans différentes régions. Vous pouvez également configurer des profils avec différents AWS comptes ou utilisateurs et ensembles d'autorisations.

Note

Si vous avez installé la AWS CLI sur votre poste de travail et qu'un profil est déjà configuré, le Amazon GameLift Servers plugin le détectera et le répertoriera comme profil existant. Le plugin sélectionne automatiquement n'importe quel profil nommé[default]. Vous pouvez utiliser un profil existant ou en créer un nouveau.

Tous les profils doivent être initialisés pour configurer certaines AWS ressources requises sous l'utilisateur de votre compte.

Pour gérer vos AWS profils

1. Dans la barre d'outils principale de l'éditeur Unreal, choisissez le Amazon GameLift Servers menu, puis sélectionnez AWS Access Credentials. Cette action ouvre le Amazon GameLift Servers plugin sur la page Configurer votre profil utilisateur.
2. Utilisez les boutons pour créer un nouveau AWS compte ou configurer un profil utilisateur pour un AWS compte que vous possédez déjà.
3. Si vous n'avez pas encore de profil utilisateur, vous êtes invité à saisir les informations du profil et à créer un nouveau profil. Saisissez les informations suivantes :
 - Un AWS compte. Si vous créez un nouveau AWS compte, utilisez le lien vers le AWS Management Console et suivez les instructions. Voir [Créer un AWS compte](#) pour plus de détails.
 - Un AWS utilisateur autorisé à utiliser Amazon GameLift Servers les autres AWS services requis. Consultez les instructions [Configurez un Compte AWS](#) relatives à la configuration d'un utilisateur AWS Identity and Access Management (IAM) avec des Amazon GameLift Servers autorisations et un accès programmatique avec des informations d'identification à long terme.
 - Informations d'identification de votre AWS utilisateur. Ces informations d'identification se composent d'un identifiant de clé d' AWS accès et d'une clé AWS secrète. Consultez [Obtenir vos clés d'accès](#) pour plus de détails.
 - AWS région. Il s'agit d'un emplacement géographique dans lequel vous souhaitez créer vos AWS ressources pour l'hébergement. Pendant le développement, nous vous recommandons d'utiliser une région proche de votre emplacement physique. Choisissez une région dans la liste des [AWS régions prises en charge](#).
4. Si le plugin détecte des profils existants, il affiche une liste des profils disponibles. Sélectionnez un profil existant dans la liste ou choisissez Ajouter un autre profil pour en créer un nouveau.

Démarrez un profil utilisateur

Tous les profils doivent être amorcés pour être utilisés avec le Amazon GameLift Servers plugin. Le bootstrapping crée un compartiment Amazon S3 spécifique au profil. Il est utilisé pour stocker les configurations de projet, créer des artefacts et d'autres dépendances. Les buckets ne sont pas partagés entre d'autres profils.

Le bootstrapping implique la création de nouvelles AWS ressources et peut entraîner des coûts.

Pour démarrer votre profil :

1. Sur la page Informations d'identification d'AWS accès, vérifiez l'état d'amorçage du profil utilisateur que vous souhaitez utiliser. Si le statut d'amorçage du profil est « Inactif » et qu'aucun compartiment S3 n'est répertorié, vous devez démarrer le profil.
2. Sélectionnez le profil que vous souhaitez utiliser et choisissez le profil Bootstrap.
3. Attendez que le statut du bootstrap passe à « Actif ». Cette opération peut prendre quelques minutes.

Plugin pour Unreal : intégrez le code de votre jeu

Avant de pouvoir déployer votre serveur de jeu dans une flotte, vous devez apporter une série de mises à jour au code du jeu et emballer les composants du jeu à utiliser avec le Amazon GameLift Servers service.

Cette rubrique décrit les étapes à suivre pour effectuer une intégration minimale. Pour l'intégration au serveur, utilisez l'exemple de code fourni pour mettre à jour le mode de jeu de votre projet.

- [Configurer les cibles de construction et les règles du module](#)
- [Mettez à jour le code de votre serveur de jeu](#)
- [Intégrez la carte de jeu de votre client](#)
- [Package des composants de votre jeu](#)

Configurer les cibles de construction et les règles du module

Modifiez les fichiers de votre projet de jeu pour générer correctement les composants de build à utiliser avec Amazon GameLift Servers.

Pour ajouter des cibles de build pour le client et le serveur :

1. Ouvrez les fichiers de code de votre projet de jeu et localisez le `.../Games/[your application name]Source/[your application name]Target.cs` fichier. Exemple: `.../Source/GameLiftUnrealAppTarget.cs`. (Si vous utilisez Visual Studio, ouvrez le `.sln` fichier du projet.)
2. Copiez ce fichier pour créer deux nouveaux fichiers cibles dans le `Source/` répertoire.

- Client cible : renommez le nouveau fichier en *[your application name]*Client.Target.cs. Modifiez le contenu pour mettre à jour le nom de classe et les valeurs du type de cible, comme illustré dans l'exemple de code suivant :

```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppClientTarget : TargetRules
{
    public GameLiftUnrealAppClientTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Client;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp" );
    }
}
```

- Cible du serveur : renommez le nouveau fichier en *[your application name]*Server.Target.cs. Modifiez le contenu pour mettre à jour le nom de classe et les valeurs du type de cible, comme illustré dans l'exemple de code suivant :

```
using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppServerTarget : TargetRules
{
    public GameLiftUnrealAppServerTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Server;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp" );
    }
}
```

3. Régénérez vos fichiers de projet. Si vous utilisez Visual Studio, vous pouvez cliquer avec le bouton droit sur le .uproject fichier de votre projet de jeu et sélectionner Générer des fichiers de projet Visual Studio.

Pour mettre à jour les règles du module de projet de jeu :

Mettez à jour les règles du module du projet de jeu pour qu'il devienne dépendant du plugin.

1. Ouvrez les fichiers de code de votre projet de jeu et localisez le `.../Games/[your application name]/Source/[your application name].Build.cs` fichier. Exemple: `.../Source/GameLiftUnrealApp.Build.cs`. (Si vous utilisez Visual Studio, ouvrez le `.sln` fichier du projet.)
2. Localisez la `ModuleRules` classe et mettez-la à jour comme illustré dans l'exemple de code suivant :

```
using UnrealBuildTool;

public class GameLiftUnrealApp : ModuleRules
{
    public GameLiftUnrealApp ( ReadOnlyTargetRules Target ) : base ( Target )
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
        PublicDependencyModuleNames.AddRange( new string[] { "Core",
"CoreUObject", "Engine", "InputCore", "HeadMountedDisplay",
"EnhancedInput" });
        // Add the following section
        if (Target.Type == TargetType.Server)
        {
            PublicDependencyModuleNames.Add("GameLiftServerSDK");
        }
        else
        {
            PublicDefinitions.Add("WITH_GAMELIFT=0");
        }
        bEnableExceptions = true;
    }
}
```

3. Après avoir créé les nouveaux fichiers cibles et modifié les règles du module, reconstruisez votre projet de jeu.

Mettez à jour le code de votre serveur de jeu

Mettez à jour le code de votre serveur de jeu pour permettre la communication entre un processus de serveur de jeu et le Amazon GameLift Servers service. Votre serveur de jeu doit être en mesure

de répondre aux demandes Amazon GameLift Servers, par exemple pour démarrer ou arrêter de nouvelles sessions de jeu.

Pour ajouter du code serveur pour Amazon GameLift Servers

1. Dans votre éditeur de code, ouvrez le fichier solution (.sln) pour votre projet de jeu, qui se trouve généralement dans le dossier racine du projet. Par exemple : `GameLiftUnrealApp.sln`.
2. La solution étant ouverte, localisez le fichier d'en-tête du mode jeu du projet : `[project-name]GameMode.h` file. Par exemple : `GameLiftUnrealAppGameMode.h`.
3. Modifiez le fichier d'en-tête pour l'aligner sur le code suivant. Assurez-vous de remplacer « `GameLiftServer` » par le nom de votre propre projet. Ces mises à jour sont spécifiques au serveur de jeu ; nous vous recommandons de créer une copie de sauvegarde des fichiers du mode de jeu d'origine pour les utiliser avec votre client.

Exemple de code `GameMode.h`

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "GameLiftUnrealAppGameMode.generated.h"

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

UCLASS(minimalapi)
class AGameLiftUnrealAppGameMode : public AGameModeBase
{
    GENERATED_BODY()

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;
```

```
private:
    void InitGameLift();

private:
    TSharedPtr<FProcessParameters> ProcessParameters;
};
```

- Ouvrez le [project-name]GameMode.cpp fichier source correspondant (par exempleGameLiftUnrealAppGameMode.cpp). Modifiez le code pour l'aligner sur l'exemple de code suivant. Assurez-vous de remplacer « GameLiftUnrealApp » par le nom de votre propre projet. Ces mises à jour sont spécifiques au serveur de jeu ; nous vous recommandons de créer une copie de sauvegarde du fichier original pour l'utiliser avec votre client.

L'exemple de code suivant montre comment ajouter les éléments minimaux requis pour l'intégration du serveur avec Amazon GameLift Servers :

- Initialisez un client d'Amazon GameLift ServersAPI. L'InitSDK() appel avec les paramètres du serveur est obligatoire pour une flotte Amazon GameLift Servers Anywhere. Lorsque vous vous connectez à une flotte Anywhere, le plugin stocke les paramètres du serveur sous forme d'arguments de console. L'exemple de code permet d'accéder aux valeurs lors de l'exécution.
- Implémentez les fonctions de rappel requises pour répondre aux demandes du Amazon GameLift Servers service, notamment OnStartGameSessionOnProcessTerminate, etonHealthCheck.
- Appelez ProcessReady() un port désigné pour avertir le Amazon GameLift Servers service lorsque vous êtes prêt à héberger des sessions de jeu.

Exemple de code de serveur de jeu

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#include "GameLiftUnrealAppGameMode.h"

#include "UObject/ConstructorHelpers.h"
#include "Kismet/GameplayStatics.h"

#if WITH_GAMELIFT
#include "GameLiftServerSDK.h"
```



```

#include "GameLiftServerSDKModels.h"
#endif

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode() :
    ProcessParameters(nullptr)
{
    // Set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/
ThirdPerson/Blueprints/BP_ThirdPersonCharacter"));

    if (PlayerPawnBPClass.Class != NULL)
    {
        DefaultPawnClass = PlayerPawnBPClass.Class;
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing AGameLiftUnrealAppGameMode..."));
}

void AGameLiftUnrealAppGameMode::BeginPlay()
{
    Super::BeginPlay();

#ifdef WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
#ifdef WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Calling InitGameLift..."));

    // Getting the module first.
    FGameLiftServerSDKModule* GameLiftSdkModule =
    &ModuleManager::LoadModuleChecked<FGameLiftServerSDKModule>(FName("GameLiftServerSDK"));

    //Define the server parameters for a GameLift Anywhere fleet. These are not needed
    for a GameLift managed EC2 fleet.
    FServerParameters ServerParametersForAnywhere;

```

```
bool bIsAnywhereActive = false;
if (FParse::Param(FCommandLine::Get(), TEXT("glAnywhere")))
{
    bIsAnywhereActive = true;
}

if (bIsAnywhereActive)
{
    UE_LOG(GameServerLog, Log, TEXT("Configuring server parameters for
Anywhere..."));

    // If GameLift Anywhere is enabled, parse command line arguments and pass them
in the ServerParameters object.
    FString glAnywhereWebSocketUrl = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereWebSocketUrl="),
glAnywhereWebSocketUrl))
    {
        ServerParametersForAnywhere.m_webSocketUrl =
TCHAR_TO_UTF8(*glAnywhereWebSocketUrl);
    }

    FString glAnywhereFleetId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereFleetId="),
glAnywhereFleetId))
    {
        ServerParametersForAnywhere.m_fleetId = TCHAR_TO_UTF8(*glAnywhereFleetId);
    }

    FString glAnywhereProcessId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereProcessId="),
glAnywhereProcessId))
    {
        ServerParametersForAnywhere.m_processId =
TCHAR_TO_UTF8(*glAnywhereProcessId);
    }
    else
    {
        // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
        FString TimeString = FString::FromInt(std::time(nullptr));
        FString ProcessId = "ProcessId_" + TimeString;
        ServerParametersForAnywhere.m_processId = TCHAR_TO_UTF8(*ProcessId);
    }
}
```

```
FString glAnywhereHostId = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereHostId="),
glAnywhereHostId))
{
    ServerParametersForAnywhere.m_hostId = TCHAR_TO_UTF8(*glAnywhereHostId);
}

FString glAnywhereAuthToken = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAuthToken="),
glAnywhereAuthToken))
{
    ServerParametersForAnywhere.m_authToken =
TCHAR_TO_UTF8(*glAnywhereAuthToken);
}

FString glAnywhereAwsRegion = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAwsRegion="),
glAnywhereAwsRegion))
{
    ServerParametersForAnywhere.m_awsRegion =
TCHAR_TO_UTF8(*glAnywhereAwsRegion);
}

FString glAnywhereAccessKey = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAccessKey="),
glAnywhereAccessKey))
{
    ServerParametersForAnywhere.m_accessKey =
TCHAR_TO_UTF8(*glAnywhereAccessKey);
}

FString glAnywhereSecretKey = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSecretKey="),
glAnywhereSecretKey))
{
    ServerParametersForAnywhere.m_secretKey =
TCHAR_TO_UTF8(*glAnywhereSecretKey);
}

FString glAnywhereSessionToken = "";
if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSessionToken="),
glAnywhereSessionToken))
{
```

```

        ServerParametersForAnywhere.m_sessionToken =
TCHAR_TO_UTF8(*glAnywhereSessionToken);
    }

    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
    UE_LOG(GameServerLog, Log, TEXT(">>>> WebSocket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Aws Region: %s"),
*ServerParametersForAnywhere.m_awsRegion);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Access Key: %s"),
*ServerParametersForAnywhere.m_accessKey);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
    UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}

UE_LOG(GameServerLog, Log, TEXT("Initializing the GameLift Server..."));

//InitSDK will establish a local connection with GameLift's agent to enable further
communication.
FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule-
>InitSDK(ServerParametersForAnywhere);
if (InitSdkOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("GameLift InitSDK succeeded!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: InitSDK failed : ("));
    FGameLiftError GameLiftError = InitSdkOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"), *GameLiftError.m_errorMessage);
}

```

```

    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    return;
}

ProcessParameters = MakeShared<FProcessParameters>();

//When a game session is created, Amazon GameLift Servers sends an activation
request to the game server and passes along the game session object containing game
properties and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
invoke GameLiftServerAPI.ActivateGameSession()
ProcessParameters->OnStartGameSession.BindLambda( [=]
(Aws::GameLift::Server::Model::GameSession InGameSession)
{
    FString GameSessionId = FString(InGameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*GameSessionId);
    GameLiftSdkModule->ActivateGameSession();
});

//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback
before shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to
shutdown.
ProcessParameters->OnTerminate.BindLambda( [=]()
{
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    // First call ProcessEnding()
    FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule-
>ProcessEnding();
    // Then call Destroy() to free the SDK from memory
    FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();
    // Exit the process with success or failure
    if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess()) {
        UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
    }
    else {
        if (!processEndingOutcome.IsSuccess()) {
            const FGameLiftError& error = processEndingOutcome.GetError();
            UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error:
%s"),

```

```

        error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
    }
    if (!destroyOutcome.IsSuccess()) {
        const FGameLiftError& error = destroyOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
*error.m_errorMessage);
    }
}

});

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)
{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))
        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

```

```

}

//Here, the game server tells Amazon GameLift Servers where to find game session
log files.
//At the end of a game session, Amazon GameLift Servers uploads everything in the
specified
//location and stores it in the cloud for access later.
TArray<FString> Logfiles;
Logfiles.Add(TEXT("GameLiftUnrealApp/Saved/Logs/server.log"));
ProcessParameters->logParameters = Logfiles;

//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

if (ProcessReadyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("Process Ready!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: Process Ready Failed!"));
    FGameLiftError ProcessReadyError = ProcessReadyOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"),
*ProcessReadyError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}

UE_LOG(GameServerLog, Log, TEXT("InitGameLift completed!"));
#endif
}

```

Intégrez la carte de jeu de votre client

La carte du jeu de démarrage contient une logique de plan et des éléments d'interface utilisateur qui incluent déjà un code de base pour demander des sessions de jeu et utiliser les informations de connexion pour se connecter à une session de jeu. Vous pouvez utiliser la carte telle quelle ou les modifier selon vos besoins. Utilisez la carte du jeu de démarrage avec d'autres éléments du jeu,

tels que le projet de modèle à la troisième personne fourni par Unreal Engine. Ces ressources sont disponibles dans le navigateur de contenu. Vous pouvez les utiliser pour tester les flux de travail de déploiement du plugin ou comme guide pour créer un service de backend personnalisé pour votre jeu.

La carte de démarrage présente les caractéristiques suivantes :

- Il inclut une logique pour une flotte Anywhere et une EC2 flotte gérée. Lorsque vous gérez votre client, vous pouvez choisir de vous connecter à l'une ou l'autre flotte.
- Les fonctionnalités du client incluent la recherche d'une session de jeu (`SearchGameSessions()`), la création d'une nouvelle session de jeu (`CreateGameSession()`) et la participation directe à une session de jeu.
- Il obtient un identifiant de joueur unique à partir du pool d'utilisateurs Amazon Cognito de votre projet (cela fait partie d'une solution Anywhere déployée).

Pour utiliser la carte du jeu de démarrage

1. Dans l'éditeur UE, ouvrez la page Paramètres du projet, cartes et modes, puis développez la section Cartes par défaut.
2. Pour Editor Startup Map, sélectionnez StartupMap « » dans la liste déroulante. Vous devrez peut-être rechercher le fichier, qui se trouve dans... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Servers Plugin Content/Maps.
3. Pour la carte par défaut du jeu, sélectionnez le même StartupMap « » dans la liste déroulante.
4. Pour la carte par défaut du serveur, sélectionnez « ThirdPersonMap ». Il s'agit d'une carte par défaut incluse dans votre projet de jeu. Cette carte est conçue pour deux joueurs dans le jeu.
5. Ouvrez le panneau de détails de la carte par défaut du serveur. Définissez GameMode Override sur « Aucun ».
6. Développez la section Modes par défaut et définissez le mode de jeu par défaut global sur le mode de jeu que vous avez mis à jour pour l'intégration de votre serveur.

Après avoir apporté ces modifications à votre projet, vous êtes prêt à créer les composants de votre jeu.

Package des composants de votre jeu

Pour emballer les versions de votre serveur de jeu et de votre client de jeu

1. Ouvrez votre projet de jeu dans une version source de l'éditeur Unreal Engine.
2. Utilisez l'éditeur pour emballer les versions du client et du serveur de votre jeu.
 - a. Choisissez une cible. Accédez à Plateformes, Windows et sélectionnez l'une des options suivantes :
 - Serveur : `[your-application-name]Server`
 - Client : `[your-application-name]Client`
 - b. Démarrez le build. Accédez à Platform, Windows, Package Project.

Chaque processus d'emballage génère un exécutable : `[your-application-name]Client.exe` ou `[your-application-name]Server.exe`.

Dans le plugin, définissez les chemins d'accès aux exécutables de génération du client et du serveur sur votre poste de travail local.

Plugin pour Unreal : hébergez votre jeu localement avec Anywhere Amazon GameLift Servers

Utilisez ce flux de travail pour configurer votre station de travail locale en tant qu'hôte de serveur de jeu à l'aide d'une flotte Anywhere. Vous pouvez l'utiliser pour tester l'intégration de votre serveur de jeu avant de le déployer sur une flotte gérée dans le cloud. Cela peut également être utile pour les tests locaux lors du développement itératif de jeux.

Pour démarrer le flux de travail Amazon GameLift Servers Anywhere :

- Dans la barre d'outils principale de l'éditeur Unreal, choisissez le Amazon GameLift Servers menu, puis sélectionnez Host with Anywhere. Cette action ouvre la page du plugin Deploy Anywhere, qui présente un processus en six étapes pour intégrer, créer et lancer les composants de votre jeu.

Étape 1 : Définissez votre profil

Choisissez le profil que vous souhaitez utiliser lorsque vous suivez ce flux de travail. Le profil que vous sélectionnez a un impact sur toutes les étapes du flux de travail. Toutes les ressources que

vous créez sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

Pour définir un profil utilisateur

1. Sélectionnez un profil dans la liste déroulante des profils disponibles. Si vous n'avez pas encore de profil ou si vous souhaitez en créer un nouveau, rendez-vous dans le GameLift menu Amazon et choisissez Set AWS User Profiles.
2. Si le statut du bootstrap n'est pas « Actif », choisissez le profil Bootstrap et attendez que le statut passe à « Actif ».

Étape 2 : configurez votre code de jeu

Au cours de cette étape, préparez les versions de votre serveur de jeu et de votre client de jeu à utiliser Amazon GameLift Servers. Si vous n'avez pas encore intégré le code de votre jeu, consultez [Plugin pour Unreal : intégrez le code de votre jeu](#). Entrez les chemins d'accès aux exécutables de vos jeux sur votre poste de travail local.

- Serveur de jeu : intégrez votre serveur de jeu au SDK du serveur Amazon GameLift Servers et empaquetez la version de votre serveur de jeu. Pour obtenir des instructions, veuillez consulter [Plugin pour Unreal : intégrez le code de votre jeu](#). Le serveur de jeu doit être intégré au SDK du serveur afin d'établir une communication avec le Amazon GameLift Servers service et de répondre aux demandes de démarrage de nouvelles sessions de jeu et d'accepter les connexions des clients du jeu.
- Client de jeu : au minimum, vous avez besoin d'un client de jeu capable de se connecter à votre serveur de jeu à l'aide de l'adresse IP et des informations de port. Si les composants de votre client de jeu ne sont pas encore configurés Amazon GameLift Servers, vous pouvez utiliser l' AWS CLI outil pour demander manuellement de nouvelles sessions de jeu, obtenir des informations de connexion et utiliser ces informations pour connecter le client de jeu.

À un moment donné, vous aurez besoin d'un service principal pour envoyer de nouvelles demandes de sessions de jeu au Amazon GameLift Servers service et transmettre les informations de connexion à un client de jeu. Vous pouvez utiliser les cartes de test incluses dans le plugin pour ajouter des Amazon GameLift Servers fonctionnalités client à votre projet de jeu. Pour obtenir de l'aide sur la création d'une solution personnalisée, consultez [Ajoutez Amazon GameLift Servers à votre client de jeu](#).

Étape 3 : Connectez-vous à une flotte n'importe où

Au cours de cette étape, vous désignez une flotte Anywhere à utiliser. Une flotte Anywhere définit un ensemble de ressources informatiques, qui peuvent être situées n'importe où, pour l'hébergement de serveurs de jeux.

- Si le AWS compte que vous utilisez actuellement possède des flottes Anywhere existantes, ouvrez le champ déroulant Nom de la flotte et choisissez une flotte. Cette liste déroulante affiche uniquement les flottes Anywhere de la AWS région correspondant au profil utilisateur actuellement actif.
- S'il n'existe aucune flotte existante, ou si vous souhaitez en créer une nouvelle, choisissez Create new Anywhere fleet et saisissez un nom de flotte.

Après avoir choisi une flotte Anywhere pour votre projet, vérifiez que l'état de Amazon GameLift Servers la flotte est actif et affiche l'identifiant de la flotte. Vous pouvez suivre la progression de cette demande dans le journal de sortie de l'éditeur Unreal.

Étape 4 : Enregistrez votre poste de travail

Au cours de cette étape, vous enregistrez votre poste de travail local en tant que ressource de calcul dans le nouveau parc Anywhere.

Pour enregistrer votre station de travail en tant que système de calcul n'importe où

1. Entrez un nom de calcul pour votre machine locale. Si vous ajoutez plusieurs ordinateurs au parc, les noms doivent être uniques.
2. Fournissez une adresse IP pour votre machine locale. Ce champ correspond par défaut à l'adresse IP publique de votre machine. Vous pouvez également utiliser localhost (127.0.0.1) tant que vous exécutez votre client de jeu et votre serveur sur la même machine.
3. Choisissez Register compute. Vous pouvez suivre la progression de cette demande dans le journal de sortie de l'éditeur Unreal.

En réponse à cette action, Amazon GameLift Servers vérifie qu'il peut se connecter au calcul et renvoie des informations sur le calcul nouvellement enregistré. Il crée également les arguments de console dont les exécutables de jeu ont besoin lors de l'initialisation de la communication avec le service. Amazon GameLift Servers

Étape 5 : générer un jeton d'authentification

Les processus du serveur de jeu qui s'exécutent sur votre ordinateur Anywhere ont besoin d'un jeton d'authentification pour appeler le Amazon GameLift Servers service. Le plugin génère et stocke automatiquement un jeton d'authentification pour la flotte Anywhere chaque fois que vous lancez le serveur de jeu depuis le plugin. La valeur du jeton d'authentification est stockée sous forme d'argument de ligne de commande, que le code de votre serveur peut récupérer lors de l'exécution.

Les exemples de code donnés ci-dessus vous permettent également d'utiliser [AWS Signature Version 4 \(SigV4\) pour les demandes d'API](#). SigV4 est le protocole de AWS signature permettant d'ajouter des informations d'authentification aux demandes d'API.

Vous n'avez aucune action à effectuer au cours de cette étape.

Étape 6 : Lancer le jeu

À ce stade, vous avez terminé toutes les tâches nécessaires pour lancer et jouer à votre jeu multijoueur sur un poste de travail local en utilisant Amazon GameLift Servers.

Pour jouer à votre jeu hébergé

1. Lancez votre serveur de jeu. Le serveur de jeu vous avertira Amazon GameLift Servers lorsqu'il sera prêt à héberger des sessions de jeu.
2. Lancez votre client de jeu et utilisez les nouvelles fonctionnalités pour démarrer une nouvelle session de jeu. Cette demande est envoyée Amazon GameLift Servers via le nouveau service principal. En réponse Amazon GameLift Servers, appelle le serveur de jeu, qui fonctionne sur votre machine locale, pour démarrer une nouvelle session de jeu. Lorsque la session de jeu est prête à accepter des joueurs, Amazon GameLift Servers fournit des informations de connexion permettant au client du jeu de rejoindre la session de jeu.

Plugin pour Unreal : déployez votre jeu sur une flotte gérée EC2

Dans ce flux de travail, déployez votre jeu pour l'héberger sur des ressources informatiques basées sur le cloud gérées par Amazon GameLift Servers. Téléchargez la version de votre serveur de jeu intégré sur le Amazon GameLift Servers service pour le déploiement. Si vous n'avez pas encore intégré le code de votre jeu, consultez [Plugin pour Unreal : intégrez le code de votre jeu](#). Lorsque ce flux de travail sera terminé, vous disposerez d'un client de jeu fonctionnel qui pourra se connecter à vos serveurs de jeu dans le cloud.

Pour démarrer le EC2 flux de travail Amazon Amazon GameLift Servers géré :

- Dans la barre d'outils principale de l'éditeur Unreal, choisissez le Amazon GameLift Servers menu, puis sélectionnez Host with Managed EC2. Cette action ouvre la page du plugin Deploy Amazon EC2 Fleet, qui présente un processus en six étapes pour intégrer, créer, déployer et lancer les composants de votre jeu.

Étape 1 : Définissez votre profil

Choisissez le profil que vous souhaitez utiliser lorsque vous suivez ce flux de travail. Le profil que vous sélectionnez a un impact sur toutes les étapes du flux de travail. Toutes les ressources que vous créez sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

Pour définir un profil utilisateur

1. Sélectionnez un profil dans la liste déroulante des profils disponibles. Si vous n'avez pas encore de profil ou si vous souhaitez en créer un nouveau, rendez-vous dans le GameLift menu Amazon et choisissez Set AWS User Profiles.
2. Si le statut du bootstrap n'est pas « Actif », choisissez le profil Bootstrap et attendez que le statut passe à « Actif ».

Étape 2 : configurez votre code de jeu

Au cours de cette étape, préparez les versions de votre serveur de jeu et de votre client de jeu pour qu'elles fonctionnent avec le SDK du serveur Amazon GameLift Servers C++ pour Unreal. Si vous n'avez pas encore intégré le code de votre jeu et créé les exécutables du client et du serveur de jeu, consultez. [Plugin pour Unreal : intégrez le code de votre jeu](#) Entrez les chemins d'accès aux exécutables de vos jeux sur votre poste de travail local.

À cette étape du flux de travail, le plugin fournit des liens vers des instructions et du code source pour configurer une version source de l'Unreal Editor. Vous devez utiliser la version source lors de la création de vos composants client et serveur.

Après avoir créé un serveur de jeu intégré au SDK du serveur, effectuez les tâches suivantes pour le préparer au téléchargement vers Amazon GameLift Servers un hébergement.

Pour préparer la version de votre serveur pour le déploiement dans le cloud (Windows)

Dans le `WindowsServer` dossier où l'éditeur Unreal stocke les fichiers de build de votre serveur par défaut, effectuez les ajouts suivants :

1. Copiez le script d'installation du build du serveur à la racine du **WindowsServer** dossier. Le script d'installation est inclus dans le téléchargement du plugin. Cherchez le fichier `[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat`. Amazon GameLift Servers utilise ce fichier pour installer la version du serveur sur vos ordinateurs d'hébergement.
2. Copiez le **VC_redist.x64.exe** fichier à la racine du **WindowsServer** dossier. Vous pouvez ignorer cette étape si vous utilisez Unreal Engine version 5.6 ou ultérieure. Ce fichier est inclus dans votre installation de Visual Studio. Il est généralement situé à `C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`.
3. Ajoutez les fichiers de bibliothèque OpenSSL à la version de votre serveur de jeu. . Vous pouvez ignorer cette étape si votre serveur de jeu est intégré au SDK 5.3 ou à une version ultérieure. Cette version est incluse dans le Amazon GameLift Servers plugin pour Unreal, version 3.0 ou ultérieure.

Localisez et copiez manuellement les bibliothèques OpenSSL dans le répertoire de votre package de construction de jeu à l'adresse. `<YourGame>/Binaries/Win64` Vous devez utiliser la même version d'OpenSSL que celle utilisée par votre version d'Unreal Engine 5. Les versions de jeu déployées avec les mauvaises bibliothèques OpenSSL ne seront pas en mesure de communiquer avec le service. Amazon GameLift Servers

Recherchez les bibliothèques OpenSSL dans la source de votre moteur de jeu. L'emplacement varie en fonction de votre environnement de développement :

Sous Windows :

- `[ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libssl-1_1-x64.dll`
- `[ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libcrypto-1_1-x64.dll`

Sous Linux :

- `Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1`

- Engine/Source/Thirdparty/openssl/1.1.1n/include/libcrypto.so.1.1

Pour préparer la version de votre serveur pour le déploiement dans le cloud (Linux)

Pour des instructions plus détaillées sur la préparation d'un serveur de jeu conçu pour Linux, consultez [Création du SDK du serveur Amazon GameLift Servers pour Unreal Engine 5 sur Amazon Linux](#).

1. Désignez un répertoire de travail pour organiser vos fichiers de compilation. La structure du répertoire de travail est déployée telle quelle sur chaque ordinateur d'hébergement. Ajoutez votre serveur de jeu créé sous Linux et tous les fichiers dépendants.
2. Créez un script d'installation de compilation du serveur à la racine de votre répertoire de travail. Si nécessaire, créez un `install.sh` fichier et ajoutez les commandes nécessaires pour installer correctement la version de votre serveur de jeu. Amazon GameLift Servers utilise ce fichier pour installer le build du serveur sur chaque ressource EC2 d'hébergement.
3. Ajoutez les fichiers de bibliothèque OpenSSL à la version de votre serveur de jeu. Vous pouvez ignorer cette étape si votre serveur de jeu est intégré au SDK 5.3 ou à une version ultérieure.

Localisez et copiez manuellement les bibliothèques. Vous devez utiliser la même version d'OpenSSL que celle utilisée par votre version d'Unreal Engine 5. Les versions de jeu déployées avec les mauvaises bibliothèques OpenSSL ne seront pas en mesure de communiquer avec le service. Amazon GameLift Servers

- a. Recherchez les bibliothèques OpenSSL dans la source de votre moteur de jeu. L'emplacement varie en fonction de votre environnement de développement :

Sous Windows :

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libssl-1_1-x64.dll
- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libcrypto-1_1-x64.dll

Sous Linux :

- Engine/Source/Thirdparty/openssl/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/openssl/1.1.1n/include/libcrypto.so.1.1

- b. Lorsque vous localisez les bibliothèques OpenSSL, copiez-les dans le répertoire du package de construction de votre jeu à l'adresse. <YourGame>/Binaries/Linux

Étape 3 : Sélection du scénario de déploiement

Au cours de cette étape, vous choisissez la solution d'hébergement de jeux que vous souhaitez déployer à ce stade. Vous pouvez effectuer plusieurs déploiements de votre jeu, en utilisant n'importe quel scénario.

- Flotte à région unique : déploye votre serveur de jeu sur une flotte unique de ressources d'hébergement dans la région par défaut AWS du profil actif. Ce scénario constitue un bon point de départ pour tester l'intégration de votre serveur AWS et la configuration de la version du serveur. Il déploie les ressources suivantes :
 - AWS flotte (à la demande) avec la version de votre serveur de jeu installée et en cours d'exécution.
 - Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
 - Autorisateur de passerelle API qui relie le groupe d'utilisateurs à APIs.
 - Web API pour limiter les appels excessifs des joueurs à la passerelle API.
 - Passerelle API + fonction Lambda permettant aux joueurs de demander une machine à sous. Cette fonction appelle `CreateGameSession()` si aucune fonction n'est disponible.
 - Passerelle API + fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.
- FlexMatch flotte : déploye votre serveur de jeu sur un ensemble de flottes et met en place un FlexMatch système de matchmaking avec des règles pour créer des parties entre joueurs. Ce scénario utilise un hébergement Spot à faible coût avec une structure multi-flottes et multi-sites pour une disponibilité durable. Cette approche est utile lorsque vous êtes prêt à commencer à concevoir un composant de matchmaking pour votre solution d'hébergement. Dans ce scénario, vous allez créer les ressources de base pour cette solution, que vous pourrez personnaliser ultérieurement selon vos besoins. Il déploie les ressources suivantes :
 - FlexMatch configuration du matchmaking et règles de matchmaking définies pour accepter les demandes des joueurs et former des matchs.
 - Trois AWS flottes équipées de votre version de serveur de jeu sont installées et fonctionnent à plusieurs endroits. Comprend deux flottes Spot et une flotte à la demande en tant que sauvegarde.

- AWS file d'attente de placement de sessions de jeu qui répond aux demandes de matchs proposés en trouvant la meilleure ressource d'hébergement possible (en fonction de la viabilité, du coût, de la latence des joueurs, etc.) et en démarrant une session de jeu.
- Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
- Autorisateur de passerelle API qui relie le groupe d'utilisateurs à APIs.
- Web ACI pour limiter les appels excessifs des joueurs à la passerelle API.
- Passerelle API + fonction Lambda permettant aux joueurs de demander une machine à sous. Cette fonction appelle `StartMatchmaking()`.
- Passerelle API + fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.
- Tables Amazon DynamoDB pour stocker les tickets de matchmaking pour les joueurs et les informations relatives aux sessions de jeu.
- Rubrique SNS + Fonction Lambda pour `GameSessionQueue` gérer les événements.

Étape 4 : définir les paramètres du jeu

Au cours de cette étape, vous décrivez le jeu sur lequel vous souhaitez le télécharger ; AWS

- Nom de version du serveur : donnez un nom significatif à la version de votre serveur de jeu. AWS utilise ce nom pour faire référence à la copie de la version de votre serveur qui est téléchargée et utilisée pour les déploiements.
- Système d'exploitation pour la version du serveur : entrez le système d'exploitation sur lequel votre serveur est conçu pour fonctionner. Cela indique le AWS type de ressources informatiques à utiliser pour héberger votre jeu.
- Dossier du serveur de jeu : identifiez le chemin d'accès au dossier de compilation de votre serveur local.
- Version du serveur de jeu : identifiez le chemin d'accès au fichier exécutable du serveur de jeu.
- Chemin du client de jeu : identifiez le chemin d'accès à l'exécutable du client de jeu.
- Résultat de configuration du client : ce champ doit pointer vers un dossier de votre build client contenant votre AWS configuration. Recherchez-le à l'endroit suivant : `[client-build]/[project-name]/Content/CloudFormation`.

Étape 5 : scénario de déploiement

Au cours de cette étape, vous déployez votre jeu sur une solution d'hébergement cloud en fonction du scénario de déploiement que vous avez choisi. Ce processus peut prendre plusieurs minutes pour AWS valider la version de votre serveur, approvisionner les ressources d'hébergement, installer votre serveur de jeu, lancer les processus du serveur et le préparer à héberger des sessions de jeu.

Pour démarrer le déploiement, choisissez Deploy CloudFormation. Vous pouvez suivre l'état de votre hébergement de jeux ici. Pour obtenir des informations plus détaillées, vous pouvez vous connecter à la console AWS de gestion AWS et consulter les notifications d'événements. Assurez-vous de vous connecter en utilisant le même compte, le même utilisateur et AWS la même région que le profil utilisateur actif dans le plugin.

Lorsque le déploiement est terminé, votre serveur de jeu est installé sur une AWS EC2 instance. Au moins un processus serveur est en cours d'exécution et prêt à démarrer une session de jeu.

Étape 6 : Lancer le client

À ce stade, vous avez terminé toutes les tâches nécessaires pour lancer et jouer à votre jeu multijoueur hébergé par Amazon GameLift Servers. Pour jouer au jeu, lancez une instance de votre client de jeu.

Si vous avez déployé le scénario de flotte unique, vous pouvez ouvrir une instance client unique avec un seul joueur, accéder à la carte du serveur et vous déplacer. Ouvrez des instances supplémentaires du client de jeu pour ajouter un deuxième joueur à la même carte de jeu du serveur.

Si vous avez déployé le FlexMatch scénario, la solution attend qu'au moins deux clients soient mis en file d'attente pour le placement des sessions de jeu avant que les joueurs puissent accéder à la carte du serveur.

Plugin pour Unreal : déployez votre jeu sur une flotte de conteneurs gérée

Utilisez ce flux de travail guidé pour créer une image de conteneur pour votre serveur de jeu et la déployer sur une solution d'hébergement basée sur des conteneurs. Si vous n'avez pas encore intégré le code de votre jeu, consultez [Plugin pour Unreal : intégrez le code de votre jeu](#). Lorsque vous avez terminé ce flux de travail avec succès, votre serveur de jeu conteneurisé fonctionne dans le cloud et vous pouvez utiliser le plugin pour démarrer un client de jeu, vous connecter à une session de jeu et jouer au jeu.

Avant de commencer

Ce flux de travail suppose que vous avez effectué les tâches suivantes.

- Intégrez le code de votre serveur de jeu au SDK Amazon GameLift Servers du serveur. Votre serveur de jeu hébergé doit être en mesure de communiquer avec le Amazon GameLift Servers service afin qu'il puisse répondre aux demandes de démarrage de nouvelles sessions de jeu et signaler l'état des sessions de jeu. Si vous n'avez pas terminé cette tâche, nous vous recommandons de suivre d'abord le flux de travail du plugin Host with Anywhere. Pour obtenir des conseils sur la préparation du code de votre serveur de jeu, consultez [Mettez à jour le code de votre serveur de jeu](#). Pour une flotte de conteneurs gérée, vous devez intégrer votre jeu à la version 5.2 ou supérieure du SDK du serveur.

Note

Si vous utilisez la carte de démarrage du jeu, cette tâche est déjà terminée pour vous.

- Package le fichier exécutable de votre serveur de jeu pour qu'il fonctionne sous Linux. Si vous développez sous Windows et que vous intégrez le SDK du serveur C++ version 5.2.x ou antérieure, vous devrez utiliser le kit d'outils de compilation croisée [Unreal](#). Vous pouvez également configurer un espace de travail Linux distinct ou utiliser un outil tel que le sous-système Windows pour Linux (WSL).
- Rassemblez les fichiers à déployer avec la version de votre serveur de jeu. Sur votre machine locale, créez un répertoire de travail pour organiser les fichiers, qui seront intégrés à l'image conteneur de votre serveur de jeu. Il peut s'agir de dépendances au jeu, d'un script pour lancer des serveurs de jeu et d'autres processus lors du démarrage d'un conteneur, etc.
- Ajoutez les fichiers de bibliothèque OpenSSL pour la compilation de votre serveur de jeu. Vous pouvez ignorer cette étape si votre serveur de jeu est intégré au SDK 5.3 ou à une version ultérieure.

Localisez et copiez manuellement les bibliothèques OpenSSL dans le répertoire de votre package de construction de jeu à l'adresse. <YourGame>/Binaries/Win64 Vous devez utiliser la même version d'OpenSSL que celle utilisée par votre version d'Unreal Engine 5. Les versions de jeu déployées avec les mauvaises bibliothèques OpenSSL ne seront pas en mesure de communiquer avec le service. Amazon GameLift Servers

Pour trouver les bibliothèques OpenSSL, consultez la source de votre moteur de jeu.

L'emplacement varie en fonction de votre environnement de développement :

Sous Windows :

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libssl-1_1-x64.dll
- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libmobiledevice\x64\libcrypto-1_1-x64.dll

Sous Linux :

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1
- Intégrez le code client de votre jeu à Amazon GameLift Servers. Pour effectuer cette tâche, vous pouvez ajouter un exemple de ressource (inclus dans le plugin) déjà intégré. Pour obtenir des conseils sur la préparation du code client de votre jeu, consultez [Intégrez la carte de jeu de votre client](#).
- Installez Docker sur votre machine locale. Vous devez installer cet outil si vous souhaitez que le plugin crée des images de conteneur pour vous et les envoie vers un référentiel ECR. Vous pouvez également effectuer ces tâches manuellement et demander au plugin d'utiliser une image de conteneur existante. Pour plus d'informations sur la création manuelle de votre image, voir [Création d'une image de conteneur pour Amazon GameLift Servers](#).

Pour démarrer le flux de travail des conteneurs Amazon GameLift Servers gérés :

- Dans la barre d'outils principale de l'éditeur Unreal, choisissez le Amazon GameLift Servers menu, puis sélectionnez Managed Containers. Cette action ouvre la page du plugin Host with Managed Containers, qui présente un step-by-step processus permettant de créer une image de conteneur avec la version de votre serveur de jeu, de la déployer dans une flotte de conteneurs et de lancer votre jeu.

Étape 0 : Définissez votre profil

Cette section affiche le profil utilisateur actuellement sélectionné. Vérifiez que le profil utilisateur actuel est celui que vous souhaitez utiliser pour ce flux de travail. Toutes les ressources que vous créez dans ce flux de travail sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

Il se peut que vous deviez modifier le profil utilisateur sélectionné si :

- Aucun profil n'est actuellement sélectionné.
- Vous souhaitez sélectionner un autre profil ou créer un nouveau profil.
- Vous devez amorcer le profil sélectionné (si le statut du bootstrap est inactif).

Pour définir ou modifier le profil utilisateur sélectionné

- Dans le Amazon GameLift Servers menu, choisissez Open AWS Access Credentials.

Étape 1 : Évaluer l'état de préparation du conteneur

Avant de déployer votre serveur de jeu sur un parc de conteneurs, vous devez le regrouper dans une image de conteneur et le stocker dans un référentiel Amazon ECR. Le plugin peut effectuer ces tâches pour vous ou vous pouvez les effectuer manuellement. Au cours de cette étape, fournissez des informations sur le statut de votre image de conteneur et du référentiel ECR.

Utilisez les questions d'évaluation pour indiquer au plugin les étapes à suivre :

- Créez une nouvelle image de conteneur. Si vous choisissez cette option, l'étape suivante vous demandera l'emplacement du répertoire de compilation de votre serveur de jeu et de l'exécutable de compilation. Le plugin utilise un modèle Dockerfile (fourni par Amazon GameLift Servers) et le configure automatiquement pour votre jeu. Vous pouvez consulter le modèle à l'adresse [Créez une image de conteneur pour Amazon GameLift Servers](#). Après avoir choisi cette option, indiquez où vous souhaitez que le plugin stocke la nouvelle image :
- Créez un nouveau référentiel Amazon ECR et envoyez-y l'image du conteneur. Le plugin crée un dépôt ECR privé en utilisant le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné.
- Transférez l'image du conteneur vers un référentiel Amazon ECR créé précédemment. Si vous choisissez cette option, l'étape suivante vous invitera à sélectionner un référentiel Amazon ECR existant dans une liste. La liste inclut tous les référentiels Amazon ECR pour le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné. Vous pouvez sélectionner un dépôt public ou privé.
- Utilisez une image de conteneur existante. Si vous avez créé une image manuellement, nous vous recommandons d'utiliser le modèle Dockerfile fourni par Amazon GameLift Servers, disponible sur [Créez une image de conteneur pour Amazon GameLift Servers](#). Après avoir choisi cette option, indiquez où se trouve l'image.

- Une image générée par Docker stockée localement. Si vous choisissez cette option, le plugin crée un nouveau référentiel privé Amazon ECR et y envoie le fichier image local. L'étape suivante vous demandera un identifiant d'image, que le plugin utilisera pour localiser le fichier image.
- Une image de conteneur déjà stockée dans un référentiel Amazon ECR. Si vous choisissez cette option, l'étape suivante vous invitera à sélectionner un référentiel Amazon ECR et une image existants dans une liste. La liste inclut tous les référentiels Amazon ECR pour le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné. Vous pouvez sélectionner un dépôt public ou privé.

Étape 2 : configurer le déploiement des images

Au cours de cette étape, fournissez les informations dont le plugin a besoin pour déployer votre image de conteneur dans une flotte de conteneurs. Cette étape demande les informations suivantes :

- L'emplacement de la version de votre serveur de jeu, de l'image de conteneur ou du référentiel Amazon ECR, en fonction de vos sélections à l'étape 1.
- Le scénario à utiliser pour le déploiement de vos conteneurs gérés.
- Le chemin de sortie de configuration du client. Sélectionnez le dossier dans la version de votre client qui contient votre AWS configuration. Recherchez-le à l'endroit suivant : `[client-build]/[project-name]/Content/CloudFormation`.
- Paramètres de déploiement facultatifs. Cette section contient les paramètres de configuration que le plugin utilise par défaut. Vous pouvez les modifier ou conserver les valeurs par défaut
 - Le nom du jeu est défini sur le nom de votre projet de jeu par défaut. Toutes les AWS ressources créées par le plugin font référence à la valeur du nom du jeu.
 - La plage de ports, la limite de mémoire et la limite de vCPU sont des paramètres de configuration pour le parc de conteneurs. Pour plus d'informations sur la personnalisation de ces valeurs, reportez-vous [Configuration des connexions réseau](#) aux sections pour la plage de ports de connexion et [Définissez des limites de ressources](#) pour les limites de ressources.
 - La balise d'image de conteneur est utilisée pour classer vos images de conteneur dans Amazon ECR. La valeur par défaut est `unreal-gamelift-plugin`.
 - Nom du référentiel Amazon ECR. Vous pouvez modifier ce champ pour suggérer un nom personnalisé uniquement lorsque le plugin crée un référentiel ECR pour vous. La valeur par défaut est `unreal-game lift-plugin-ecr-repository`.

Options de scénario de déploiement

Flotte de conteneurs unirégionale

Ce scénario déploie votre serveur de jeu sur une seule flotte de conteneurs. C'est un bon point de départ pour tester l'intégration de votre serveur AWS et la configuration de votre conteneur. Il déploie les ressources suivantes.

- Amazon GameLift ServersLa définition d'un groupe de conteneurs décrit comment déployer et exécuter vos images de conteneurs sur une flotte de conteneurs.
- Amazon GameLift Serversflotte de conteneurs (à la demande) avec le conteneur de votre serveur de jeu installé et en cours d'exécution, avec alias.
- Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
- Autorisateur API Gateway qui relie le groupe d'utilisateurs à APIs.
- Liste de contrôle d'accès Web (ACL) pour limiter les appels excessifs des joueurs à API Gateway.
- Service principal pour envoyer des demandes au Amazon GameLift Servers service au nom des clients du jeu, par exemple pour demander des sessions de jeu et rejoindre des jeux :
 - API Gateway + Fonction Lambda permettant aux joueurs de demander un créneau de session de jeu. Cette fonction appelle `CreateGameSession()` si aucun emplacement ouvert n'est disponible.
 - API Gateway + Fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.

Flotte de conteneurs à région unique avec FlexMatch

Ce scénario déploie votre serveur de jeu sur une flotte de conteneurs, configure le placement des sessions de jeu et met en place FlexMatch le matchmaking. Ce scénario est utile lorsque vous êtes prêt à commencer à concevoir un système de matchmaking personnalisé pour votre solution d'hébergement. Utilisez ce scénario pour créer les ressources de base de cette solution, que vous pourrez personnaliser ultérieurement selon vos besoins. Il déploie les ressources suivantes :

- Amazon GameLift Serversdéfinition de groupe de conteneurs qui décrit comment déployer et exécuter vos images de conteneurs sur une flotte de conteneurs.
- Amazon GameLift Serversflotte de conteneurs (à la demande) avec le conteneur de votre serveur de jeu installé et en cours d'exécution, avec alias.

- FlexMatch configuration du matchmaking et règles de matchmaking définies pour accepter les demandes des joueurs et former des matchs.
- Amazon GameLift Servers file d'attente de session de jeu qui répond aux demandes de matchs proposés en trouvant la meilleure ressource d'hébergement possible (en fonction de la viabilité, du coût, de la latence des joueurs, etc.) et en démarrant une session de jeu.
- Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
- Autorisateur API Gateway qui relie le groupe d'utilisateurs à APIs.
- Liste de contrôle d'accès Web (ACL) pour limiter les appels excessifs des joueurs à API Gateway.
- Service principal pour envoyer des demandes au Amazon GameLift Servers service au nom des clients du jeu, par exemple pour demander des sessions de jeu et rejoindre des jeux :
 - API Gateway + Fonction Lambda permettant aux joueurs de demander un créneau de session de jeu. Cette fonction appelle `StartMatchmaking()` si aucun emplacement ouvert n'est disponible.
 - API Gateway + Fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.
- Tables DynamoDB pour stocker les tickets de matchmaking pour les joueurs et les informations sur les sessions de jeu.
- Rubrique Amazon SNS + Fonction Lambda pour gérer les événements. `GameSessionQueue`

Déployer une flotte de conteneurs

Lorsque la configuration de votre flotte est terminée, cliquez sur le bouton **Déployer une flotte de conteneurs** pour démarrer le déploiement. Ce processus peut prendre plusieurs minutes pendant que le plugin crée une image de conteneur et la transmet à ECR, provisionne les ressources d'hébergement pour la flotte de conteneurs, déploie la flotte et d'autres AWS ressources pour le scénario de solution d'hébergement sélectionné.

Lorsque vous démarrez un déploiement, vous pouvez suivre la progression de chaque étape. Selon votre configuration, les étapes peuvent inclure les suivantes :

- Configuration de l'image du conteneur
- Création d'un référentiel Amazon ECR
- Création d'une image et diffusion vers Amazon ECR
- Création d'une définition de groupe de conteneurs

- Création d'une flotte de conteneurs

Pour obtenir des informations de déploiement plus détaillées, choisissez Afficher dans AWS la console de gestion. Lorsque la flotte de conteneurs atteint le statut actif, elle gère activement des conteneurs avec des processus de serveur prêts à héberger des sessions de jeu.

Une fois le déploiement terminé, vous disposez d'une flotte de conteneurs fonctionnelle prête à accueillir des sessions de jeu et à accepter les connexions entre joueurs.

Vous ne pouvez pas arrêter un déploiement en cours. Si le déploiement entre dans un mauvais état ou échoue, vous pouvez recommencer à zéro en utilisant l'option Réinitialiser le déploiement.

Lancer le client

À ce stade, vous avez terminé toutes les tâches pour lancer et jouer à votre jeu multijoueur hébergé par Amazon GameLift Servers. Pour jouer à votre jeu, choisissez Start Client pour lancer une instance locale de votre client de jeu.

- Si vous avez déployé le scénario à flotte unique, ouvrez une instance de votre client de jeu avec un joueur et accédez à la carte du serveur pour vous déplacer. Vous pouvez ouvrir une deuxième instance du client de jeu pour ajouter un deuxième joueur à la même carte de jeu du serveur.
- Si vous avez déployé le FlexMatch scénario, la solution d'hébergement attend qu'au moins deux clients du jeu fassent des demandes de matchmaking. Ouvrez au moins deux instances de votre client de jeu avec un joueur. Les deux joueurs seront jumelés et invités à rejoindre une session de jeu pour le match.

Mettre à jour une flotte de conteneurs

Si vous avez déployé avec succès une solution d'hébergement de conteneurs gérés, vous pouvez utiliser la fonctionnalité de déploiement des mises à jour. Cette option vous permet de mettre à jour les paramètres de configuration d'une flotte de conteneurs déployée, sans avoir à créer une nouvelle flotte.

Lorsque vous mettez à jour un déploiement, vous pouvez déployer une image de conteneur avec une version de serveur de jeu différente, modifier le référentiel Amazon ECR, choisir un autre scénario de déploiement et personnaliser les paramètres de configuration facultatifs.

Lorsque vous êtes prêt à déployer vos modifications, choisissez **Mettre à jour**. Le temps requis pour une mise à jour de déploiement est similaire à celui d'un déploiement complet. Pour obtenir des informations détaillées sur le déploiement, choisissez **Afficher dans AWS** la console de gestion.

Nettoyez les ressources déployées

Il est recommandé de nettoyer les AWS ressources de votre solution de conteneurs gérés dès que vous n'en avez plus besoin. Il est possible que ces ressources continuent de vous coûter cher si vous ne les supprimez pas.

Supprimez les ressources suivantes :

- Pile de ressources de conteneurs gérée. Les ressources de cette pile dépendent du scénario de déploiement que vous avez sélectionné. Pour supprimer l'intégralité de la pile, utilisez la AWS CloudFormation console. Les piles générées à partir du Amazon GameLift Servers plugin utilisent la convention de dénomination suivante : `GameLiftPluginForUnreal-{GameName}-Containers`. Attendez que le processus de suppression de la pile soit terminé avant de lancer un nouveau déploiement de conteneurs gérés dans le plugin. Pour plus d'informations, consultez [Supprimer une pile de la CloudFormation console](#).
- Référentiel Amazon ECR. Si vous avez utilisé le plugin pour créer un référentiel pour votre image de conteneur, vous souhaitez peut-être supprimer tous les référentiels dont vous n'avez plus besoin. Il n'est pas nécessaire de supprimer un référentiel avant de réinitialiser un déploiement de conteneurs gérés. Si vous mettez à jour ou réinitialisez un déploiement, le plugin utilisera automatiquement le même référentiel à moins qu'il ne soit demandé d'en utiliser un autre. Pour plus d'informations, consultez [Supprimer un référentiel privé dans Amazon ECR](#).

Amazon GameLift Serversplugin pour Unity (SDK serveur 5.x)

Ce plugin ajoute le SDK et les outils du serveur Amazon GameLift Servers C# à l'éditeur Unity. Utilisez les flux de travail guidés de l'interface utilisateur pour intégrer les fonctionnalités du SDK du serveur dans votre projet de jeu et déployer une solution Amazon GameLift Servers d'hébergement pour votre serveur de jeu.

Avec le plugin, vous pouvez créer une solution d'hébergement fonctionnelle de base, puis l'optimiser et la personnaliser selon vos besoins. Configurez une flotte Amazon GameLift Servers Anywhere avec votre poste de travail local comme hôte. Pour un hébergement dans le cloud avec des flottes de conteneurs gérés EC2 ou gérés, déployez votre serveur de jeu avec une solution complète pour gérer les demandes de sessions de jeu et les connexions clients.

Rubriques

- [Installez le plugin pour votre projet de jeu Unity](#)
- [Plugin pour Unity : configurer un profil AWS utilisateur](#)
- [Plugin pour Unity : configurez des tests locaux avec Amazon GameLift Servers Anywhere](#)
- [Plugin pour Unity : déployez votre jeu sur une EC2 flotte gérée](#)
- [Plugin pour Unity : déployez votre jeu sur une flotte de conteneurs gérée](#)

Installez le plugin pour votre projet de jeu Unity

[Téléchargez le Amazon GameLift Servers plugin pour Unity sur GitHub](#)

Consultez le fichier readme du GitHub référentiel pour savoir comment installer le plugin pour un projet de jeu.

Le plugin inclut les composants suivants :

- Modules de plug-in pour l'éditeur Unity. Lorsque le plugin est installé, un nouvel élément du menu principal vous donne accès aux Amazon GameLift Servers fonctionnalités.
- bibliothèques C# pour l'API de Amazon GameLift Servers service avec fonctionnalité côté client.
- bibliothèques C# pour le SDK du Amazon GameLift Servers serveur (version 5.x).
- Des exemples de contenu de jeu, y compris des éléments et des scènes, que vous pourrez essayer Amazon GameLift Servers même si vous n'avez pas de jeu multijoueur prêt à être compilé.
- Configurations de solution, fournies sous forme de AWS CloudFormation modèles, que le plugin utilise lors du déploiement de votre serveur de jeu sur le cloud à des fins d'hébergement.

Ce plugin utilise des AWS CloudFormation modèles pour déployer des solutions d'hébergement pour des scénarios de jeu courants. Utilisez ces solutions telles que fournies ou personnalisez-les selon les besoins de vos jeux.

Plugin pour Unity : configurer un profil AWS utilisateur

Après avoir installé le plugin, configurez un profil utilisateur avec un AWS compte valide. Vous pouvez gérer plusieurs profils dans le plugin, mais vous ne pouvez sélectionner qu'un seul profil à la fois. Chaque fois que vous travaillez dans le plugin, sélectionnez un profil à utiliser. Chaque page de flux de travail affiche le profil actuellement sélectionné.

La gestion de plusieurs profils vous permet de passer d'un déploiement d'hébergement à un autre. Par exemple, vous pouvez configurer des profils qui utilisent le même AWS compte mais qui sont déployés AWS dans différentes régions. Vous pouvez également configurer des profils avec différents AWS comptes ou utilisateurs et ensembles d'autorisations.

 Note

Si vous avez installé la AWS CLI sur votre poste de travail et qu'un profil est déjà configuré, le Amazon GameLift Servers plugin le détectera et le répertoriera comme profil existant. Le plugin sélectionne automatiquement n'importe quel profil nommé `[default]`. Vous pouvez utiliser un profil existant ou en créer un nouveau.

Tous les profils doivent être initialisés pour configurer certaines AWS ressources requises sous l'utilisateur de votre compte.

Pour gérer vos AWS profils

1. Dans la barre d'outils principale de Unity, choisissez le Amazon GameLift Servers menu, puis sélectionnez AWS Access Credentials. Cette action ouvre le Amazon GameLift Servers plugin sur la page Configurer votre profil utilisateur.
2. Utilisez les boutons pour créer un nouveau AWS compte ou configurer un profil utilisateur pour un AWS compte que vous possédez déjà.
3. Si vous n'avez pas encore de profil utilisateur, vous êtes invité à saisir les informations du profil et à créer un nouveau profil. Saisissez les informations suivantes :
 - Un AWS compte. Si vous créez un nouveau AWS compte, utilisez le lien vers le AWS Management Console et suivez les instructions. Voir [Créer un AWS compte](#) pour plus de détails.
 - Un AWS utilisateur autorisé à utiliser Amazon GameLift Servers les autres AWS services requis. Consultez les instructions [Configurez un Compte AWS](#) relatives à la configuration d'un utilisateur AWS Identity and Access Management (IAM) avec des Amazon GameLift Servers autorisations et un accès programmatique avec des informations d'identification à long terme.
 - Informations d'identification de votre AWS utilisateur. Ces informations d'identification se composent d'un identifiant de clé d' AWS accès et d'une clé AWS secrète. Consultez [Obtenir vos clés d'accès](#) pour plus de détails.

- AWS région. Il s'agit d'un emplacement géographique dans lequel vous souhaitez créer vos AWS ressources pour l'hébergement. Pendant le développement, nous vous recommandons d'utiliser une région proche de votre emplacement physique. Choisissez une région dans la liste des [AWS régions prises en charge](#).
4. Si le plugin détecte des profils existants, il affiche une liste des profils disponibles. Sélectionnez un profil existant dans la liste ou choisissez Ajouter un autre profil pour en créer un nouveau.

Démarrez un profil utilisateur

Tous les profils doivent être amorcés pour être utilisés avec le Amazon GameLift Servers plugin. Le bootstrapping crée un compartiment Amazon S3 spécifique au profil. Il est utilisé pour stocker les configurations de projet, créer des artefacts et d'autres dépendances. Les buckets ne sont pas partagés entre d'autres profils.

Le bootstrapping implique la création de nouvelles AWS ressources et peut entraîner des coûts.

Pour démarrer votre profil, procédez comme suit :

1. Sur la page Informations d'identification d'AWS accès, vérifiez l'état d'amorçage du profil utilisateur que vous souhaitez utiliser. Si le statut d'amorçage du profil est « Inactif » et qu'aucun compartiment S3 n'est répertorié, vous devez démarrer le profil.
2. Sélectionnez le profil que vous souhaitez utiliser et choisissez le profil Bootstrap.
3. Attendez que le statut du bootstrap passe à « Actif ». Cette opération peut prendre quelques minutes.

Plugin pour Unity : configurez des tests locaux avec Amazon GameLift Servers Anywhere

Dans ce flux de travail, vous ajoutez du code de jeu client et serveur pour les Amazon GameLift Servers fonctionnalités et vous utilisez le plug-in pour désigner votre station de travail locale comme hôte du serveur de jeu de test. Lorsque vous avez terminé les tâches d'intégration, utilisez le plugin pour créer les composants du client et du serveur de votre jeu.

Pour démarrer le flux de travail Amazon GameLift Servers Anywhere :

- Dans le menu principal de l'éditeur Unity, choisissez Amazon GameLift Servers et sélectionnez Host with Anywhere. Cette action ouvre la page du plugin permettant de configurer votre jeu

avec une flotte @Anywhere. La page présente un processus en cinq étapes pour intégrer, créer et lancer les composants de votre jeu.

Définissez votre profil

Choisissez le profil que vous souhaitez utiliser lorsque vous suivez ce flux de travail. Le profil que vous sélectionnez a un impact sur toutes les étapes du flux de travail. Toutes les ressources que vous créez sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

1. Sélectionnez un profil dans la liste déroulante des profils disponibles. Si vous n'avez pas encore de profil ou si vous souhaitez en créer un nouveau, rendez-vous dans le Amazon GameLift Servers menu et choisissez Définir les profils de AWS compte.
2. Si le statut du bootstrap n'est pas « Actif », choisissez le profil Bootstrap et attendez que le statut passe à « Actif ».

Intégrez le code de votre jeu au SDK du serveur C#

Note

Si vous avez importé l'exemple de jeu, vous pouvez ignorer cette étape. Les exemples de ressources de jeu contiennent déjà le code serveur et client nécessaire.

Pour cette étape du flux de travail, vous devez mettre à jour le code du client et du serveur dans votre projet de jeu.

- * Les serveurs de jeu doivent être en mesure de communiquer avec le Amazon GameLift Servers service pour recevoir des instructions pour démarrer une session de jeu, fournir des informations de connexion à la session de jeu et signaler le statut.
- Les clients du jeu doivent être en mesure d'obtenir des informations sur les sessions de jeu, de rejoindre ou de démarrer des sessions de jeu, et d'obtenir des informations de connexion pour rejoindre une partie.

Intégrez le code de votre serveur

Si vous utilisez votre propre projet de jeu avec des scènes personnalisées, utilisez l'exemple de code fourni pour ajouter le code de serveur requis à votre projet de jeu :

1. Dans les fichiers de votre projet de jeu, ouvrez le `Assets/Scripts/Server` dossier. S'il n'existe pas, créez-le.
2. Accédez au GitHub repo [aws/ amazon-gamelift-plugin-unity](#) et ouvrez le chemin. `Samples~/SampleGame/Assets/Scripts/Server`
3. Localisez le fichier `GameLiftServer.cs` et copiez-le dans le `Server` dossier de votre projet de jeu. Lorsque vous créez un exécutable de serveur, utilisez ce fichier comme cible de génération.

L'exemple de code inclut les éléments minimaux requis suivants, qui utilisent le SDK du serveur Amazon GameLift Servers C# (version 5) :

- Initialise un client d'Amazon GameLift ServersAPI. L'`InitSDK()` appel avec les paramètres du serveur est obligatoire pour une flotte Amazon GameLift Servers Anywhere. Ces paramètres sont automatiquement définis pour être utilisés dans le plugin.
- Implémente les fonctions de rappel requises pour répondre aux demandes du Amazon GameLift Servers service, notamment `OnStartGameSessionOnProcessTerminate`, `onHealthCheck`.
- Appels `ProcessReady()` avec un port désigné pour avertir le Amazon GameLift Servers service lorsque le processus serveur est prêt à héberger des sessions de jeu.

Si vous souhaitez personnaliser l'exemple de code du serveur, consultez les ressources suivantes :

- [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)

Intégrez votre code client

Si vous utilisez votre propre projet de jeu avec des scènes personnalisées, vous devez intégrer les fonctionnalités de base dans votre client de jeu. Vous devez également ajouter des éléments d'interface utilisateur afin que les joueurs puissent se connecter et rejoindre une session de jeu. Utilisez l'API de service pour Amazon GameLift Servers (dans le AWS SDK) pour obtenir des informations sur les sessions de jeu, créer de nouvelles sessions de jeu ou rejoindre des sessions de jeu existantes,

Lorsque vous créez un client pour des tests locaux avec une flotte Anywhere, vous pouvez ajouter des appels directs au Amazon GameLift Servers service. Lorsque vous développez votre jeu pour l'hébergement dans le cloud, ou si vous envisagez d'utiliser des flottes Anywhere pour l'hébergement de production, vous devez créer un service principal côté client pour gérer toutes les communications entre les clients du jeu et le service. Amazon GameLift Servers

Pour l'Amazon GameLift Servers intégrer à votre code client, utilisez les ressources suivantes comme guide.

- Intégrez le client à la `GameLiftCoreApi` classe dans le GitHub dépôt `amazon-gamelift-plugin-unity aws/`. Cette classe fournit des contrôles pour l'authentification des joueurs et pour la récupération des informations de session de jeu.
- Consultez des exemples d'intégrations de jeux, disponibles dans le GitHub dépôt `amazon-gamelift-plugin-unity aws/`. `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- Suivez les instructions de la section `Ajouter Amazon GameLift Servers à votre client de jeu Unity`.

Pour les clients de jeu qui se connectent à une flotte Anywhere, votre client de jeu a besoin des informations suivantes. Le plugin met automatiquement à jour votre projet de jeu pour utiliser les ressources que vous avez créées dans le plugin.

- `FleetId` - L'identifiant unique de votre flotte Anywhere.
- `FleetLocation` - L'emplacement personnalisé de votre flotte Anywhere.
- `AwsRegion` - La AWS région où est hébergée votre flotte Anywhere. Il s'agit de la région que vous avez définie dans votre profil utilisateur.
- `ProfileName` - Un profil AWS d'identification sur votre machine locale qui permet d'accéder au AWS SDK pour Amazon GameLift Servers. Le client du jeu utilise ces informations d'identification pour authentifier les demandes adressées au Amazon GameLift Servers service.

Note

Le profil d'identification est généré par le plugin et stocké sur la machine locale. Par conséquent, vous devez exécuter le client sur la machine locale (ou sur une machine ayant le même profil).

Connectez-vous à une flotte n'importe où

Au cours de cette étape, vous désignez une flotte Anywhere à utiliser. Une flotte Anywhere définit un ensemble de ressources informatiques, qui peuvent être situées n'importe où, pour l'hébergement de serveurs de jeux.

- Si le AWS compte que vous utilisez actuellement possède des flottes Anywhere existantes, ouvrez le champ déroulant Nom de la flotte et choisissez une flotte. Cette liste déroulante affiche uniquement les flottes Anywhere de la AWS région correspondant au profil utilisateur actuellement actif.
- S'il n'existe aucune flotte existante, ou si vous souhaitez en créer une nouvelle, choisissez Create new Anywhere fleet et saisissez un nom de flotte.

Après avoir choisi une flotte Anywhere pour votre projet, vérifiez que l'état de Amazon GameLift Servers la flotte est actif et affiche l'identifiant de la flotte. Vous pouvez suivre la progression de cette demande dans le journal de sortie de l'éditeur Unity.

Enregistrer un ordinateur

Au cours de cette étape, vous enregistrez votre poste de travail local en tant que ressource de calcul dans le nouveau parc Anywhere.

1. Entrez un nom de calcul pour votre machine locale. Si vous ajoutez plusieurs ordinateurs au parc, les noms doivent être uniques.
2. Choisissez Register compute. Vous pouvez suivre la progression de cette demande dans le journal de sortie de l'éditeur Unity.

Le plugin enregistre votre station de travail locale avec l'adresse IP définie sur localhost (127.0.0.1). Ce paramètre suppose que vous exécuterez le client et le serveur du jeu sur la même machine.

En réponse à cette action, Amazon GameLift Servers vérifie qu'il peut se connecter au calcul et renvoie des informations sur le calcul nouvellement enregistré.

Lancer le jeu

Au cours de cette étape, vous créez les composants de votre jeu et vous les lancez pour jouer au jeu. Réalisez les tâches suivantes :

1. Configurez votre client de jeu. Au cours de cette étape, vous demandez au plugin de mettre à jour une `GameLiftClientSettings` ressource pour votre projet de jeu. Le plugin utilise cet actif pour stocker certaines informations dont votre client de jeu a besoin pour se connecter au Amazon GameLift Servers service.
 - a. Si vous n'avez pas importé et initialisé le jeu d'exemple, créez une nouvelle `GameLiftClientSettings` ressource. Dans le menu principal de l'éditeur Unity, choisissez Assets, Create, Amazon GameLift, Client Settings. Si vous créez plusieurs copies de `GameLiftClientSettings` votre projet, le plugin le détecte automatiquement et vous indique quel actif le plugin va mettre à jour.
 - b. Dans Launch Game, choisissez Configurer le client : appliquer les paramètres n'importe où. Cette action met à jour les paramètres de votre client de jeu pour utiliser la flotte Anywhere que vous venez de configurer.
2. Créez et exécutez votre client de jeu.
 - a. Créez un exécutable client en utilisant le processus de construction standard de Unity. Dans Fichier, Paramètres de compilation, basculez la plateforme sur Windows, Mac, Linux. Si vous avez importé l'exemple de jeu et initialisé les paramètres, la liste des builds et la cible des builds sont automatiquement mis à jour.
 - b. Lancez une ou plusieurs instances du nouveau fichier exécutable du client de jeu.
3. Lancez un serveur de jeu dans votre flotte Anywhere. Choisissez Serveur : Lancer le serveur dans l'éditeur. Cette tâche démarre un serveur live auquel votre client peut se connecter tant que l'éditeur Unity reste ouvert.
4. Démarrez ou rejoignez une session de jeu. Dans vos instances de client de jeu, utilisez l'interface utilisateur pour associer chaque client à une session de jeu. La manière dont vous procédez dépend de la manière dont vous avez ajouté des fonctionnalités au client.

Si vous utilisez l'exemple de client de jeu, il présente les caractéristiques suivantes :

- Un composant de connexion des joueurs. Lorsque vous vous connectez à un serveur de jeu d'une flotte Anywhere, aucune validation n'est requise par le joueur. Vous pouvez saisir n'importe quelle valeur pour rejoindre la session de jeu.
- Une interface utilisateur simple pour rejoindre le jeu. Lorsqu'un client tente de rejoindre une partie, il recherche automatiquement une session de jeu active avec un emplacement de joueur disponible. Si aucune session de jeu n'est disponible, le client demande une nouvelle session de jeu. Si une session de jeu est disponible, le client demande à rejoindre la session de jeu disponible.

Lorsque vous testez votre jeu avec plusieurs clients simultanés, le premier client démarre la session de jeu et les autres clients rejoignent automatiquement la session de jeu existante.

- Sessions de jeu avec des machines à sous pour quatre joueurs. Vous pouvez lancer jusqu'à quatre instances de client de jeu simultanément et elles rejoindront la même session de jeu.

Lancer depuis un exécutable du serveur (facultatif)

Vous pouvez créer et lancer le fichier exécutable de votre serveur de jeu pour le tester sur une flotte Anywhere.

1. Créez un exécutable de serveur en utilisant le processus de construction standard de Unity. Dans Fichier, Paramètres de construction, passez de la plate-forme à un serveur dédié et créez.
2. Obtenez un jeton d'authentification à court terme en appelant la commande AWS CLI [get-compute-auth-token](#) avec votre identifiant de flotte Anywhere et votre AWS région. L'identifiant du parc est affiché dans Connect to an Anywhere Fleet lorsque vous créez le parc. La AWS région s'affiche dans Set Your Profile lorsque vous sélectionnez votre profil actif.

```
aws gamelift get-compute-auth-token --fleet-id [your anywhere fleet ID] --region  
[your AWS region]
```

3. Lancez le nouveau fichier exécutable du serveur de jeu à partir d'une ligne de commande et transmettez un jeton d'authentification valide.

```
my_project.exe --authToken [token]
```

Plugin pour Unity : déployez votre jeu sur une EC2 flotte gérée

Dans ce flux de travail, vous utilisez le plugin pour préparer votre jeu à l'hébergement sur des ressources informatiques basées sur le cloud gérées par Amazon GameLift Servers. Vous ajoutez le code de jeu du client et du serveur pour les Amazon GameLift Servers fonctionnalités, puis vous téléchargez la version de votre serveur sur le Amazon GameLift Servers service pour l'hébergement. Lorsque ce flux de travail sera terminé, vous disposerez de serveurs de jeu exécutés dans le cloud et d'un client de jeu fonctionnel qui pourra s'y connecter.

Pour démarrer le EC2 flux de travail Amazon Amazon GameLift Servers géré :

- Dans le menu principal de l'éditeur Unity, choisissez Amazon GameLift Servers et sélectionnez Host with Managed EC2. Ce flux de travail présente un processus en six étapes pour intégrer, créer, déployer et lancer les composants de votre jeu.

Définissez votre profil

Choisissez le profil que vous souhaitez utiliser lorsque vous suivez ce flux de travail. Le profil que vous sélectionnez a un impact sur toutes les étapes du flux de travail. Toutes les ressources que vous créez sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

1. Sélectionnez un profil dans la liste déroulante des profils disponibles. Si vous n'avez pas encore de profil ou si vous souhaitez en créer un nouveau, rendez-vous dans le Amazon GameLift Servers menu et choisissez Définir les profils de AWS compte.
2. Si le statut du bootstrap n'est pas « Actif », choisissez le profil Bootstrap et attendez que le statut passe à « Actif ».

Intégrez votre jeu avec Amazon GameLift Servers

Pour cette tâche, vous devez mettre à jour le code du client et du serveur dans votre projet de jeu.

- Les serveurs de jeu doivent être en mesure de communiquer avec le Amazon GameLift Servers service pour recevoir des instructions pour démarrer une session de jeu, fournir des informations de connexion à la session de jeu et signaler le statut.
- Les clients du jeu doivent être en mesure d'obtenir des informations sur les sessions de jeu, de rejoindre ou de démarrer des sessions de jeu, et d'obtenir des informations de connexion pour rejoindre une partie.

Note

Si vous avez importé l'exemple de jeu, vous pouvez ignorer cette étape. Les exemples de ressources de jeu contiennent déjà le code serveur et client nécessaire.

Intégrez le code de votre serveur

Lorsque vous utilisez votre propre projet de jeu avec des scènes personnalisées, utilisez l'exemple de code fourni pour ajouter le code de serveur requis à votre projet de jeu. Si vous avez intégré votre projet de jeu à des fins de test à une flotte Anywhere, vous avez déjà suivi les instructions de cette étape.

1. Dans les fichiers de votre projet de jeu, ouvrez le `Assets/Scripts/Server` dossier. S'il n'existe pas, créez-le.
2. Accédez au GitHub repo [aws/ amazon-gamelift-plugin-unity](#) et ouvrez le chemin. `Samples~/SampleGame/Assets/Scripts/Server`
3. Localisez le fichier `GameLiftServer.cs` et copiez-le dans le `Server` dossier de votre projet de jeu. Lorsque vous créez un exécutable de serveur, utilisez ce fichier comme cible de génération.

L'exemple de code inclut les éléments minimaux requis suivants, qui utilisent le SDK du serveur Amazon GameLift Servers C# (version 5) :

- Initialise un client d'Amazon GameLift ServersAPI. L'appel `initSDK ()` avec les paramètres du serveur est requis pour une flotte Anywhere. Amazon GameLift Servers Ces paramètres sont automatiquement définis pour être utilisés dans le plugin.
- Implémente les fonctions de rappel requises pour répondre aux demandes du Amazon GameLift Servers service, notamment `OnStartGameSessionOnProcessTerminate`, `onHealthCheck`.
- Appels `ProcessReady ()` avec un port désigné pour avertir le Amazon GameLift Servers service lorsque le processus serveur est prêt à héberger des sessions de jeu.

Si vous souhaitez personnaliser l'exemple de code du serveur, consultez les ressources suivantes :

- [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)

Intégrez votre code client

Pour les clients de jeux qui se connectent à des serveurs de jeu basés sur le cloud, il est recommandé d'utiliser un service principal côté client pour passer des appels au Amazon GameLift Servers service, au lieu de passer les appels directement depuis le client du jeu.

Dans le flux de travail du plugin pour l'hébergement sur un EC2 parc géré, chaque scénario de déploiement inclut un service principal prédéfini qui inclut les composants suivants :

- Ensemble de fonctions Lambda et de tables DynamoDB utilisées pour demander des sessions de jeu et récupérer les informations relatives aux sessions de jeu. Ces composants utilisent une passerelle API comme proxy.
- Un groupe d'utilisateurs Amazon Cognito qui génère des joueurs uniques IDs et authentifie les connexions entre joueurs.

Pour utiliser ces composants, votre client de jeu a besoin de fonctionnalités lui permettant d'envoyer des demandes au service principal pour effectuer les opérations suivantes :

- Créez un utilisateur joueur dans le AWS groupe d'utilisateurs de Cognito et authentifiez-le.
- Participez à une session de jeu et recevez des informations de connexion.
- Rejoignez une partie en utilisant le matchmaking.

Utilisez les ressources suivantes comme guide.

- Intégrez le client à la [GameLiftCoreApi](#) classe dans le GitHub dépôt [amazon-gamelift-plugin-unityaws/](#). Cette classe fournit des contrôles pour l'authentification des joueurs et pour la récupération des informations de session de jeu.
- Pour voir les exemples d'intégrations de jeux, rendez-vous dans le GitHub dépôt [amazon-gamelift-plugin-unityaws/](#),. `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- [Ajoutez Amazon GameLift Servers à votre client de jeu.](#)

Sélectionnez le scénario de déploiement

Au cours de cette étape, vous choisissez la solution d'hébergement de jeux que vous souhaitez déployer à ce stade. Vous pouvez effectuer plusieurs déploiements de votre jeu, en utilisant n'importe quel scénario.

- Flotte à région unique : déployez votre serveur de jeu sur une flotte unique de ressources d'hébergement dans la région par défaut AWS du profil actif. Ce scénario constitue un bon point de départ pour tester l'intégration de votre serveur AWS et la configuration de la version du serveur. Il déploie les ressources suivantes :

- AWS flotte (à la demande) avec la version de votre serveur de jeu installée et en cours d'exécution.
 - Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
 - Autorisateur de passerelle API qui relie le groupe d'utilisateurs à APIs.
 - Web ACI pour limiter les appels excessifs des joueurs à la passerelle API.
 - Passerelle API + fonction Lambda permettant aux joueurs de demander une machine à sous. Cette fonction appelle `CreateGameSession()` si aucune fonction n'est disponible.
 - Passerelle API + fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.
- FlexMatch flotte : déploie votre serveur de jeu sur un ensemble de flottes et met en place un FlexMatch système de matchmaking avec des règles pour créer des parties entre joueurs. Ce scénario utilise un hébergement Spot à faible coût avec une structure multi-flottes et multi-sites pour une disponibilité durable. Cette approche est utile lorsque vous êtes prêt à commencer à concevoir un composant de matchmaking pour votre solution d'hébergement. Dans ce scénario, vous allez créer les ressources de base pour cette solution, que vous pourrez personnaliser ultérieurement selon vos besoins. Il déploie les ressources suivantes :
- FlexMatch configuration du matchmaking et règles de matchmaking définies pour accepter les demandes des joueurs et former des matchs.
 - Trois AWS flottes équipées de votre version de serveur de jeu sont installées et fonctionnent à plusieurs endroits. Comprend deux flottes Spot et une flotte à la demande en tant que sauvegarde.
 - AWS file d'attente de placement de sessions de jeu qui répond aux demandes de matchs proposés en trouvant la meilleure ressource d'hébergement possible (en fonction de la viabilité, du coût, de la latence des joueurs, etc.) et en démarrant une session de jeu.
 - Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
 - Autorisateur de passerelle API qui relie le groupe d'utilisateurs à APIs.
 - Web ACI pour limiter les appels excessifs des joueurs à la passerelle API.
 - Passerelle API + fonction Lambda permettant aux joueurs de demander une machine à sous. Cette fonction appelle `StartMatchmaking()`.
 - Passerelle API + fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.

- Tables Amazon DynamoDB pour stocker les tickets de matchmaking pour les joueurs et les informations relatives aux sessions de jeu.
- Rubrique SNS + Fonction Lambda pour GameSessionQueue gérer les événements.

Définissez les paramètres du jeu

Dans cette étape, vous décrivez le jeu vers lequel vous souhaitez le télécharger AWS.

- Nom du jeu : Donnez un nom significatif à votre projet de jeu. Ce nom est utilisé dans le plugin.
- Nom du parc : Donnez un nom significatif à votre EC2 parc géré. Amazon GameLift Servers utilise ce nom (ainsi que l'ID de flotte) lors de la liste des ressources dans la AWS console.
- Nom de la version : donnez un nom significatif à la version de votre serveur. AWS utilise ce nom pour faire référence à la copie de la version de votre serveur qui est téléchargée Amazon GameLift Servers et utilisée pour les déploiements.
- Paramètres de lancement : entrez les instructions facultatives à exécuter lors du lancement de l'exécutable du serveur sur une instance de EC2 flotte gérée. La longueur maximale est de 1024 caractères.
- Dossier du serveur de jeu : indiquez le chemin d'accès à un dossier local contenant la version de votre serveur.
- Fichier du serveur de jeu : Spécifiez le nom du fichier exécutable du serveur.

Scénario de déploiement

Au cours de cette étape, vous déployez votre jeu sur une solution d'hébergement cloud en fonction du scénario de déploiement que vous avez choisi. Ce processus peut prendre plusieurs minutes pour AWS valider la version de votre serveur, approvisionner les ressources d'hébergement, installer votre serveur de jeu, lancer les processus du serveur et le préparer à héberger des sessions de jeu.

Pour démarrer le déploiement, choisissez Deploy CloudFormation. Vous pouvez suivre l'état de votre hébergement de jeux ici. Pour obtenir des informations plus détaillées, vous pouvez vous connecter à la console AWS de gestion AWS et consulter les notifications d'événements. Assurez-vous de vous connecter en utilisant le même compte, le même utilisateur et AWS la même région que le profil utilisateur actif dans le plugin.

Lorsque le déploiement est terminé, votre serveur de jeu est installé sur une AWS EC2 instance. Au moins un processus serveur est en cours d'exécution et prêt à démarrer une session de jeu.

Lancer le client du jeu

Lorsque votre flotte est déployée avec succès, des serveurs de jeu sont désormais opérationnels et disponibles pour héberger des sessions de jeu. Vous pouvez maintenant créer votre client, le lancer, vous connecter pour rejoindre la session de jeu.

1. Configurez votre client de jeu. Au cours de cette étape, vous demandez au plugin de mettre à jour une `GameLiftClientSettings` ressource pour votre projet de jeu. Le plugin utilise cet actif pour stocker certaines informations dont votre client de jeu a besoin pour se connecter au Amazon GameLift Servers service.
 - a. Si vous n'avez pas importé et initialisé le jeu d'exemple, créez une nouvelle `GameLiftClientSettings` ressource. Dans le menu principal de l'éditeur Unity, choisissez Assets, Create, Amazon GameLift, Client Settings. Si vous créez plusieurs copies de `GameLiftClientSettings` votre projet, le plugin le détecte automatiquement et vous indique quel actif le plugin va mettre à jour.
 - b. Dans Launch Game, choisissez Configurer le client : Appliquer EC2 les paramètres gérés. Cette action met à jour les paramètres de votre client de jeu afin d'utiliser la EC2 flotte gérée que vous venez de déployer.
2. Créez votre client de jeu. Créez un exécutable client en utilisant le processus de construction standard de Unity. Dans Fichier, Paramètres de compilation, basculez la plateforme sur Windows, Mac, Linux. Si vous avez importé l'exemple de jeu et initialisé les paramètres, la liste des builds et la cible des builds sont automatiquement mis à jour.
3. Lancez le nouveau fichier exécutable du client de jeu. Pour commencer à jouer, lancez deux à quatre instances clientes et utilisez l'interface utilisateur de chacune pour rejoindre une session de jeu.

Si vous utilisez l'exemple de client de jeu, il présente les caractéristiques suivantes :

- Un composant de connexion des joueurs. Lorsque vous vous connectez à un serveur de jeu d'une flotte Anywhere, aucune validation n'est requise par le joueur. Vous pouvez saisir n'importe quelle valeur pour rejoindre la session de jeu.
- Une interface utilisateur simple pour rejoindre le jeu. Lorsqu'un client tente de rejoindre une partie, il recherche automatiquement une session de jeu active avec un emplacement de joueur disponible. Si aucune session de jeu n'est disponible, le client demande une nouvelle session de jeu. Si une session de jeu est disponible, le client demande à rejoindre la session de jeu disponible.

Lorsque vous testez votre jeu avec plusieurs clients simultanés, le premier client démarre la session de jeu et les autres clients rejoignent automatiquement la session de jeu existante.

- Sessions de jeu avec des machines à sous pour quatre joueurs. Vous pouvez lancer jusqu'à quatre instances de client de jeu simultanément et elles rejoindront la même session de jeu.

Plugin pour Unity : déployez votre jeu sur une flotte de conteneurs gérée

Utilisez ce flux de travail guidé pour créer une image de conteneur pour votre serveur de jeu et la déployer sur une solution d'hébergement basée sur des conteneurs. Lorsque vous avez terminé ce flux de travail avec succès, votre serveur de jeu conteneurisé fonctionne dans le cloud et vous pouvez utiliser le plugin pour démarrer un client de jeu, vous connecter à une session de jeu et jouer au jeu.

Avant de commencer

Ce flux de travail suppose que vous avez effectué les tâches suivantes.

- Intégrez le code de votre serveur de jeu au SDK Amazon GameLift Servers du serveur. notre serveur de jeu hébergé doit être en mesure de communiquer avec le Amazon GameLift Servers service afin qu'il puisse répondre aux demandes de démarrage de nouvelles sessions de jeu et signaler l'état des sessions de jeu. Si vous n'avez pas terminé cette tâche, nous vous recommandons de suivre d'abord le flux de travail du plugin Host with Anywhere. Pour obtenir des conseils sur la préparation du code de votre serveur de jeu, consultez [Intégrez le code de votre serveur](#). Pour une flotte de conteneurs gérée, vous devez intégrer votre jeu à la version 5.2 ou supérieure du SDK du serveur.

Note

Si vous utilisez la carte de démarrage du jeu, cette tâche est déjà terminée pour vous.

- Package le fichier exécutable de votre serveur de jeu pour qu'il fonctionne sous Linux.
- Rassemblez les fichiers à déployer avec la version de votre serveur de jeu. Sur votre machine locale, créez un répertoire de travail pour organiser les fichiers, qui seront intégrés à l'image conteneur de votre serveur de jeu. Il peut s'agir de dépendances au jeu, d'un script pour lancer des serveurs de jeu et d'autres processus lors du démarrage d'un conteneur, etc.
- Intégrez le code client de votre jeu à Amazon GameLift Servers. Pour effectuer cette tâche, vous pouvez ajouter un exemple de ressource (inclus dans le plugin) déjà intégré. Pour obtenir des conseils sur la préparation du code client de votre jeu, consultez [Intégrez votre code client](#).

- Installez Docker sur votre machine locale. Vous devez installer cet outil si vous souhaitez que le plugin crée des images de conteneur pour vous et les envoie vers un référentiel ECR. Vous pouvez également effectuer ces tâches manuellement et demander au plugin d'utiliser une image de conteneur existante. Pour plus d'informations sur la création manuelle de votre image, voir [Création d'une image de conteneur pour Amazon GameLift Servers](#).

Pour démarrer le flux de travail des conteneurs Amazon GameLift Servers gérés :

- Dans la barre d'outils principale de l'éditeur Unity, choisissez le Amazon GameLift Servers menu, puis sélectionnez Managed Containers. Cette action ouvre la page du plugin Host with Managed Containers, qui présente un step-by-step processus permettant de créer une image de conteneur avec la version de votre serveur de jeu, de la déployer dans une flotte de conteneurs et de lancer votre jeu.

Étape 0 : Définissez votre profil

Cette section affiche le profil utilisateur actuellement sélectionné. Vérifiez que le profil utilisateur actuel est celui que vous souhaitez utiliser pour ce flux de travail. Toutes les ressources que vous créez dans ce flux de travail sont associées au AWS compte du profil et sont placées dans la AWS région par défaut du profil. Les autorisations de l'utilisateur du profil déterminent votre accès aux AWS ressources et aux actions.

Vous devrez peut-être modifier le profil utilisateur sélectionné si :

- Aucun profil n'est actuellement sélectionné.
- Vous souhaitez sélectionner un autre profil ou créer un nouveau profil.
- Vous devez amorcer le profil sélectionné (si le statut du bootstrap est inactif).

Pour définir ou modifier le profil utilisateur sélectionné

- Dans le Amazon GameLift Servers menu, choisissez Open AWS Access Credentials.

Étape 1 : Évaluer l'état de préparation du conteneur

Avant de déployer votre serveur de jeu sur un parc de conteneurs, vous devez le regrouper dans une image de conteneur et le stocker dans un référentiel Amazon ECR. Le plugin peut effectuer ces

tâches pour vous ou vous pouvez les effectuer manuellement. Au cours de cette étape, fournissez des informations sur le statut de votre image de conteneur et du référentiel ECR.

Utilisez les questions d'évaluation pour indiquer au plugin les étapes à suivre :

- Créez une nouvelle image de conteneur. Si vous choisissez cette option, l'étape suivante vous demandera l'emplacement du répertoire de compilation de votre serveur de jeu et de l'exécutable de compilation. Le plugin utilise un modèle Dockerfile (fourni par Amazon GameLift Servers) et le configure automatiquement pour votre jeu. Vous pouvez consulter le modèle sur [Créez une image de conteneur pour Amazon GameLift Servers](#). Après avoir choisi cette option, indiquez où vous souhaitez que le plugin stocke la nouvelle image :
- Créez un nouveau référentiel Amazon ECR et envoyez-y l'image du conteneur. Le plugin crée un dépôt ECR privé en utilisant le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné.
- Transférez l'image du conteneur vers un référentiel Amazon ECR créé précédemment. Si vous choisissez cette option, l'étape suivante vous invitera à sélectionner un référentiel Amazon ECR existant dans une liste. La liste inclut tous les référentiels Amazon ECR pour le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné. Vous pouvez sélectionner un dépôt public ou privé.
- Utilisez une image de conteneur existante. Si vous avez créé une image manuellement, nous vous recommandons d'utiliser le modèle Dockerfile fourni par Amazon GameLift Servers, disponible sur [Créez une image de conteneur pour Amazon GameLift Servers](#). Après avoir choisi cette option, indiquez où se trouve l'image.
- Une image générée par Docker stockée localement. Si vous choisissez cette option, le plugin crée un nouveau référentiel privé Amazon ECR et y envoie le fichier image local. L'étape suivante vous demandera un identifiant d'image, que le plugin utilisera pour localiser le fichier image.
- Une image de conteneur déjà stockée dans un référentiel Amazon ECR. Si vous choisissez cette option, l'étape suivante vous invitera à sélectionner un référentiel Amazon ECR et une image existants dans une liste. La liste inclut tous les référentiels Amazon ECR pour le AWS compte et par défaut Région AWS dans le profil utilisateur que vous avez sélectionné. Vous pouvez sélectionner un dépôt public ou privé.

Étape 2 : configurer le déploiement de l'image

Au cours de cette étape, fournissez les informations dont le plugin a besoin pour déployer votre image de conteneur dans une flotte de conteneurs. Cette étape demande les informations suivantes :

- L'emplacement de la version de votre serveur de jeu, de l'image de conteneur ou du référentiel Amazon ECR, en fonction de vos sélections à l'étape 1.
- Le scénario à utiliser pour le déploiement de vos conteneurs gérés.
- Paramètres de déploiement facultatifs. Cette section contient les paramètres de configuration que le plugin utilise par défaut. Vous pouvez les modifier ou conserver les valeurs par défaut.
 - Le nom du jeu est défini sur le nom de votre projet de jeu par défaut. Toutes les AWS ressources créées par le plugin font référence à la valeur du nom du jeu.
 - La plage de ports, la limite de mémoire et la limite de vCPU sont des paramètres de configuration pour le parc de conteneurs. Pour plus d'informations sur la personnalisation de ces valeurs, reportez-vous [Configuration des connexions réseau](#) aux sections pour la plage de ports de connexion et [Définissez des limites de ressources](#) pour les limites de ressources.
 - La balise d'image de conteneur est utilisée pour classer vos images de conteneur dans Amazon ECR. La valeur par défaut est `unity-gamelift-plugin`.

Options de scénario de déploiement

Flotte de conteneurs à région unique

Ce scénario déploie votre serveur de jeu sur une seule flotte de conteneurs. C'est un bon point de départ pour tester l'intégration de votre serveur AWS et la configuration de votre conteneur. Il déploie les ressources suivantes.

- Amazon GameLift ServersLa définition d'un groupe de conteneurs décrit comment déployer et exécuter vos images de conteneurs sur une flotte de conteneurs.
- Amazon GameLift Serversflotte de conteneurs (à la demande) avec le conteneur de votre serveur de jeu installé et en cours d'exécution, avec alias.
- Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
- Autorisateur API Gateway qui relie le groupe d'utilisateurs à APIs.
- Liste de contrôle d'accès Web (ACL) pour limiter les appels excessifs des joueurs à API Gateway.

- Service principal pour envoyer des demandes au Amazon GameLift Servers service au nom des clients du jeu, par exemple pour demander des sessions de jeu et rejoindre des jeux :
- API Gateway + Fonction Lambda permettant aux joueurs de demander un créneau de session de jeu. Cette fonction appelle `CreateGameSession()` si aucun emplacement ouvert n'est disponible.
- API Gateway + Fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.

Flotte de conteneurs unirégionale avec FlexMatch

Ce scénario déploie votre serveur de jeu sur une flotte de conteneurs, configure le placement des sessions de jeu et met en place FlexMatch le matchmaking. Ce scénario est utile lorsque vous êtes prêt à commencer à concevoir un système de matchmaking personnalisé pour votre solution d'hébergement. Utilisez ce scénario pour créer les ressources de base de cette solution, que vous pourrez personnaliser ultérieurement selon vos besoins. Il déploie les ressources suivantes :

- Amazon GameLift Servers définition de groupe de conteneurs qui décrit comment déployer et exécuter vos images de conteneurs sur une flotte de conteneurs.
- Amazon GameLift Servers flotte de conteneurs (à la demande) avec le conteneur de votre serveur de jeu installé et en cours d'exécution, avec alias.
- FlexMatch configuration du matchmaking et règles de matchmaking définies pour accepter les demandes des joueurs et former des matchs.
- Amazon GameLift Servers file d'attente de session de jeu qui répond aux demandes de matchs proposés en trouvant la meilleure ressource d'hébergement possible (en fonction de la viabilité, du coût, de la latence des joueurs, etc.) et en démarrant une session de jeu.
- Groupe d'utilisateurs et client Amazon Cognito pour permettre aux joueurs de s'authentifier et de démarrer une partie.
- Autorisateur API Gateway qui relie le groupe d'utilisateurs à APIs.
- Liste de contrôle d'accès Web (ACL) pour limiter les appels excessifs des joueurs à API Gateway.
- Service principal pour envoyer des demandes au Amazon GameLift Servers service au nom des clients du jeu, par exemple pour demander des sessions de jeu et rejoindre des jeux :
 - API Gateway + Fonction Lambda permettant aux joueurs de demander un créneau de session de jeu. Cette fonction appelle `StartMatchmaking()` si aucun emplacement ouvert n'est disponible.

- API Gateway + Fonction Lambda permettant aux joueurs d'obtenir des informations de connexion pour leur demande de jeu.
- Tables DynamoDB pour stocker les tickets de matchmaking pour les joueurs et les informations sur les sessions de jeu.
- Rubrique Amazon SNS + Fonction Lambda pour gérer les événements. GameSessionQueue

Déployer une flotte de conteneurs

Lorsque la configuration de votre flotte est terminée, cliquez sur le bouton Déployer une flotte de conteneurs pour démarrer le déploiement. Ce processus peut prendre plusieurs minutes pendant que le plugin crée une image de conteneur et la transmet à ECR, provisionne les ressources d'hébergement pour la flotte de conteneurs, déploie la flotte et d'autres AWS ressources pour le scénario de solution d'hébergement sélectionné.

Lorsque vous démarrez un déploiement, vous pouvez suivre la progression de chaque étape. Selon votre configuration, les étapes peuvent inclure les suivantes :

- Configuration de l'image du conteneur
- Création d'un référentiel Amazon ECR
- Création d'une image et diffusion vers Amazon ECR
- Création d'une définition de groupe de conteneurs
- Création d'une flotte de conteneurs

Pour obtenir des informations de déploiement plus détaillées, choisissez Afficher dans AWS la console de gestion. Lorsque la flotte de conteneurs atteint le statut actif, elle gère activement des conteneurs avec des processus de serveur prêts à héberger des sessions de jeu.

Une fois le déploiement terminé, vous disposez d'une flotte de conteneurs fonctionnelle prête à accueillir des sessions de jeu et à accepter les connexions entre joueurs.

Vous ne pouvez pas arrêter un déploiement en cours. Si le déploiement entre dans un mauvais état ou échoue, vous pouvez recommencer à zéro en utilisant l'option Réinitialiser le déploiement.

Lancer le client

À ce stade, vous avez terminé toutes les tâches pour lancer et jouer à votre jeu multijoueur hébergé par Amazon GameLift Servers. Pour jouer à votre jeu, choisissez Start Client pour lancer une instance locale de votre client de jeu.

- Si vous avez déployé le scénario à flotte unique, ouvrez une instance de votre client de jeu avec un joueur et accédez à la carte du serveur pour vous déplacer. Vous pouvez ouvrir une deuxième instance du client de jeu pour ajouter un deuxième joueur à la même carte de jeu du serveur.
- Si vous avez déployé le FlexMatch scénario, la solution d'hébergement attend qu'au moins deux clients du jeu fassent des demandes de matchmaking. Ouvrez au moins deux instances de votre client de jeu avec un joueur. Les deux joueurs seront jumelés et invités à rejoindre une session de jeu pour le match.

Mettre à jour une flotte de conteneurs

Si vous avez déployé avec succès une solution d'hébergement de conteneurs gérés, vous pouvez utiliser la fonctionnalité de déploiement des mises à jour. Cette option vous permet de mettre à jour les paramètres de configuration d'une flotte de conteneurs déployée, sans avoir à créer une nouvelle flotte.

Lorsque vous mettez à jour un déploiement, vous pouvez déployer une image de conteneur avec une version de serveur de jeu différente, modifier le référentiel Amazon ECR, choisir un autre scénario de déploiement et personnaliser les paramètres de configuration facultatifs.

Lorsque vous êtes prêt à déployer vos modifications, choisissez Mettre à jour. Le temps requis pour une mise à jour de déploiement est similaire à celui d'un déploiement complet. Pour obtenir des informations détaillées sur le déploiement, choisissez Afficher dans AWS la console de gestion.

Nettoyez les ressources déployées

Il est recommandé de nettoyer les AWS ressources de votre solution de conteneurs gérés dès que vous n'en avez plus besoin. Il se peut que vous continuiez à encourir des coûts pour ces ressources si vous ne les supprimez pas.

Supprimez les ressources suivantes :

- Pile de ressources de conteneurs gérée. Les ressources de cette pile dépendent du scénario de déploiement que vous avez sélectionné. Pour supprimer l'intégralité de la pile, utilisez la AWS CloudFormation console. Les piles générées à partir du Amazon GameLift Servers plugin utilisent la convention de dénomination suivante :GameLiftPluginForUnity-{GameName}-Containers. Attendez que le processus de suppression de la pile soit terminé avant de lancer un nouveau déploiement de conteneurs gérés dans le plugin. Pour plus d'informations, consultez [Supprimer une pile de la CloudFormation console](#).

- **Référentiel Amazon ECR.** Si vous avez utilisé le plugin pour créer un référentiel pour votre image de conteneur, vous souhaitez peut-être supprimer tous les référentiels dont vous n'avez plus besoin. Il n'est pas nécessaire de supprimer un référentiel avant de réinitialiser un déploiement de conteneurs gérés. Si vous mettez à jour ou réinitialisez un déploiement, le plugin utilisera automatiquement le même référentiel à moins qu'il ne soit demandé d'en utiliser un autre. Pour plus d'informations, consultez [Supprimer un référentiel privé dans Amazon ECR](#).

Amazon GameLift Serversplugin pour Unity (SDK de serveur 4.x)

Note

Cette rubrique fournit des informations relatives à une version antérieure du Amazon GameLift Servers plugin pour Unity. La version 1.0.0 (publiée en 2021) utilise le SDK du serveur pour Amazon GameLift Servers 4.x ou version antérieure. Pour obtenir de la documentation sur la dernière version du plugin, qui utilise le SDK 5.x du serveur et prend en charge Amazon GameLift Servers Anywhere, voir. [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#)

Amazon GameLift Serversfournit des outils pour préparer l'exécution de vos serveurs de jeux multijoueursAmazon GameLift Servers. Le Amazon GameLift Servers plugin pour Unity facilite l'intégration Amazon GameLift Servers dans vos projets de jeux Unity et le déploiement de Amazon GameLift Servers ressources pour l'hébergement dans le cloud. Utilisez le plugin pour Unity afin d'accéder à des AWS CloudFormation modèles pour des scénarios de jeu courants Amazon GameLift Servers APIs et de les déployer.

Après avoir configuré le plugin, vous pouvez essayer l'[exemple Amazon GameLift Servers Unity](#) sur GitHub.

Rubriques

- [Intégration Amazon GameLift Servers à un projet de serveur de jeu Unity](#)
- [Intégration Amazon GameLift Servers à un projet client de jeu Unity](#)
- [Installation et configuration du plugin](#)
- [Testez votre jeu localement](#)
- [Déployer un scénario](#)
- [Intégrer des jeux Amazon GameLift Servers dans Unity](#)

- [Importez et lancez un exemple de jeu](#)

Intégration Amazon GameLift Servers à un projet de serveur de jeu Unity

Note

Cette rubrique fournit des informations relatives à une version antérieure du Amazon GameLift Servers plugin pour Unity. La version 1.0.0 (publiée en 2021) utilise le SDK du serveur pour Amazon GameLift Servers 4.x ou version antérieure. Pour obtenir de la documentation sur la dernière version du plugin, qui utilise le SDK 5.x du serveur et prend en charge Amazon GameLift Servers Anywhere, voir. [Amazon GameLift Servers plugin pour Unity \(SDK serveur 5.x\)](#)

Cette rubrique vous aide à préparer votre serveur de jeu personnalisé pour l'hébergement sur Amazon GameLift Servers. Le serveur de jeu doit être en mesure de Amazon GameLift Servers notifier son statut, de démarrer et d'arrêter des sessions de jeu lorsqu'il y est invité, et d'effectuer d'autres tâches. Pour plus d'informations, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Prérequis

Avant d'intégrer votre serveur de jeu, effectuez les tâches suivantes :

- [Configurer un rôle de service IAM pour Amazon GameLift Servers](#)
- [Installation et configuration du plugin](#)

Configuration d'un nouveau processus de serveur

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Configurez la communication avec le processus serveur Amazon GameLift Servers et signalez qu'il est prêt à héberger une session de jeu.

1. Initialisez le SDK du serveur en appelant. `InitSDK()`
2. Pour préparer le serveur à accepter une session de jeu, appelez `ProcessReady()` avec le port de connexion et les détails de l'emplacement de la session de jeu. Incluez les noms des fonctions de rappel invoquées par le Amazon GameLift Servers service, telles que `OnGameSession()`, `OnGameSessionUpdate()`, `OnProcessTerminate()`, `OnHealthCheck()` Amazon GameLift ServersLe rappel peut prendre quelques minutes.
3. Amazon GameLift Serversmet à jour l'état du processus du serveur sur `ACTIVE`.
4. Amazon GameLift Serversappelle `onHealthCheck` périodiquement.

L'exemple de code suivant montre comment configurer un processus serveur simple avec Amazon GameLift Servers.

```
//initSDK
var initSDKOutcome = GameLiftServerAPI.InitSDK();

//processReady
// Set parameters and call ProcessReady
var processParams = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    // Examples of log and error files written by the game server
    new LogParameters(new List<string>()
        {
            "C:\\game\\logs",
            "C:\\game\\error"
        })
);

var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);

// Implement callback functions
void OnGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

```
void OnProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();
}

bool OnHealthCheck()
{
    bool isHealthy;
    // complete health evaluation within 60 seconds and set health
    return isHealthy;
}
```

Démarrer une session de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Une fois l'initialisation du jeu terminée, vous pouvez démarrer une session de jeu.

1. Implémentez la fonction de rappel. `onStartGameSession` Amazon GameLift Servers invoque cette méthode pour démarrer une nouvelle session de jeu sur le processus du serveur et recevoir les connexions des joueurs.
2. Pour activer une session de jeu, appelez `ActivateGameSession()`. Pour plus d'informations sur le SDK, consultez [SDK du serveur C# pour Amazon GameLift Servers 4.x -- Actions](#).

L'exemple de code suivant montre comment démarrer une session de jeu avec Amazon GameLift Servers.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    ...
    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

```
}
```

Mettre fin à une session de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Avertir Amazon GameLift Servers de la fin d'une session de jeu. Il est recommandé d'arrêter les processus du serveur une fois les sessions de jeu terminées afin de recycler et d'actualiser les ressources d'hébergement.

1. Configurez une fonction nommée `onProcessTerminate` pour recevoir les demandes Amazon GameLift Servers et les appeler `ProcessEnding()`.
2. L'état du processus passe à `TERMINATED`.

L'exemple suivant décrit comment mettre fin à un processus pour une session de jeu.

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();

if (processReadyOutcome.Success)
    Environment.Exit(0);

// otherwise, exit with error code
Environment.Exit(errorCode);
```

Créer un serveur, le compiler et le télécharger sur Amazon GameLift Servers

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Après avoir intégré votre serveur de jeu Amazon GameLift Servers, téléchargez les fichiers de build dans une flotte afin de Amazon GameLift Servers pouvoir les déployer pour l'hébergement

de jeux. Pour plus d'informations sur le transfert de votre serveur vers Amazon GameLift Servers, consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#).

Intégration Amazon GameLift Servers à un projet client de jeu Unity

Note

Cette rubrique fournit des informations relatives à une version antérieure du Amazon GameLift Servers plugin pour Unity. La version 1.0.0 (publiée en 2021) utilise le SDK du serveur pour Amazon GameLift Servers 4.x ou version antérieure. Pour obtenir de la documentation sur la dernière version du plugin, qui utilise le SDK 5.x du serveur et prend en charge Amazon GameLift Servers Anywhere, voir. [Amazon GameLift Servers plugin pour Unity \(SDK serveur 5.x\)](#)

Cette rubrique vous aide à configurer un client de jeu pour qu'il se connecte à des sessions de jeu Amazon GameLift Servers hébergées via un service principal. Amazon GameLift Servers APIs À utiliser pour initier le matchmaking, demander le placement d'une session de jeu, et plus encore.

Ajoutez du code au projet de service principal pour permettre la communication avec le Amazon GameLift Servers service. Un service principal gère toutes les communications du client du jeu avec le GameLift service. Pour plus d'informations sur les services principaux, consultez.

Un serveur principal gère les tâches suivantes du client de jeu :

- Personnalisez l'authentification pour vos joueurs.
- Demande d'informations sur les sessions de jeu actives à partir du service Amazon GameLift Servers.
- Créez une nouvelle session de jeu.
- Ajoutez un joueur à une session de jeu existante.
- Supprimer un joueur d'une session de jeu existante.

Rubriques

- [Prérequis](#)
- [Initialisation d'un client de jeu](#)
- [Créer une session de jeu sur une flotte spécifique](#)

- [Ajouter des joueurs aux sessions de jeu](#)
- [Supprimer un joueur d'une session de jeu](#)

Prérequis

Avant de configurer la communication entre le serveur de jeu et le Amazon GameLift Servers client, effectuez les tâches suivantes :

- [Configurez un Compte AWS](#)
- [Installation et configuration du plugin](#)
- [Intégration Amazon GameLift Servers à un projet de serveur de jeu Unity](#)
- [Configuration d'un parc d'hébergement avec Amazon GameLift Servers](#)

Initialisation d'un client de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Ajoutez du code pour initialiser un client de jeu. Exécutez ce code au lancement, il est nécessaire pour les autres Amazon GameLift Servers fonctions.

1. `InitialiserAmazonGameLiftClient`. Appelez `AmazonGameLiftClient` avec une configuration client par défaut ou une configuration personnalisée. Pour plus d'informations sur la configuration d'un client, consultez [Configuration Amazon GameLift Servers sur un service principal](#).
2. Générez un identifiant de joueur unique pour que chaque joueur puisse se connecter à une session de jeu. Pour de plus amples informations, veuillez consulter [Générer un joueur IDs](#).

Les exemples suivants montrent comment configurer un Amazon GameLift Servers client.

```
public class GameLiftClient
{
    private GameLift gl;
    //A sample way to generate random player IDs.
    bool includeBrackets = false;
    bool includeDashes = true;
```

```
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
includeDashes);

private Amazon.GameLift.Model.PlayerSession psession = null;
public AmazonGameLiftClient aglc = null;

public void CreateGameLiftClient()
{
    //Access Amazon GameLift Servers service by setting up a configuration.
    //The default configuration specifies a location.
    var config = new AmazonGameLiftConfig();
    config.RegionEndpoint = Amazon.RegionEndpoint.USEast1;

    CredentialProfile profile = null;
    var nscf = new SharedCredentialsFile();
    nscf.TryGetProfile(profileName, out profile);
    AWSCredentials credentials = profile.GetAWSCredentials(null);
    //Initialize Amazon GameLift Servers Client with default client
configuration.
    aglc = new AmazonGameLiftClient(credentials, config);
}
}
```

Créer une session de jeu sur une flotte spécifique

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Ajoutez du code pour démarrer de nouvelles sessions de jeu sur vos flottes déployées et les mettre à disposition des joueurs. Après avoir Amazon GameLift Servers créé la nouvelle session de jeu et renvoyé unGameSession, vous pouvez y ajouter des joueurs.

- Faites une demande pour une nouvelle session de jeu.
 - Si votre jeu utilise des flottes, appelez CreateGameSession() avec un identifiant de flotte ou un alias, un nom de session et le nombre maximum de joueurs simultanés pour le jeu.

- Si votre jeu utilise des files d'attente, appelez `StartGameSessionPlacement()`.

L'exemple suivant montre comment créer une session de jeu.

```
public Amazon.GameLift.Model.GameSession()
{
    var cgsreq = new Amazon.GameLift.Model.CreateGameSessionRequest();
    //A unique identifier for the alias with the fleet to create a game session in.
    cgsreq.AliasId = aliasId;
    //A unique identifier for a player or entity creating the game session
    cgsreq.CreatorId = playerId;
    //The maximum number of players that can be connected simultaneously to the game
    session.
    cgsreq.MaximumPlayerSessionCount = 4;

    //Prompt an available server process to start a game session and retrieves
    connection information for the new game session
    Amazon.GameLift.Model.CreateGameSessionResponse cgsres =
    aglc.CreateGameSession(cgsreq);
    string gsid = cgsres.GameSession != null ? cgsres.GameSession.GameSessionId : "N/A";
    Debug.Log((int)cgsres.HttpStatusCode + " GAME SESSION CREATED: " + gsid);
    return cgsres.GameSession;
}
```

Ajouter des joueurs aux sessions de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Après avoir Amazon GameLift Servers créé la nouvelle session de jeu et renvoyé un `GameSession` objet, vous pouvez y ajouter des joueurs.

1. Réservez une place de joueur dans une session de jeu en créant une nouvelle session de joueur. Utilisez `CreatePlayerSession` ou `CreatePlayerSessions` avec l'identifiant de session de jeu et un identifiant unique pour chaque joueur.

2. Connectez-vous à la session de jeu. Récupérez l'PlayerSession objet pour obtenir les informations de connexion de la session de jeu. Vous pouvez utiliser ces informations pour établir une connexion directe avec le processus du serveur :
 - a. Utilisez le port spécifié et le nom DNS ou l'adresse IP du processus serveur.
 - b. Utilisez le nom DNS et le port de vos flottes. Le nom et le port DNS sont obligatoires si la génération de certificats TLS est activée dans vos flottes.
 - c. Référencez l'identifiant de session du joueur. L'identifiant de session du joueur est requis si votre serveur de jeu valide les connexions entrantes des joueurs.

Les exemples suivants montrent comment réserver une place de joueur dans une session de jeu.

```
public Amazon.GameLift.Model.PlayerSession
CreatePlayerSession(Amazon.GameLift.Model.GameSession gsession)
{
    var cpsreq = new Amazon.GameLift.Model.CreatePlayerSessionRequest();
    cpsreq.GameSessionId = gsession.GameSessionId;
    //Specify game session ID.
    cpsreq.PlayerId = playerId;
    //Specify player ID.
    Amazon.GameLift.Model.CreatePlayerSessionResponse cpsres =
aglc.CreatePlayerSession(cpsreq);
    string psid = cpsres.PlayerSession != null ? cpsres.PlayerSession.PlayerSessionId :
"N/A";
    return cpsres.PlayerSession;
}
```

Le code suivant montre comment connecter un joueur à la session de jeu.

```
public bool ConnectPlayer(int playerId, string playerSessionId)
{
    //Call ConnectPlayer with player ID and player session ID.
    return server.ConnectPlayer(playerId, playerSessionId);
}
```

Supprimer un joueur d'une session de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Vous pouvez retirer les joueurs de la session de jeu lorsqu'ils quittent le jeu.

1. Informez le Amazon GameLift Servers service qu'un joueur s'est déconnecté du processus serveur. Appelez `RemovePlayerSession` avec l'identifiant de session du joueur.
2. Vérifiez que cela `RemovePlayerSession` revient `Success`. Amazon GameLift Servers modifie ensuite l'emplacement de joueur qui sera disponible, qui Amazon GameLift Servers peut être attribué à un nouveau joueur.

L'exemple suivant montre comment supprimer une session de joueur.

```
public void DisconnectPlayer(int playerIdX)
{
    //Receive the player session ID.
    string playerSessionId = playerSessions[playerIdx];
    var outcome = GameLiftServerAPI.RemovePlayerSession(playerSessionId);
    if (outcome.Success)
    {
        Debug.Log (":) PLAYER SESSION REMOVED");
    }
    else
    {
        Debug.Log(":(PLAYER SESSION REMOVE FAILED. RemovePlayerSession()
        returned " + outcome.Error.ToString());
    }
}
```

Installation et configuration du plugin

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Cette section explique comment télécharger, installer et configurer le Amazon GameLift Servers plugin pour Unity, version 1.0.0.

Prérequis

- Unity pour Windows 2019.4 LTS, Windows 2020.3 LTS ou Unity pour macOS
- Version actuelle de Java
- Version actuelle de .NET 4.x

Pour télécharger et installer le plugin pour Unity

1. Téléchargez le Amazon GameLift Servers plugin pour Unity. Vous pouvez trouver la dernière version sur la page du [référentiel du Amazon GameLift Servers plugin pour Unity](#). Dans la [dernière version](#), choisissez Assets, puis téléchargez le `com.amazonaws.gamelift-version.tgz` fichier.
2. Lancez Unity et choisissez un projet.
3. Dans la barre de navigation supérieure, sous Window, choisissez Package Manager :
4. Dans l'onglet Gestionnaire de packages, choisissez +, puis choisissez Ajouter un package depuis une archive... :
5. Dans la fenêtre Sélectionner les packages sur le disque, accédez au `com.amazonaws.gamelift` dossier, choisissez le fichier `com.amazonaws.gamelift-version.tgz` , puis choisissez Ouvrir :
6. Une fois que Unity a chargé le plug-in, Amazon GameLift Servers apparaît sous la forme d'un nouvel élément dans le menu Unity. L'installation et la recompilation des scripts peuvent

prendre quelques minutes. L'onglet Paramètres du Amazon GameLift Servers plugin s'ouvre automatiquement.

7. Dans le volet SDK, choisissez Utiliser .NET 4.x.

Une fois configuré, le statut passe de Non configuré à Configuré.

Testez votre jeu localement

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Utilisez Amazon GameLift Servers Local pour exécuter Amazon GameLift Servers sur votre appareil local. Vous pouvez utiliser Amazon GameLift Servers Local pour vérifier les modifications de code en quelques secondes, sans connexion réseau.

Configuration des tests locaux

1. Dans la fenêtre du plugin pour Unity, choisissez l'onglet Test.
2. Dans le volet Test, choisissez Download Amazon GameLift Servers Local. Le plugin pour Unity ouvre une fenêtre de navigateur et télécharge le `GameLift_06_03_2021.zip` fichier dans votre dossier de téléchargements.

Le téléchargement inclut le SDK du serveur C#, les fichiers source .NET et un composant .NET compatible avec Unity.

3. Décompressez le fichier `GameLift_06_03_2021.zip` téléchargé.
4. Dans la fenêtre Paramètres du Amazon GameLift Servers plugin, choisissez Amazon GameLift ServersLocal Path, accédez au dossier décompressé, choisissez le fichier **GameLiftLocal.jar**, puis choisissez Ouvrir.

Une fois configuré, le statut des tests locaux passe de Non configuré à Configuré.

5. Vérifiez l'état du JRE. Si le statut est Non configuré, choisissez Télécharger JRE et installez la version Java recommandée.

Après avoir installé et configuré l'environnement Java, le statut passe à Configuré.

Lancez votre jeu local

1. Dans l'onglet Plugin pour Unity, choisissez l'onglet Test.
2. Dans le volet Test, choisissez Open Local Test UI.
3. Dans la fenêtre de test local, spécifiez le chemin exécutable du serveur. Sélectionnez... pour sélectionner le chemin et le nom du fichier exécutable de votre application serveur.
4. Dans la fenêtre de test local, spécifiez un port GL local.
5. Choisissez Deploy & Run pour déployer et exécuter le serveur.
6. Pour arrêter votre serveur de jeu, choisissez Arrêter ou fermez les fenêtres du serveur de jeu.

Déployer un scénario

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Un scénario utilise un AWS CloudFormation modèle pour créer les ressources dont vous avez besoin pour déployer une solution d'hébergement cloud pour votre jeu. Cette section décrit les scénarios Amazon GameLift Servers proposés et explique comment les utiliser.

Prérequis

Pour déployer le scénario, vous avez besoin d'un rôle IAM pour le Amazon GameLift Servers service. Pour plus d'informations sur la création d'un rôle pour Amazon GameLift Servers, consultez [Configurez un Compte AWS](#).

Chaque scénario nécessite des autorisations d'accès aux ressources suivantes :

- Amazon GameLift Servers
- Amazon S3
- AWS CloudFormation
- API Gateway
- AWS Lambda
- AWS WAFV2

- Amazon Cognito

Scénarios

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Le Amazon GameLift Servers plug-in pour Unity inclut les scénarios suivants :

Authentification uniquement

Ce scénario crée un service de backend de jeu qui effectue l'authentification des joueurs sans capacité de serveur de jeu. Le modèle crée les ressources suivantes dans votre compte :

- Un groupe d'utilisateurs Amazon Cognito pour stocker les informations d'authentification des joueurs.
- Un AWS Lambda gestionnaire basé sur un point de terminaison Amazon API Gateway REST qui démarre des jeux et affiche les informations de connexion aux jeux.

Flotte d'une seule région

Ce scénario crée un service de backend de jeu avec une seule Amazon GameLift Servers flotte. Il crée les ressources suivantes :

- Un pool d'utilisateurs Amazon Cognito permettant à un joueur de s'authentifier et de démarrer une partie.
- Un AWS Lambda gestionnaire chargé de rechercher une session de jeu existante avec un emplacement de joueur libre dans la flotte. S'il ne trouve pas d'emplacement libre, il crée une nouvelle session de jeu.

Une flotte multirégionale avec une file d'attente et un système de matchmaking personnalisé

Ce scénario forme des matchs en utilisant des Amazon GameLift Servers files d'attente et un système de matchmaking personnalisé pour regrouper les joueurs les plus âgés du pool d'attente. Il crée les ressources suivantes :

- Rubrique Amazon Simple Notification Service qui Amazon GameLift Servers publie des messages sur. Pour plus d'informations sur les sujets et les notifications liés aux réseaux sociaux, consultez [Configurer la notification des événements pour le placement des sessions de jeu](#).
- Fonction Lambda invoquée par le message qui communique les détails du placement et de la connexion au jeu.
- Une table Amazon DynamoDB pour stocker les informations relatives à l'emplacement et à la connexion au jeu. GetGameConnectionles appels sont lus dans ce tableau et renvoient les informations de connexion au client du jeu.

Repérez les flottes grâce à une file d'attente et à un système de matchmaking personnalisé

Ce scénario forme des correspondances en utilisant des Amazon GameLift Servers files d'attente et un système de matchmaking personnalisé et configure trois flottes. Il crée les ressources suivantes :

- Deux flottes Spot contenant différents types d'instances afin de garantir la durabilité en cas d'indisponibilité de Spot.
- Une flotte à la demande qui fait office de sauvegarde pour les autres flottes Spot. Pour plus d'informations sur la conception de vos flottes, consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Une Amazon GameLift Servers file d'attente pour maintenir la disponibilité des serveurs à un niveau élevé et à un faible coût. Pour plus d'informations et les meilleures pratiques concernant les files d'attente, consultez [Personnaliser une file d'attente de session de jeu](#).

FlexMatch

Ce scénario utilise FlexMatch, un service de matchmaking géré, pour associer les joueurs du jeu. Pour plus d'informations FlexMatch, voir [Qu'est-ce que Amazon GameLift Servers FlexMatch](#). Ce scénario crée les ressources suivantes :

- Une fonction Lambda pour créer un ticket de matchmaking après avoir reçu StartGame des demandes.
- Une fonction Lambda séparée pour écouter les événements FlexMatch du match.

Pour éviter des frais inutiles Compte AWS, supprimez les ressources créées par chaque scénario une fois que vous avez fini de les utiliser. Supprimez la AWS CloudFormation pile correspondante.

Mettre à jour AWS les identifiants

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Le Amazon GameLift Servers plugin pour Unity nécessite des informations d'identification de sécurité pour déployer un scénario. Vous pouvez créer de nouvelles informations d'identification ou utiliser des informations d'identification existantes.

Pour plus d'informations sur la configuration des informations d'identification, voir [Comprendre et obtenir vos AWS informations d'identification](#).

Pour mettre à jour AWS les identifiants

1. Dans Unity, dans l'onglet Plug-in pour Unity, choisissez l'onglet Déployer.
2. Dans le volet Déployer, sélectionnez AWS Credentials.
3. Vous pouvez créer de nouvelles AWS informations d'identification ou choisir des informations d'identification existantes.
 - Pour créer des informations d'identification, choisissez Créer un nouveau profil d'informations d'identification, puis spécifiez le nouveau nom du profil, AWS l'ID de clé d'accès, la cléAWS secrète et Région AWS.
 - Pour choisir des informations d'identification existantes, choisissez Choisir un profil d'informations d'identification existant, puis sélectionnez un nom de profil et Région AWS.
4. Dans la fenêtre Mettre à jour les AWS informations d'identification, choisissez Mettre à jour le profil des informations d'identification.

Mettre à jour le bootstrap du compte

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

L'emplacement du bootstrap est un compartiment Amazon S3 utilisé lors du déploiement. Il est utilisé pour stocker les actifs du serveur de jeu et d'autres dépendances. Le compartiment Région AWS que vous choisissez doit être la même région que celle que vous utiliserez pour le déploiement du scénario.

Pour plus d'informations sur les compartiments Amazon S3, consultez [Création, configuration et utilisation des compartiments Amazon Simple Storage Service](#).

Pour mettre à jour l'emplacement du bootstrap du compte

1. Dans Unity, dans l'onglet Plug-in pour Unity, choisissez l'onglet Déployer.
2. Dans le volet Déploiement, choisissez Update Account Bootstrap.
3. Dans la fenêtre Account Bootstrapping, vous choisissez un compartiment Amazon S3 existant ou vous créez un nouveau compartiment Amazon S3 :
 - Pour choisir un compartiment existant, choisissez Choisir un compartiment Amazon S3 existant et Mettre à jour pour enregistrer votre sélection.
 - Choisissez Créer un nouveau compartiment Amazon S3 pour créer un nouveau compartiment Amazon Simple Storage Service, puis choisissez une politique. La politique précise la date d'expiration du compartiment Amazon S3. Choisissez Create pour créer le bucket.

Déployer un scénario de jeu

 Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Vous pouvez utiliser un scénario pour tester votre jeu Amazon GameLift Servers. Chaque scénario utilise un AWS CloudFormation modèle pour créer une pile avec les ressources requises. La plupart des scénarios nécessitent un exécutable sur le serveur de jeu et un chemin de compilation. Lorsque vous déployez le scénario, Amazon GameLift Servers copie les ressources du jeu sur l'emplacement de démarrage dans le cadre du déploiement.

Vous devez configurer des AWS informations d'identification et un bootstrap de AWS compte pour déployer un scénario.

Pour déployer un scénario

1. Dans Unity, dans l'onglet Plug-in pour Unity, choisissez l'onglet Déployer.
2. Dans le volet Déploiement, choisissez Open Deployment UI.
3. Dans la fenêtre Déploiement, choisissez un scénario.
4. Entrez un nom de jeu. Il doit être unique. Le nom du jeu fait partie du nom de la AWS CloudFormation pile lorsque vous déployez le scénario.
5. Choisissez le chemin du dossier de compilation du serveur de jeu. Le chemin du dossier de construction pointe vers le dossier contenant le fichier exécutable du serveur et ses dépendances.
6. Choisissez le chemin du fichier .exe de compilation du serveur de jeu. Le chemin du fichier exécutable de compilation pointe vers le fichier exécutable du serveur de jeu.
7. Choisissez Démarrer le déploiement pour commencer à déployer un scénario. Vous pouvez suivre l'état de la mise à jour dans la fenêtre Déploiement sous État actuel. Le déploiement des scénarios peut prendre plusieurs minutes.
8. Lorsque le scénario est terminé, l'état actuel est mis à jour pour inclure l'ID du client Cognito et le point de terminaison API Gateway que vous pouvez copier et coller dans le jeu.
9. Pour mettre à jour les paramètres du jeu, dans le menu Unity, choisissez Accéder aux paramètres de connexion du client. Cela affiche un onglet Inspector sur le côté droit de l'écran Unity.
10. Désélectionnez le mode de test local.
11. Entrez le point de terminaison API Gateway et l'ID du client Cognito. Choisissez le même Région AWS que celui que vous avez utilisé pour le déploiement du scénario. Vous pouvez ensuite reconstruire et exécuter le client du jeu à l'aide des ressources du scénario déployées.

Supprimer les ressources créées par le scénario

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Pour supprimer les ressources créées pour le scénario, supprimez la AWS CloudFormation pile correspondante.

Pour supprimer les ressources créées par le scénario

1. Dans la fenêtre de déploiement du Amazon GameLift Servers plugin pour Unity, sélectionnez Afficher AWS CloudFormation la console pour ouvrir la AWS CloudFormation console.
2. Dans la AWS CloudFormation console, choisissez Stacks, puis choisissez la pile qui inclut le nom du jeu spécifié lors du déploiement.
3. Choisissez Supprimer pour supprimer la pile. La suppression d'une pile peut prendre quelques minutes. Après avoir AWS CloudFormation supprimé la pile utilisée par le scénario, son statut passe à ROLLBACK_COMPLETE.

Intégrer des jeux Amazon GameLift Servers dans Unity

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Intégrez votre jeu Unity Amazon GameLift Servers en effectuant les tâches suivantes :

- [Intégration Amazon GameLift Servers à un projet de serveur de jeu Unity](#)
- [Intégration Amazon GameLift Servers à un projet client de jeu Unity](#)

Le schéma suivant montre un exemple de flux d'intégration d'un jeu. Dans le schéma, une flotte avec le serveur de jeu est déployée sur Amazon GameLift Servers. Le client de jeu communique avec le serveur de jeu, qui communique avec Amazon GameLift Servers.

Importez et lancez un exemple de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Le Amazon GameLift Servers plugin pour Unity inclut un exemple de jeu que vous pouvez utiliser pour découvrir les bases de l'intégration de votre jeu Amazon GameLift Servers. Dans cette section, vous allez créer le client de jeu et le serveur de jeu, puis tester localement à l'aide de Amazon GameLift Servers Local.

Prérequis

- [Configurez un Compte AWS](#)
- [Installation et configuration du plugin](#)

Créez et exécutez l'exemple de serveur de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Configurez les fichiers du serveur de jeu de l'exemple de jeu.

1. Dans Unity, dans le menu, choisissez Amazon GameLift Servers, puis choisissez Importer un exemple de jeu.
2. Dans la fenêtre Importer un exemple de jeu, choisissez Importer pour importer le jeu, ses actifs et ses dépendances.
3. Construisez le serveur de jeu. Dans Unity, dans le menu, choisissez Amazon GameLift Servers, puis choisissez Appliquer les paramètres de build de Windows Sample Server ou Appliquer les paramètres de build de macOS Sample Server. Après avoir configuré les paramètres du serveur de jeu, Unity recompile les ressources.
4. Dans Unity, dans le menu, choisissez Fichier, puis Build. Choisissez Server Build, choisissez Build, puis choisissez un dossier de build spécifiquement pour les fichiers du serveur.

Unity construit l'exemple de serveur de jeu en plaçant le fichier exécutable et les ressources requises dans le dossier de compilation spécifié.

Créez et exécutez l'exemple de client de jeu

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Configurez les fichiers client du jeu d'exemple.

1. Dans Unity, dans le menu, choisissez Amazon GameLift Servers, puis choisissez Appliquer les paramètres de génération du client d'exemple de Windows ou Appliquer les paramètres de génération du client d'exemple de macOS. Une fois les paramètres du client de jeu configurés, Unity recompile les actifs.
2. Dans Unity, dans le menu, sélectionnez Accéder aux paramètres du client. Cela affichera un onglet Inspector sur le côté droit de l'écran Unity. Dans l'onglet Paramètres du Amazon GameLift Servers client, sélectionnez Mode de test local.
3. Créez le client du jeu. Dans Unity, dans le menu, choisissez Fichier. Vérifiez que Server Build n'est pas cochée, choisissez Build, puis sélectionnez un dossier de build spécifiquement pour les fichiers client.

Unity crée l'exemple de client de jeu en plaçant l'exécutable et les ressources requises dans le dossier de construction du client spécifié.

4. Vous n'avez pas créé le serveur et le client de jeu. Dans les étapes suivantes, vous lancerez le jeu et vous verrez comment il interagit avec Amazon GameLift Servers.

Testez l'exemple de jeu localement

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Lancez l'exemple de jeu que vous avez importé à l'aide de Amazon GameLift Servers Local.

1. Lancez le serveur de jeu. Dans Unity, dans l'onglet Plug-in pour Unity, choisissez l'onglet Déployer.

2. Dans le volet Test, choisissez Open Local Test UI.
3. Dans la fenêtre de test local, spécifiez le chemin du fichier .exe du serveur de jeu. Le chemin doit inclure le nom de l'exécutable. Par exemple, C:/MyGame/GameServer/MyGameServer.exe.
4. Choisissez Deploy and Run. Le plugin pour Unity lance le serveur de jeu et ouvre une fenêtre de journal Amazon GameLift Servers local. La fenêtre contient des messages de journal, notamment des messages envoyés entre le serveur de jeu et Amazon GameLift Servers Local.
5. Lancez le client du jeu. Trouvez l'emplacement de compilation à l'aide de l'exemple de client de jeu et choisissez le fichier exécutable.
6. Dans le jeu Amazon GameLift Servers d'exemple, saisissez un e-mail et un mot de passe, puis choisissez Se connecter. L'e-mail et le mot de passe ne sont ni validés ni utilisés.
7. Dans le jeu Amazon GameLift Servers d'exemple, choisissez Start. Le client du jeu recherche une session de jeu. S'il ne trouve pas de session, il en crée une. Le client de jeu démarre ensuite la session de jeu. Vous pouvez voir l'activité du jeu dans les journaux.

Exemples de journaux de serveurs de jeux

```
...
2021-09-15T19:55:3495 PID:20728 Log :) GAMELIFT AWAKE
2021-09-15T19:55:3512 PID:20728 Log :) I AM SERVER
2021-09-15T19:55:3514 PID:20728 Log :) GAMELIFT StartServer at port 33430.
2021-09-15T19:55:3514 PID:20728 Log :) SDK VERSION: 4.0.2
2021-09-15T19:55:3556 PID:20728 Log :) SERVER IS IN A GAMELIFT FLEET
2021-09-15T19:55:3577 PID:20728 Log :) PROCESSREADY SUCCESS.
2021-09-15T19:55:3577 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
...
2021-09-15T19:55:3634 PID:20728 Log :) GAMELOGIC AWAKE
2021-09-15T19:55:3635 PID:20728 Log :) GAMELOGIC START
2021-09-15T19:55:3636 PID:20728 Log :) LISTENING ON PORT 33430
2021-09-15T19:55:3636 PID:20728 Log SERVER: Frame: 0 HELLO WORLD!
...
2021-09-15T19:56:2464 PID:20728 Log :) GAMELIFT SESSION REQUESTED
2021-09-15T19:56:2468 PID:20728 Log :) GAME SESSION ACTIVATED
2021-09-15T19:56:3578 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:57:3584 PID:20728 Log :) GAMELIFT HEALTH CHECK REQUESTED (HEALTHY)
2021-09-15T19:58:0334 PID:20728 Log SERVER: Frame: 8695 Connection accepted: playerId
0 joined
2021-09-15T19:58:0335 PID:20728 Log SERVER: Frame: 8696 Connection accepted: playerId
1 joined
```

```

2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 0 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0338 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 0:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 0
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from playerId 1 Msg:
CONNECT: server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 Msg rcvd from player 1:CONNECT:
server IP localhost
2021-09-15T19:58:0339 PID:20728 Log SERVER: Frame: 8697 CONNECT: player index 1

```

Exemples de journaux Amazon GameLift Servers locaux

```

12:55:26,000 INFO || - [SocketIOServer] main - Session store / pubsub factory used:
MemoryStoreFactory (local session store only)
12:55:28,092 WARN || - [ServerBootstrap] main - Unknown channel option 'SO_LINGER' for
channel '[id: 0xe23d0a14]'
12:55:28,101 INFO || - [SocketIOServer] nioEventLoopGroup-2-1 - SocketIO server
started at port: 5757
12:55:28,101 INFO || - [SDKConnection] main - GameLift SDK server (communicates with
your game server) has started on http://localhost:5757
12:55:28,120 INFO || - [SdkWebSocketServer] WebSocketSelector-20 - WebSocket Server
started on address localhost/127.0.0.1:5759
12:55:28,166 INFO || - [StandAloneServer] main - GameLift Client server (listens for
GameLift client APIs) has started on http://localhost:8080
12:55:28,179 INFO || - [StandAloneServer] main - GameLift server sdk http listener has
started on http://localhost:5758
12:55:35,453 INFO || - [SdkWebSocketServer] WebSocketWorker-12 - onOpen
socket: /?pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp and handshake /?
pID=20728&sdkVersion=4.0.2&sdkLanguage=CSharp
12:55:35,551 INFO || - [HostProcessManager] WebSocketWorker-12 - client connected with
pID 20728
12:55:35,718 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: ProcessReady for pId 20728
12:55:35,718 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for processReady from 20728
12:55:35,738 INFO || - [ProcessReadyHandler] GameLiftSdkHttpHandler-thread-0 -
onProcessReady data: port: 33430
12:55:35,739 INFO || - [HostProcessManager] GameLiftSdkHttpHandler-thread-0 -
Registered new process with pId 20728
12:55:35,789 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: ReportHealth for pId 20728

```



```

12:55:35,790 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for ReportHealth from 20728
12:55:35,794 INFO || - [ReportHealthHandler] GameLiftSdkHttpHandler-thread-0 -
ReportHealth data: healthStatus: true
12:56:24,098 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.DescribeGameSessions
12:56:24,119 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,241 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.CreateGameSession
12:56:24,242 INFO || - [CreateGameSessionDispatcher] Thread-12 - Received API call to
create game session with input: {"FleetId":"fleet-123","MaximumPlayerSessionCount":4}
12:56:24,265 INFO || - [HostProcessManager] Thread-12 - Reserved process:
20728 for gameSession: arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d
12:56:24,266 INFO || - [WebSocketInvoker] Thread-12 - StartGameSessionRequest:
gameSessionId=arn:aws:gamelift:local::gamesession/fleet-123/
gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d, fleetId=fleet-123, gameSessionName=null,
maxPlayers=4, properties=[], ipAddress=127.0.0.1, port=33430, gameSessionData?=false,
matchmakerData?=false, dnsName=localhost
12:56:24,564 INFO || - [CreateGameSessionDispatcher] Thread-12 - GameSession with
id: arn:aws:gamelift:local::gamesession/fleet-123/gsess-59f6cc44-4361-42f5-95b5-
fdb5825c0f3d created
12:56:24,585 INFO || - [GameLiftHttpHandler] Thread-12 - API to use:
GameLift.DescribeGameSessions
12:56:24,585 INFO || - [DescribeGameSessionsDispatcher] Thread-12 - Received API call
to describe game sessions with input: {"FleetId":"fleet-123"}
12:56:24,660 INFO || - [GameLiftSdkHttpHandler] GameLiftSdkHttpHandler-thread-0 -
GameLift API to use: GameSessionActivate for pId 20728
12:56:24,661 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0 -
Received API call for GameSessionActivate from 20728
12:56:24,678 INFO || - [GameSessionActivateHandler] GameLiftSdkHttpHandler-thread-0
- GameSessionActivate data: gameSessionId: "arn:aws:gamelift:local::gamesession/
fleet-123/gsess-59f6cc44-4361-42f5-95b5-fdb5825c0f3d"

```

Arrêter le processus du serveur

Note

Cette rubrique fait référence au Amazon GameLift Servers plugin pour Unity version 1.0.0, qui utilise le SDK du serveur 4.x ou une version antérieure.

Une fois que vous avez terminé votre exemple de jeu, arrêtez le serveur dans Unity.

1. Dans le client de jeu, choisissez Quitter ou fermez la fenêtre pour arrêter le client de jeu.
2. Dans Unity, dans la fenêtre de test local, choisissez Arrêter ou fermez les fenêtres du serveur de jeu pour arrêter le serveur.

Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Managed EC2

Cette feuille de route vous explique comment développer une solution d' EC2 hébergement Amazon GameLift Servers géré pour votre jeu multijoueur. Amazon GameLift Servers propose plusieurs options d'hébergement de jeux ; pour plus d'informations sur ces options, consultez [Amazon GameLift Servers solutions](#).

Avec l'hébergement Amazon GameLift Servers géré, votre serveur de jeu est hébergé sur des ressources AWS Cloud informatiques virtuelles qui Amazon GameLift Servers possèdent et fonctionnent en fonction de votre configuration. Vous bénéficiez de la sécurité, de la fiabilité et de la disponibilité mondiale des instances Amazon Elastic Compute Cloud (Amazon EC2) qui sont encore optimisées pour une utilisation avec l'hébergement de jeux multijoueurs. Amazon GameLift Servers rationalise la gestion de l'hébergement grâce à des outils tels que les déploiements automatiques de serveurs, la gestion du cycle de vie et l'auto-scaling des capacités.

Une solution Amazon GameLift Servers gérée est composée des composants suivants :

- Une ou plusieurs flottes Amazon GameLift Servers gérées, qui utilisent des instances Amazon Elastic Compute Cloud (Amazon EC2) optimisées pour l'hébergement de jeux multijoueurs.
- Une version de serveur de jeu, intégrée au SDK du serveur pour Amazon GameLift Servers, à déployer sur toutes les flottes.
- Un client de jeu et un service principal, intégrés au AWS SDK, pour interagir avec le Amazon GameLift Servers service et demander des sessions de jeu.
- Une Amazon GameLift Servers file d'attente pour placer de nouvelles sessions de jeu avec les serveurs de jeu disponibles dans toutes les flottes.
- (Facultatif) Un FlexMatch système de matchmaking pour créer des matchs multijoueurs et configurer des sessions de jeu pour eux.

Cette feuille de route présente une approche simplifiée pour que votre jeu multijoueur soit opérationnel avec succès grâce Amazon GameLift Servers à l'EC2 hébergement géré. Une fois les composants nécessaires en place, vous pouvez continuer à développer le jeu et à personnaliser votre solution d'hébergement. À l'approche du lancement, consultez-les [Préparation de votre jeu pour le lancement grâce à l'Amazon GameLift Servers hébergement](#) pour vous aider à préparer votre solution d'hébergement pour une utilisation en production.

i Démarrez rapidement avec le Amazon GameLift Servers plugin pour Unreal Engine et Unity

Pour un déploiement plus rapide, essayez le [Amazon GameLift Servers plugin](#) pour Unreal Engine et Unity. Il fournit des flux de travail d'interface utilisateur guidés pour déployer rapidement votre serveur de jeu avec une configuration minimale, afin que vous puissiez tester les composants de votre jeu en action. Vous pouvez ensuite vous appuyer sur cette base pour créer une solution d'hébergement personnalisée pour votre jeu. Pour en savoir plus, consultez [Explorez avec le Amazon GameLift Servers](#) ..

Étape 1 : Préparez votre serveur de jeu pour qu'il fonctionne Amazon GameLift Servers

Ajoutez des fonctionnalités à votre serveur de jeu afin qu'il puisse communiquer avec le Amazon GameLift Servers service lorsqu'il est déployé pour l'hébergement.

- Procurez-vous le SDK du serveur pour Amazon GameLift Servers (version 5.x) pour votre projet de jeu. Le SDK du serveur est disponible en C++, C# et Go. [Téléchargez un SDK Amazon GameLift Servers de serveur](#).
- Modifiez le code de votre serveur de jeu pour ajouter les fonctionnalités du SDK du serveur. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
 - Ajoutez du code pour initialiser le Amazon GameLift Servers SDK et établir une WebSocket connexion avec le Amazon GameLift Servers service. Utilisez l'action `InitSdk()` du SDK du serveur.
 - Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur est prêt à héberger des sessions de jeu. Utilisez l'action `ProcessReady()` du SDK du serveur.
 - Implémentez les fonctions de rappel `requisitesOnProcessTerminate()`, et `OnStartGameSession()` Grâce à ces fonctions, les processus du serveur de jeu peuvent

maintenir une connexion avec le Amazon GameLift Servers service, lancer une session de jeu lorsque cela Amazon GameLift Servers est demandé et répondre à une demande d'arrêt du processus du serveur de jeu.

- Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur met fin à une session de jeu. Utilisez l'action `ProcessEnding()` du SDK du serveur.
- Package de la version de votre serveur de jeu. Créez un script d'installation avec vos fichiers de compilation, vos dépendances et les logiciels de support. Consultez [Conditionner les fichiers de version de jeu](#). Nous vous recommandons d'utiliser un bucket Amazon Simple Storage Service (Amazon S3) pour stocker les versions de votre build de jeu.
- Testez l'intégration de votre serveur de jeu. Pour cette tâche, nous vous recommandons de configurer un parc Amazon GameLift Servers Anywhere pour un poste de travail local, comme décrit dans [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#). Pour cette étape, installez manuellement le build de votre serveur de jeu sur le périphérique de test et lancez un processus serveur. Utilisez la AWS CLI pour demander une nouvelle session de jeu et vérifiez que le Amazon GameLift Servers service invite correctement votre processus serveur à démarrer une session de jeu.

Étape 2 : Préparez votre client de jeu pour qu'il rejoigne les sessions de jeu hébergées

Créez un moyen pour votre client de jeu de demander à rejoindre une session de jeu, d'obtenir des informations de connexion, puis de se connecter directement à une session de jeu hébergée. L'approche la plus courante consiste à configurer une fonctionnalité de service principal qui sert d'intermédiaire entre le client de jeu et le service. Amazon GameLift Servers Cette approche protège vos ressources d'hébergement et vous permet de mieux contrôler la manière dont les joueurs sont placés dans les sessions de jeu.

- Développez des fonctionnalités de service principal pour l'hébergement. Le service principal communique avec le Amazon GameLift Servers service et fournit des informations de connexion à un client de jeu. Cette fonctionnalité inclut le démarrage de sessions de jeu, le placement de joueurs dans des jeux et la récupération des informations de session de jeu. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
- Procurez-vous le AWS SDK pour Amazon GameLift Servers et ajoutez-le à votre projet de service principal. Consultez les [ressources du Amazon GameLift Servers SDK pour les services clients](#).

- Ajoutez du code pour initialiser un Amazon GameLift Servers client et enregistrer les paramètres clés. Consultez [Configuration Amazon GameLift Servers sur un service principal](#).
- Ajoutez une fonctionnalité permettant d'appeler l'action du AWS SDK `CreateGameSession()` et de fournir des informations de connexion à une session de jeu à un client de jeu. Voir [Créer une session de jeu sur une flotte spécifique](#).

`CreateGameSession()` L'appel est un point de départ pratique pour demander de nouvelles sessions de jeu. Une fois que vous aurez mis en place un système de placement de sessions de jeu (voir étape 3), vous remplacerez ce code par un appel à `StartGameSessionPlacement()` (ou `StartMatchmaking()` si vous utilisez `FlexMatch`).

Pour obtenir des conseils sur la conception de votre service principal, consultez [Concevez le service client de votre jeu](#).

- Ajoutez des fonctionnalités à votre client de jeu qui permettent aux joueurs de rejoindre une session de jeu hébergée. Le client du jeu adresse des requêtes à votre service principal, et non directement à Amazon GameLift Servers. Une fois que le service principal a fourni les informations de connexion à la session de jeu, le client du jeu se connecte directement à la session de jeu pour jouer au jeu.
- Testez l'intégration de votre client de jeu. Vous pouvez utiliser le même parc Amazon GameLift Servers Anywhere avec un poste de travail local à des fins de test.

Pour un développement itératif rapide ou lorsque vous travaillez avec des équipes composées de plusieurs personnes, nous vous recommandons de configurer un environnement de [test basé sur le cloud](#). Cette solution Amazon GameLift Servers Toolkit imite le comportement d'une flotte Amazon GameLift Servers gérée mais vous permet de mettre à jour les versions des serveurs de jeu avec un délai d'exécution minimal.

Étape 3 : configurer le placement des sessions de jeu

Personnalisez la façon dont vous Amazon GameLift Servers souhaitez traiter les demandes de nouvelle session de jeu et localisez les serveurs de jeu disponibles pour les héberger. Amazon GameLift Servers suit automatiquement la disponibilité de tous les serveurs de jeu sur toutes les flottes. Lorsqu'un client de jeu envoie une demande pour rejoindre une session de jeu, Amazon GameLift Servers recherche le « meilleur emplacement possible » en fonction d'un ensemble de priorités définies telles que le minimum de latence, de coût et de disponibilité.

- Créez une file d'attente de session de jeu pour placer une nouvelle session de jeu parmi les serveurs de jeu disponibles. Les files d'attente sont le principal mécanisme de placement des sessions de jeu. Pour de plus amples informations, consultez [Création d'une file d'attente de session de jeu](#).
- Ajoutez au minimum vos flottes Anywhere en tant que destinations dans votre file d'attente. Tous les autres paramètres sont des personnalisations facultatives.
- Dans votre code de service principal, convertissez l'**CreateGameSession()** appel en **StartGameSessionPlacement()**. Consultez la section [Création d'une session de jeu dans une file d'attente multisite](#).
- Créez un mécanisme pour avertir un client de jeu lorsqu'une session de jeu est prête à être ouverte. Pendant le développement, vous pouvez demander l'état d'une session de jeu en appelant à `DescribeGameSessionPlacement`. Avant d'utiliser une file d'attente pour traiter des volumes importants, vous devez toutefois activer les notifications d'événements. Consultez [Configurer la notification des événements pour le placement des sessions de jeu](#).
- (Facultatif) Ajoutez des composants FlexMatch de matchmaking. Pour obtenir des conseils, consultez le [guide du Amazon GameLift Servers FlexMatch développeur](#).

Étape 4 : créer des flottes gérées dans le cloud

Jusqu'à présent, vous avez travaillé avec des flottes Anywhere autogérées pour tester et itérer les composants de votre jeu, et vous avez affiné le placement de vos sessions de jeu. Le dernier élément de votre solution consiste à configurer le type de ressources d'hébergement dont vous aurez besoin pour un système de production. Pour commencer à planifier et à configurer pour la production, vous souhaitez passer à l'utilisation d'un parc Amazon GameLift Servers géré.

- Package de la compilation et du téléchargement de votre serveur de jeu sur Amazon GameLift Servers. Créez un script d'installation avec vos fichiers de compilation, vos dépendances et les logiciels de support. Consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#). Vous pouvez télécharger votre build à Amazon GameLift Servers l'aide de la console ou de la AWS CLI.

Avant de télécharger votre build, déterminez le type de flotte que Région AWS vous souhaitez créer. Vous devez télécharger le build dans la même région. Pour en savoir plus sur le choix de l'emplacement d'une flotte, voir [Emplacement de la flotte](#).

- Créez une EC2 flotte gérée. Lorsque vous créez une flotte, commencez Amazon GameLift Servers immédiatement à déployer la version de votre serveur de jeu pour l'hébergement. Vous pouvez

configurer de nombreux aspects d'un parc géré. Pour de plus amples informations, consultez [Création d'une EC2 flotte Amazon GameLift Servers gérée](#). Procédez au minimum comme suit :

- Donnez un nom à la flotte et spécifiez la version de jeu téléchargée à déployer.
- Choisissez des instances à la demande pour votre flotte et sélectionnez un type d'instance disponible dans l'emplacement de votre flotte. Les flottes Spot constituent une option intéressante, mais nécessitent une conception et une configuration supplémentaires.
- Créez une configuration d'exécution pour le parc. Spécifiez au minimum le chemin de lancement de l'exécutable de votre serveur de jeu.
- Spécifiez les paramètres de port pour autoriser le trafic entrant à accéder à vos serveurs de jeu.
- Ajoutez les flottes gérées à votre file d'attente. Dans la file d'attente de votre session de jeu, remplacez les flottes Anywhere par les que vous gérez.
- Testez l'hébergement de jeux avec vos flottes gérées. À ce stade, vous devriez être en mesure de tester l'ensemble du cycle d'hébergement, avec un client de jeu demandant une session de jeu, obtenant des informations de connexion et se connectant avec succès à une session de jeu.

Étape 5 : Personnalisez vos flottes gérées

Alors que vous vous préparez au lancement du jeu, vous devrez affiner vos ressources d'hébergement géré. Parmi les décisions à prendre en compte, mentionnons les suivantes :

- Envisagez d'ajouter des flottes Spot pour réaliser des économies. Consultez [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#).
- Si votre serveur de jeu doit communiquer avec d'autres AWS ressources, configurez des rôles IAM pour gérer l'accès. Consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
- Déterminez l'emplacement géographique où vous souhaitez positionner les serveurs de jeu. Ajoutez des sites distants à vos flottes gérées. Consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Optimisez les performances du parc en sélectionnant un type et une taille d'instance et en configurant le moteur d'exécution pour exécuter plusieurs processus de serveur. Consultez [Gérez comment Amazon GameLift Servers lance des serveurs de jeu](#).
- Testez les options de placement des sessions de jeu pour les flottes gérées, notamment en personnalisant les paramètres de priorisation. Consultez [Personnaliser une file d'attente de session de jeu](#).

- Configurez le dimensionnement automatique des capacités pour répondre à la demande attendue des joueurs. Consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).
- Configurez des flottes de réserve dans d'autres, modifiez les files d'attente Régions AWS et le dimensionnement automatique pour gérer les basculements si nécessaire.
- Configurez les outils d'observabilité de l'hébergement, notamment les outils d'analyse et de journalisation. Consultez [Surveillance des Amazon GameLift Servers](#).
- Automatisez votre déploiement à l'aide de l'[infrastructure en tant que code \(IaC\)](#). Consultez [Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation](#).

Amazon GameLift Servers prend en charge l'utilisation de AWS CloudFormation modèles pour toutes les configurations spécifiques au déploiement. Vous pouvez également utiliser le AWS Cloud Development Kit (AWS CDK) pour définir vos Amazon GameLift Servers ressources. Pour plus d'informations à ce sujet AWS CDK, consultez le [guide du AWS Cloud Development Kit \(AWS CDK\) développeur](#).

Pour gérer le déploiement de vos AWS CloudFormation stacks, nous vous recommandons d'utiliser des outils et services d'intégration continue et de livraison continue (CI/CD) tels que. AWS CodePipeline Ces outils vous permettent de déployer automatiquement ou avec approbation chaque fois que vous créez le binaire d'un serveur de jeu. Avec un CI/CD outil ou un service, le déploiement des ressources pour une nouvelle version de serveur de jeu peut ressembler à ceci :

- Créez et testez le binaire de votre serveur de jeu.
- Téléchargez le fichier binaire sur Amazon GameLift Servers.
- Déployez de nouvelles flottes avec la nouvelle version.
- Ajoutez les nouvelles flottes à votre file d'attente de session de jeu et supprimez les flottes de la version précédente.
- Lorsque les flottes de la version précédente n'hébergent plus de sessions de jeu actives, supprimez les AWS CloudFormation piles de ces flottes.

Feuille de route de développement pour l'hébergement avec des conteneurs Amazon GameLift Servers gérés

Cette feuille de route vous explique comment développer une solution d'hébergement Amazon GameLift Servers géré pour vos serveurs de jeux conteneurisés. Les conteneurs gérés ne

sont qu'une des solutions d'hébergement proposées par Amazon GameLift Servers. Pour plus d'informations sur les options d'hébergement, consultez [Amazon GameLift Servers solutions](#).

Une solution de conteneur géré Amazon GameLift Servers comportant les composants suivants :

- Une ou plusieurs flottes de conteneurs utilisant des instances Amazon Elastic Compute Cloud (Amazon EC2) optimisées pour l'hébergement de jeux multijoueurs.
- Une image de conteneur contenant la version de votre serveur de jeu, téléchargée dans le référentiel privé Amazon Elastic Container Registry (Amazon ECR). La version du serveur de jeu est intégrée au SDK du serveur Amazon GameLift Servers et conçue pour fonctionner sous Linux.
- Un service principal qui interagit avec le Amazon GameLift Servers service pour le compte de vos clients de jeu. Le service principal utilise les fonctionnalités de l'API de service pour Amazon GameLift Servers, qui fait partie du AWS SDK.
- Une file d'attente de session de Amazon GameLift Servers jeu qui traite les demandes de nouvelles sessions de jeu, recherche les serveurs de jeu disponibles dans toutes les flottes et invite un serveur de jeu à démarrer une session de jeu.
- (Facultatif) Un FlexMatch système de matchmaking pour créer des matchs multijoueurs et configurer des sessions de jeu pour eux.

Cette feuille de route présente une approche rationalisée pour que vos serveurs de jeux conteneurisés soient opérationnels avec succès grâce à des conteneurs Amazon GameLift Servers gérés. Une fois les composants nécessaires en place, vous pouvez continuer à développer le jeu et à personnaliser votre solution d'hébergement. À l'approche du lancement, consultez les [Préparation de votre jeu pour le lancement grâce à l'Amazon GameLift Servers hébergement](#) pour vous aider à préparer votre solution d'hébergement pour une utilisation en production.

 Accélérez l'intégration grâce à ces outils pour les conteneurs gérés :

- Le [kit de démarrage pour conteneurs](#) rationalise l'intégration et la configuration de la flotte. Il ajoute des fonctionnalités essentielles de gestion des sessions de jeu à votre serveur de jeu et utilise des modèles préconfigurés pour créer une flotte de conteneurs et un pipeline de déploiement automatisé pour votre serveur de jeu. Après le déploiement, utilisez la Amazon GameLift Servers console et les outils d'API pour surveiller les performances de la flotte, gérer les sessions de jeu et analyser les indicateurs.
- Pour les développeurs d'Unreal Engine et Unity, utilisez les [Amazon GameLift Servers plugins](#) pour intégrer votre serveur de jeu et créer une flotte de conteneurs à partir

de l'environnement de développement de votre moteur de jeu. Les flux de travail guidés du plugin vous aident à créer une solution simple et rapide avec un hébergement basé sur le cloud à l'aide de conteneurs gérés. Vous pouvez vous appuyer sur cette base pour créer une solution d'hébergement personnalisée pour votre jeu.

Étape 1 : Préparez votre serveur de jeu pour qu'il fonctionne Amazon GameLift Servers

Ajoutez des fonctionnalités à votre serveur de jeu afin qu'il puisse communiquer avec le Amazon GameLift Servers service lorsqu'il est déployé pour l'hébergement.

- Procurez-vous le SDK du serveur pour Amazon GameLift Servers (version 5.2 ou supérieure) pour votre projet de jeu. Le SDK du serveur est disponible en C++, C# et Go. [Téléchargez le SDK du serveur pour Amazon GameLift Servers](#). Le SDK du serveur est disponible en C++, C# et Go.
- Modifiez le code de votre serveur de jeu pour ajouter les fonctionnalités du SDK du serveur. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
 - Ajoutez du code pour initialiser le Amazon GameLift Servers SDK et établir une WebSocket connexion avec le Amazon GameLift Servers service. Utilisez l'action `InitSdk()` du SDK du serveur.
 - Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur est prêt à héberger des sessions de jeu. Utilisez l'action `ProcessReady()` du SDK du serveur.
 - Implémentez les fonctions de rappel `requisitesOnStartGameSession()`, et `OnProcessTerminate()`. Grâce à ces fonctions, les processus du serveur de jeu peuvent maintenir une connexion avec le Amazon GameLift Servers service, lancer une session de jeu lorsque cela Amazon GameLift Servers est demandé et répondre à une demande de fin du processus du serveur de jeu.
 - Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur met fin à une session de jeu. Utilisez l'action `ProcessEnding()` du SDK du serveur.
- Package de la version de votre serveur de jeu. Concevez votre serveur de jeu pour qu'il fonctionne sous Linux. Préparez le build et les autres fichiers nécessaires au fonctionnement du serveur de jeu. Si vous développez sous Windows, cette étape peut impliquer de configurer un espace de travail Linux distinct ou d'utiliser un outil tel que le sous-système Windows pour Linux (WSL). Vous

aurez besoin d'un environnement Linux pour tester la version de votre serveur de jeu, ainsi que pour créer et tester vos images de conteneur.

- Testez l'intégration de votre serveur de jeu. Vérifiez que votre serveur de jeu intégré peut se connecter au Amazon GameLift Servers service et répondre aux demandes. Nous vous recommandons de configurer une flotte Amazon GameLift Servers Anywhere simple avec un poste de travail local comme hôte de test, comme décrit dans [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#). Installez le build de votre serveur de jeu sur l'hôte de test et lancez un processus de serveur. Utilisez la AWS CLI pour demander une nouvelle session de jeu et vérifiez que le Amazon GameLift Servers service invite correctement votre processus serveur à démarrer une session de jeu.

Étape 2 : Préparez votre client de jeu pour qu'il rejoigne les sessions de jeu hébergées

Créez un moyen pour votre client de jeu de demander à rejoindre une session de jeu, d'obtenir des informations de connexion, puis de se connecter directement à une session de jeu hébergée. L'approche la plus courante consiste à configurer une fonctionnalité de service principal qui sert d'intermédiaire entre le client de jeu et le service. Amazon GameLift Servers Cette approche protège vos ressources d'hébergement et vous permet de mieux contrôler la manière dont les joueurs sont placés dans les sessions de jeu.

- Développez des fonctionnalités de service principal pour l'hébergement. Le service principal communique avec le Amazon GameLift Servers service et fournit des informations de connexion à un client de jeu. Cette fonctionnalité inclut le démarrage de sessions de jeu, le placement de joueurs dans des jeux et la récupération des informations de session de jeu. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
- Procurez-vous le AWS SDK pour Amazon GameLift Servers et ajoutez-le à votre projet de service principal. Consultez les [ressources du Amazon GameLift Servers SDK pour les services clients](#).
- Ajoutez du code pour initialiser un Amazon GameLift Servers client et enregistrer les paramètres clés. Consultez [Configuration Amazon GameLift Servers sur un service principal](#).
- Ajoutez une fonctionnalité permettant d'appeler l'action du AWS SDK `CreateGameSession()` et de fournir des informations de connexion à une session de jeu à un client de jeu. Voir [Créer une session de jeu sur une flotte spécifique](#).

`CreateGameSession()` L'appel est un point de départ pratique pour demander de nouvelles sessions de jeu. Une fois que vous aurez mis en place un système de placement de sessions de jeu (voir étape 3), vous remplacerez ce code par un appel à `StartGameSessionPlacement()` (ou `StartMatchmaking()` si vous utilisez `FlexMatch`).

Pour obtenir des conseils sur la conception de votre service principal, consultez [Concevez le service client de votre jeu](#).

- Ajoutez des fonctionnalités à votre client de jeu qui permettent aux joueurs de rejoindre une session de jeu hébergée. Le client du jeu adresse des requêtes à votre service principal, et non directement à Amazon GameLift Servers. Une fois que le service principal a fourni les informations de connexion à la session de jeu, le client du jeu se connecte directement à la session de jeu pour jouer au jeu.
- Testez l'intégration de votre client de jeu. Vous pouvez utiliser votre flotte Amazon GameLift Servers Anywhere existante avec un poste de travail local à des fins de test. Utilisez le nouveau service principal pour demander une nouvelle session de jeu et vérifiez que : (1) le Amazon GameLift Servers service invite correctement le processus de votre serveur à démarrer une session de jeu, et (2) un client de jeu peut se connecter à la session de jeu.

Étape 3 : configurer le placement des sessions de jeu

Personnalisez la façon dont vous Amazon GameLift Servers souhaitez traiter les demandes de nouvelle session de jeu et localisez les serveurs de jeu disponibles pour les héberger. Amazon GameLift Servers suit automatiquement la disponibilité de tous les serveurs de jeu sur toutes les flottes. Lorsqu'un client de jeu envoie une demande pour rejoindre une session de jeu, Amazon GameLift Servers il recherche le « meilleur emplacement possible » en fonction d'un ensemble de priorités définies telles que le minimum de latence, de coût et de disponibilité.

- Créez une file d'attente de session de jeu pour placer une nouvelle session de jeu parmi les serveurs de jeu disponibles. Les files d'attente sont le principal mécanisme de placement des sessions de jeu. Pour de plus amples informations, consultez [Création d'une file d'attente de session de jeu](#).
 - Ajoutez au minimum vos flottes Anywhere en tant que destinations dans votre file d'attente. Tous les autres paramètres sont des personnalisations facultatives.
- Dans votre code de service principal, convertissez l'`CreateGameSession()` appel en `StartGameSessionPlacement()`. Consultez la section [Création d'une session de jeu dans une file d'attente multisite](#).

- Créez un mécanisme pour avertir un client de jeu lorsqu'une session de jeu est prête à être ouverte. Pendant le développement, vous pouvez demander l'état d'une session de jeu en appelant `DescribeGameSessionPlacement`. Avant d'utiliser une file d'attente pour traiter des volumes importants, vous devez toutefois activer les notifications d'événements. Consultez [Configurer la notification des événements pour le placement des sessions de jeu](#).
- Ajouter FlexMatch le matchmaking (facultatif). Créez un ensemble de règles de matchmaking et créez une configuration de matchmaking compatible avec votre file d'attente de session de jeu. Pour obtenir des conseils sur la configuration d'un système de matchmaking, consultez le [guide du Amazon GameLift Servers FlexMatch développeur](#).
- Testez le système de placement. Vous pouvez utiliser votre flotte Amazon GameLift Servers Anywhere existante avec un poste de travail local à des fins de test. Utilisez le service principal pour demander une nouvelle session de jeu et vérifiez que le Amazon GameLift Servers service invite correctement le processus de votre serveur à démarrer une session de jeu.

Étape 4 : Création d'une image de conteneur de serveur de jeu

Après avoir intégré votre serveur de jeu avec succès, créez une image de conteneur avec le fichier exécutable de votre serveur de jeu. Stockez-le dans un référentiel Amazon Elastic Container Registry (Amazon ECR) pour l'utiliser avec Amazon GameLift Servers. Pour obtenir des instructions complètes, veuillez consulter [Créez une image de conteneur pour Amazon GameLift Servers](#).

- Obtenez le modèle Dockerfile pour un conteneur de serveur de jeu (fourni par Amazon GameLift Servers). Modifiez le fichier pour les fichiers de build de votre serveur de jeu.
- Créez une image de conteneur de serveur de jeu. Dans un environnement Linux, utilisez l'outil Docker pour créer votre image.
- Transférez l'image de votre conteneur vers Amazon ECR. Créez un référentiel public ou privé dans Amazon ECR, en l'utilisant Compte AWS et à l' Région AWS endroit où vous prévoyez de déployer votre flotte de conteneurs. Insérez l'image de votre conteneur dessus.
- Testez les images de vos conteneurs à l'aide de votre flotte Anywhere (facultatif). Vous souhaitez peut-être tester vos images de conteneurs localement avant de les déployer sur une flotte de conteneurs hébergée dans le cloud. Vous pouvez utiliser votre flotte Amazon GameLift Servers Anywhere existante avec un poste de travail local à des fins de test. Installez et exécutez le conteneur du serveur de jeu et vérifiez que : (1) le Amazon GameLift Servers service invite correctement le processus de votre serveur à démarrer une session de jeu, et (2) un client de jeu peut se connecter à la session de jeu.

Étape 5 : créer une flotte de conteneurs basée sur le cloud

Jusqu'à présent, vous avez travaillé avec une flotte Anywhere autogérée pour tester et modifier les composants de votre jeu. Le dernier élément de votre solution consiste à configurer les ressources d'hébergement basées sur le cloud dont vous aurez besoin pour un système de production. Pour commencer à planifier et à configurer la production, vous souhaitez configurer un parc de conteneurs Amazon GameLift Servers géré et le personnaliser pour la production.

- Créez des définitions de groupes de conteneurs. Les définitions des groupes de conteneurs décrivent l'architecture des conteneurs d'une flotte et identifient les images de conteneurs à déployer. Consultez [Création d'une définition de groupe de conteneurs pour une flotte de Amazon GameLift Servers conteneurs](#). Créez votre définition de groupe de conteneurs dans le même Région AWS endroit où les images de conteneurs sont stockées. Pour en savoir plus sur le choix de l'emplacement d'une flotte, voir [Emplacement de la flotte](#). Procédez au minimum comme suit :
 - Créez une définition de groupe de conteneurs de serveurs de jeu.
 - Ajoutez une définition de conteneur avec une image de conteneur avec la version de votre serveur de jeu.
 - Configurez une plage de ports pour les processus du serveur de jeu du conteneur.
- Créez une flotte de conteneurs gérée. Lorsque vous créez une flotte, commencez Amazon GameLift Servers immédiatement à déployer la version de votre serveur de jeu pour l'hébergement. Vous pouvez configurer de nombreux aspects d'un parc géré. Pour de plus amples informations, consultez [Création d'une flotte de conteneurs Amazon GameLift Servers gérée](#). Procédez au minimum comme suit :
 - Configurez un rôle de service AWS Identity and Access Management (IAM) pour le parc de conteneurs. Consultez [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).
 - Spécifiez la définition du groupe de conteneurs du serveur de jeu à déployer sur les instances de flotte.
 - Utilisez les valeurs par défaut lorsqu'elles sont disponibles pour tous les autres paramètres. Amazon GameLift Servers calcule certains paramètres pour une configuration optimale.
- Ajoutez les flottes de conteneurs à votre file d'attente. Dans la file d'attente de votre session de jeu, remplacez le parc de test Anywhere par votre parc de conteneurs géré.
- Testez l'hébergement de jeux avec vos flottes de conteneurs. À ce stade, vous devriez être en mesure de tester l'ensemble de la solution. Démarrez un client de jeu et demandez une session de jeu via le service principal. Obtenez des informations de connexion et connectez-vous à une session de jeu sur la flotte de conteneurs.

- Procédez à une itération des déploiements de votre flotte. Vous pouvez mettre à jour les définitions de groupes de conteneurs et les configurations de flotte, puis déployer des mises à jour sur les flottes existantes.

Étape 6 : Personnalisez votre solution de gestion de conteneurs

Alors que vous vous préparez au lancement du jeu, vous devrez affiner vos ressources d'hébergement géré. Parmi les décisions à prendre en compte, mentionnons les suivantes :

- Optimisez la configuration de votre flotte de conteneurs. Consultez [Personnalisez une flotte de Amazon GameLift Servers conteneurs](#).
- Envisagez d'ajouter des flottes Spot pour réaliser des économies. Consultez [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#).
- Si votre serveur de jeu doit communiquer avec d'autres AWS ressources, configurez des rôles IAM pour gérer l'accès. Consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
- Déterminez l'emplacement géographique où vous souhaitez positionner les serveurs de jeu. Ajoutez des sites distants à vos flottes gérées. Consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Testez les options de placement des sessions de jeu pour les flottes gérées, notamment en personnalisant les paramètres de priorisation. Consultez [Personnaliser une file d'attente de session de jeu](#).
- Configurez le dimensionnement automatique de la capacité pour répondre à la demande attendue des joueurs. Consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).
- Créez des flottes dans d'autres, modifiez les files d'attente Régions AWS et le dimensionnement automatique pour gérer les basculements selon les besoins.
- Configurez les outils d'observabilité de l'hébergement, notamment les outils d'analyse et de journalisation. Consultez [Surveillance des Amazon GameLift Servers](#).
- Automatisez les déploiements de votre flotte à l'aide de l'[infrastructure en tant que code \(IaC\)](#). Consultez [Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation](#).

Amazon GameLift Servers prend en charge l'utilisation de AWS CloudFormation modèles pour toutes les configurations spécifiques au déploiement. Vous pouvez également utiliser le AWS Cloud Development Kit (AWS CDK) pour définir vos Amazon GameLift Servers ressources. Pour

plus d'informations à ce sujet AWS CDK, consultez le [guide du AWS Cloud Development Kit \(AWS CDK\) développeur](#).

Pour gérer le déploiement de vos AWS CloudFormation stacks, nous vous recommandons d'utiliser des outils et services d'intégration continue et de livraison continue (CI/CD) tels que AWS CodePipeline. Ces outils vous permettent de déployer automatiquement ou avec approbation chaque fois que vous créez le binaire d'un serveur de jeu. Avec un CI/CD outil ou un service, le déploiement des ressources pour une nouvelle version de serveur de jeu peut ressembler à ceci :

- Créez et testez le binaire de votre serveur de jeu.
- Téléchargez le fichier binaire sur Amazon GameLift Servers.
- Déployez de nouvelles flottes avec la nouvelle version.
- Ajoutez les nouvelles flottes à votre file d'attente de session de jeu et supprimez les flottes de la version précédente.
- Lorsque les flottes de la version précédente n'hébergent plus de sessions de jeu actives, supprimez les AWS CloudFormation piles de ces flottes.

Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Anywhere

Cette feuille de route vous explique comment développer une solution d'hébergement pour votre jeu multijoueur à utiliser avec vos propres ressources (matériel sur site ou machines virtuelles). Amazon GameLift Servers propose plusieurs options d'hébergement de jeux ; pour plus d'informations sur ces options, consultez [Amazon GameLift Servers solutions](#).

Avec l'hébergement Amazon GameLift Servers Anywhere, votre serveur de jeu est hébergé sur des ressources informatiques que vous fournissez et gérez. Vous pouvez créer une flotte Anywhere avec les configurations dont vous avez besoin et géographiquement, quel que soit l'endroit où se trouvent vos joueurs. Amazon GameLift Servers fournit les fonctionnalités suivantes pour une flotte n'importe où :

- Gère le processus de placement des sessions de jeu pour vous en fonction de votre configuration, notamment :
 - Suivi de la disponibilité des serveurs de jeu dans l'ensemble de vos flottes Anywhere.
 - Traitement des demandes de jeu émanant du service client de votre jeu et mise en correspondance des demandes de jeu avec les serveurs disponibles.

- Inviter les serveurs de jeu des flottes Anywhere à démarrer des sessions de jeu.
- Communiquer les détails de connexion aux clients du jeu.
- Recueillir des indicateurs de performance pour le processus de placement des sessions et des indicateurs d'utilisation pour les sessions de jeu et les joueurs.
- Supporte l'ensemble complet des fonctionnalités de FlexMatch matchmaking, afin que vous puissiez créer un entremetteur et l'intégrer au système de placement des sessions de jeu.
- Permet à l'Amazon GameLift Serversagent de gérer les principales tâches de gestion des hôtes sur une flotte Anywhere.
- Supporte la combinaison avec des flottes Amazon GameLift Servers gérées pour une solution hybride flexible.

Une solution Amazon GameLift Servers Anywhere est composée des composants suivants :

- Une ou plusieurs flottes Amazon GameLift Servers Anywhere contenant vos ressources sur site ou d'autres ressources d'hébergement, gérées à l'aide de vos outils de gestion de configuration et de déploiement existants. (Vous pouvez éventuellement utiliser le AWS Systems Manager.)
- Une version de serveur de jeu, intégrée au SDK du serveur pour Amazon GameLift Servers, à déployer sur toutes les flottes.
- Un client de jeu et un service principal, intégrés au AWS SDK, pour interagir avec le Amazon GameLift Servers service et demander des sessions de jeu.
- Une Amazon GameLift Servers file d'attente pour placer de nouvelles sessions de jeu avec les serveurs de jeu disponibles dans toutes les flottes.
- (Facultatif) Un FlexMatch système de matchmaking pour créer des matchs multijoueurs et configurer des sessions de jeu pour eux.

Cette feuille de route présente une approche simplifiée pour que votre jeu multijoueur soit opérationnel avec succès grâce à l'hébergement Amazon GameLift Servers Anywhere. Une fois les composants nécessaires en place, vous pouvez continuer à développer le jeu et à personnaliser votre solution d'hébergement. À l'approche du lancement, consultez les [Préparation de votre jeu pour le lancement grâce à l'Amazon GameLift Servershébergement](#) pour vous aider à préparer votre solution d'hébergement pour une utilisation en production.

- i** Démarrez rapidement avec le Amazon GameLift Servers plugin pour Unreal Engine et Unity
- Pour un déploiement plus rapide, essayez le [Amazon GameLift Serversplugin](#) pour Unreal Engine et Unity. Il fournit des flux de travail d'interface utilisateur guidés pour déployer rapidement votre serveur de jeu avec une configuration minimale, afin que vous puissiez tester les composants de votre jeu en action. Vous pouvez ensuite vous appuyer sur cette base pour créer une solution d'hébergement personnalisée pour votre jeu. Pour en savoir plus, consultez [Explorez avec le Amazon GameLift Servers ..](#)

Étape 1 : Préparez votre serveur de jeu pour qu'il fonctionne Amazon GameLift Servers

Ajoutez des fonctionnalités à votre serveur de jeu afin qu'il puisse communiquer avec le Amazon GameLift Servers service lorsqu'il est déployé pour l'hébergement.

- Procurez-vous le SDK du serveur pour Amazon GameLift Servers (version 5.x) pour votre projet de jeu. Le SDK du serveur est disponible en C++, C# et Go. [Téléchargez un SDK Amazon GameLift Servers de serveur](#).
- Modifiez le code de votre serveur de jeu pour ajouter les fonctionnalités du SDK du serveur. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
 - Ajoutez du code pour initialiser le Amazon GameLift Servers SDK et établir une WebSocket connexion avec le Amazon GameLift Servers service. Utilisez l'action du SDK du serveur `InitSdk()` et incluez les paramètres du serveur, qui sont requis pour une flotte Anywhere.
 - Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur est prêt à héberger des sessions de jeu. Utilisez l'action `ProcessReady()` du SDK du serveur.
 - Implémentez les fonctions de rappel `requisitesOnProcessTerminate()`, et `OnStartGameSession()` Grâce à ces fonctions, les processus du serveur de jeu peuvent maintenir une connexion avec le Amazon GameLift Servers service, lancer une session de jeu lorsque cela Amazon GameLift Servers est demandé et répondre à une demande d'arrêt du processus du serveur de jeu.
 - Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur met fin à une session de jeu. Utilisez l'action `ProcessEnding()` du SDK du serveur.

- Package de la version de votre serveur de jeu. Créez un script d'installation avec vos fichiers de compilation, vos dépendances et les logiciels de support. Consultez [Conditionner les fichiers de version de jeu](#). Nous vous recommandons d'utiliser un bucket Amazon Simple Storage Service (Amazon S3) pour stocker les versions de votre build de jeu.
- Testez l'intégration de votre serveur de jeu. Pour cette tâche, nous vous recommandons de configurer un parc Amazon GameLift Servers Anywhere pour un poste de travail local, comme décrit dans [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#). Pour cette étape, installez manuellement le build de votre serveur de jeu sur le périphérique de test et lancez un processus serveur. Utilisez la AWS CLI pour demander une nouvelle session de jeu et vérifiez que le Amazon GameLift Servers service invite correctement votre processus serveur à démarrer une session de jeu.

Étape 2 : Préparez votre client de jeu pour qu'il rejoigne les sessions de jeu hébergées

Créez un moyen pour votre client de jeu de demander à rejoindre une session de jeu, d'obtenir des informations de connexion, puis de se connecter directement à une session de jeu hébergée. L'approche la plus courante consiste à configurer une fonctionnalité de service principal qui sert d'intermédiaire entre le client de jeu et le service. Amazon GameLift Servers Cette approche protège vos ressources d'hébergement et vous permet de mieux contrôler la manière dont les joueurs sont placés dans les sessions de jeu.

- Développez des fonctionnalités de service principal pour l'hébergement. Le service principal communique avec le Amazon GameLift Servers service et fournit des informations de connexion à un client de jeu. Cette fonctionnalité inclut le démarrage de sessions de jeu, le placement de joueurs dans des jeux et la récupération des informations de session de jeu. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
 - Procurez-vous le AWS SDK pour Amazon GameLift Servers et ajoutez-le à votre projet de service principal. Consultez les [ressources du Amazon GameLift Servers SDK pour les services clients](#).
 - Ajoutez du code pour initialiser un Amazon GameLift Servers client et enregistrer les paramètres clés. Consultez [Configuration Amazon GameLift Servers sur un service principal](#).
 - Ajoutez une fonctionnalité permettant d'appeler l'action du AWS SDK `CreateGameSession()` et de fournir des informations de connexion à une session de jeu à un client de jeu. Voir [Créer une session de jeu sur une flotte spécifique](#).

`CreateGameSession()` L'appel est un point de départ pratique pour demander de nouvelles sessions de jeu. Une fois que vous aurez mis en place un système de placement de sessions de jeu (voir étape 3), vous remplacerez ce code par un appel à `StartGameSessionPlacement()` (ou `StartMatchmaking()` si vous utilisez `FlexMatch`).

Pour obtenir des conseils sur la conception de votre service principal, consultez [Concevez le service client de votre jeu](#).

- Ajoutez des fonctionnalités à votre client de jeu qui permettent aux joueurs de rejoindre une session de jeu hébergée. Le client du jeu adresse des requêtes à votre service principal, et non directement à Amazon GameLift Servers. Une fois que le service principal a fourni les informations de connexion à la session de jeu, le client du jeu se connecte directement à la session de jeu pour jouer au jeu.
- Testez l'intégration de votre client de jeu. Vous pouvez utiliser le même parc Amazon GameLift Servers Anywhere avec des postes de travail locaux à des fins de test.

Étape 3 : configurer le placement des sessions de jeu

Personnalisez la façon dont vous Amazon GameLift Servers souhaitez traiter les demandes de nouvelle session de jeu et localisez les serveurs de jeu disponibles pour les héberger. Amazon GameLift Servers suit automatiquement la disponibilité de tous les serveurs de jeu sur toutes les flottes. Lorsqu'un client de jeu envoie une demande pour rejoindre une session de jeu, Amazon GameLift Servers recherche le « meilleur emplacement possible » en fonction d'un ensemble de priorités définies telles que le minimum de latence, de coût et de disponibilité.

- Créez une file d'attente de session de jeu pour placer une nouvelle session de jeu parmi les serveurs de jeu disponibles. Les files d'attente sont le principal mécanisme de placement des sessions de jeu. Pour de plus amples informations, consultez [Création d'une file d'attente de session de jeu](#).
- Ajoutez au minimum vos flottes Anywhere en tant que destinations dans votre file d'attente. Tous les autres paramètres sont facultatifs.
- Dans votre code de service principal, convertissez l'`CreateGameSession()` appel en `StartGameSessionPlacement()`. Consultez la section [Création d'une session de jeu dans une file d'attente multisite](#).
- Créez un mécanisme pour avertir un client de jeu lorsqu'une session de jeu est prête à être ouverte. Pendant le développement, vous pouvez demander l'état d'une session de jeu en appelant

à `DescribeGameSessionPlacement`. Avant d'utiliser une file d'attente pour traiter des volumes importants, vous devez toutefois activer les notifications d'événements. Consultez [Configurer la notification des événements pour le placement des sessions de jeu](#).

- (Facultatif) Ajoutez des composants FlexMatch de matchmaking. Pour obtenir des conseils, consultez le [guide du Amazon GameLift ServersFlexMatch développeur](#).

Étape 4 : Configuration d'une flotte Anywhere avec l'Amazon GameLift Serversagent

Jusqu'à présent, vous avez travaillé avec des appareils locaux (enregistrés sous le nom de Anywhere fleet computes) pour tester et itérer les composants de votre jeu. L'étape suivante consiste à configurer le type de flotte dont vous aurez besoin pour un système de production. Pour ces ressources, utilisez l'Amazon GameLift Serversagent pour gérer certaines tâches clés de gestion de l'hôte sur ordinateur. Pour en savoir plus, consultez [Collaborez avec l'Amazon GameLift Serversagent](#).

- Procurez-vous l'Amazon GameLift Serversagent et ajoutez-le au package d'installation de votre serveur de jeu. Obtenez et compilez le code source de l'agent, disponible dans le [référentiel Github de l'Amazon GameLift Serversagent](#). Placez le fichier exécutable JAR obtenu dans le même répertoire que le fichier exécutable de votre build de jeu.
- Modifiez le script de démarrage de l'agent selon vos besoins. Assurez-vous que le fichier exécutable de l'agent est lancé dès qu'un calcul commence à s'exécuter. Consultez le fichier readme dans le dépôt de l'agent pour obtenir de l'aide concernant l'installation et l'exécution de l'agent sur vos ordinateurs d'hébergement. Votre commande de lancement doit inclure des options permettant de spécifier, au minimum, l'ID de flotte Anywhere Région AWS, un emplacement personnalisé et un nom de calcul.

L'agent gère automatiquement les tâches suivantes pour vous. Par conséquent, si vous les avez gérées à l'aide de scripts, vous pouvez les supprimer :

- Appels `RegisterCompute()` pour ajouter le calcul à une flotte Anywhere.
- Appels `GetComputeAuthToken()` pour authentifier les serveurs de jeu lorsqu'ils se connectent au Amazon GameLift Servers service. L'agent gère l'obtention et l'actualisation du jeton d'authentification, qui peut être utilisé par tous les processus du serveur de jeu exécutés sur l'ordinateur.
- Démarre de nouveaux processus serveur sur le calcul en fonction d'un ensemble d'instructions d'exécution.

- Créez une configuration d'exécution pour les calculs de votre flotte Anywhere. Vous pouvez utiliser la Amazon GameLift Servers console ou la AWS CLI pour créer ou modifier des instructions d'exécution pour le parc. L'agent exécute ces instructions et demande régulièrement des mises à jour au Amazon GameLift Servers service.
- Configurez ou modifiez la file d'attente de votre session de jeu selon vos besoins. Créez une nouvelle file d'attente (ou mettez à jour une file existante) pour utiliser les flottes Anywhere déployées avec l'Amazon GameLift Servers agent.
- Testez l'intégration de l'agent à vos flottes Anywhere. Vérifiez que l'agent démarre correctement les processus du serveur en fonction de la configuration d'exécution.

Étape 5 : Personnalisez vos flottes Anywhere

Alors que vous vous préparez au lancement du jeu, vous devrez affiner vos ressources d'hébergement géré. Parmi les décisions à prendre en compte, mentionnons les suivantes :

- Automatisez le processus de démarrage et d'arrêt des ordinateurs selon les besoins, notamment en installant et en exécutant le logiciel du serveur de jeu. Le recyclage des ordinateurs est utile pour garantir leur mise à jour régulière, et l'arrêt des ordinateurs peut permettre de réduire les coûts lorsqu'ils ne sont pas nécessaires.
- Si votre serveur de jeu doit communiquer avec d'autres AWS ressources, configurez des rôles IAM pour gérer l'accès. Consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
- Déterminez l'emplacement géographique où vous souhaitez positionner les serveurs de jeu. Ajoutez des sites distants à vos flottes gérées. Consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Optimisez les performances du parc en sélectionnant des configurations de ressources de calcul, puis en configurant les instructions d'exécution pour exécuter un nombre optimal de processus serveur par calcul.
- Testez les options de placement des sessions de jeu pour les flottes gérées, notamment en personnalisant les paramètres de priorisation. Consultez [Personnaliser une file d'attente de session de jeu](#).
- Créez des mécanismes pour gérer la mise à l'échelle manuelle ou automatisée des capacités afin de répondre à la demande attendue des joueurs. Réfléchissez aux facteurs qui devraient inciter le système à augmenter ou à diminuer le nombre d'ordinateurs disponibles pour héberger des sessions de jeu.
- Concevez et implémentez le basculement vers d'autres ressources si nécessaire.

- Configurez les outils d'observabilité de l'hébergement, notamment les outils d'analyse et de journalisation. Consultez [Surveillance des Amazon GameLift Servers](#).

Feuille de route de développement pour l'hébergement hybride avec Amazon GameLift Servers

Cette feuille de route vous explique comment développer une solution d'hébergement pour votre jeu multijoueur. Amazon GameLift Servers propose plusieurs options d'hébergement de jeux ; pour plus d'informations sur ces options, consultez [Amazon GameLift Servers solutions](#).

Une solution hybride utilise une combinaison de ressources d'hébergement, notamment des ressources basées sur le cloud gérées par Amazon GameLift Servers et vos propres ressources d'hébergement autogérées. Pour une discussion plus détaillée sur l'hébergement hybride, consultez cet article : [Hébergement de serveurs de jeux hybrides avec Amazon GameLift Servers Anywhere](#). Vous pouvez ainsi configurer une solution hybride qui utilise des composants et des processus communs, afin de gérer de manière centralisée un parc mondial et de déplacer facilement des charges entre tous les types de ressources. Amazon GameLift Servers

Une architecture hybride comprend les composants suivants :

- Une ou plusieurs flottes Amazon GameLift Servers gérées, qui utilisent des instances Amazon Elastic Compute Cloud (Amazon EC2) optimisées pour l'hébergement de jeux multijoueurs.
- Une ou plusieurs flottes Amazon GameLift Servers Anywhere, qui utilisent vos ressources d'hébergement existantes sur site ou autres, y compris vos outils de gestion de configuration et de déploiement. (Vous pouvez éventuellement utiliser le AWS Systems Manager.)
- Une version de serveur de jeu unique, intégrée au SDK du serveur pour Amazon GameLift Servers, à déployer sur toutes les flottes.
- Un client de jeu unique et un service principal, intégrés au AWS SDK, pour interagir avec le Amazon GameLift Servers service et demander des sessions de jeu.
- Une Amazon GameLift Servers file d'attente partagée pour placer de nouvelles sessions de jeu sur les serveurs de jeu disponibles et équilibrer la charge entre toutes les flottes.
- L'Amazon GameLift Servers agent, qui est déployé avec une flotte Anywhere, simplifie les tâches de gestion des processus de serveur sur les ordinateurs de tous les flottes.
- (Facultatif) Un FlexMatch système de matchmaking pour créer des matchs multijoueurs et configurer des sessions de jeu pour eux.

Cette feuille de route présente une voie simplifiée pour que votre jeu multijoueur soit opérationnel avec succès dans une solution d'hébergement hybride avec Amazon GameLift Servers. Une fois les composants nécessaires en place, vous pouvez continuer à développer le jeu et à personnaliser votre solution d'hébergement. À l'approche du lancement, consultez-les [Préparation de votre jeu pour le lancement grâce à l'Amazon GameLift Servers hébergement](#) pour vous aider à préparer votre solution d'hébergement pour une utilisation en production.

Démarrez rapidement avec le Amazon GameLift Servers plugin

Si vous développez des projets avec Unreal Engine ou Unity, commencez à configurer votre jeu pour l'hébergement à l'aide du Amazon GameLift Servers plugin. Avec le plugin, vous pouvez l'ajouter Amazon GameLift Servers SDKs à votre projet de jeu et utiliser les flux de travail guidés pour créer une version simple et fonctionnelle d'une solution d'hébergement hybride avec à la fois une flotte Anywhere et une flotte Amazon GameLift Servers gérée. Vous pouvez ensuite utiliser ces principes fondamentaux pour les développer et les personnaliser selon vos besoins.

Étape 1 : Préparez votre serveur de jeu pour qu'il fonctionne Amazon GameLift Servers

Ajoutez des fonctionnalités à votre serveur de jeu afin qu'il puisse communiquer avec le Amazon GameLift Servers service lorsqu'il est déployé pour l'hébergement. La même fonctionnalité est requise si le serveur de jeu fonctionne sur une flotte Amazon GameLift Servers gérée ou sur une flotte Anywhere.

- Procurez-vous le SDK du serveur pour Amazon GameLift Servers (version 5.x) pour votre projet de jeu. Le SDK du serveur est disponible en C++, C# et Go. [Téléchargez le SDK du serveur pour Amazon GameLift Servers](#).
- Modifiez le code de votre serveur de jeu pour ajouter les fonctionnalités du SDK du serveur. Pour de plus amples informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :
 - Ajoutez du code pour initialiser le Amazon GameLift Servers SDK et établir une WebSocket connexion avec le Amazon GameLift Servers service. Utilisez l'action `InitSdk()` du SDK du serveur. Incluez du code pour spécifier les paramètres du serveur lors de l'exécution sur un système de calcul de flotte Anywhere.

- Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur est prêt à héberger des sessions de jeu. Utilisez l'action `ProcessReady()` du SDK du serveur.
- Implémentez les fonctions de rappel `requisitesOnProcessTerminate()`, et `OnStartGameSession()`. Grâce à ces fonctions, les processus du serveur de jeu peuvent maintenir une connexion avec le Amazon GameLift Servers service, lancer une session de jeu lorsque cela Amazon GameLift Servers est demandé et répondre à une demande d'arrêt du processus du serveur de jeu.
- Ajoutez du code à signaler au Amazon GameLift Servers service lorsque le processus du serveur met fin à une session de jeu. Utilisez l'action `ProcessEnding()` du SDK du serveur.
- Package de la version de votre serveur de jeu. Créez un script d'installation avec vos fichiers de compilation, vos dépendances et les logiciels de support. Consultez [Conditionner les fichiers de version de jeu](#). Nous vous recommandons d'utiliser un bucket Amazon Simple Storage Service (Amazon S3) pour stocker les versions de votre build de jeu.
- Testez l'intégration de votre serveur de jeu. Pour cette tâche, nous vous recommandons de configurer un parc Amazon GameLift Servers Anywhere avec un poste de travail local, comme décrit dans [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#). Pour cette étape, installez manuellement le build de votre serveur de jeu sur le périphérique de test et lancez un processus serveur. Utilisez la AWS CLI pour demander une nouvelle session de jeu et vérifiez que le Amazon GameLift Servers service invite correctement votre processus serveur à démarrer une session de jeu.

Étape 2 : Préparez votre client de jeu pour qu'il rejoigne les sessions de jeu hébergées

Créez un moyen pour votre client de jeu de demander à rejoindre une session de jeu, d'obtenir des informations de connexion, puis de se connecter directement à une session de jeu hébergée. L'approche la plus courante consiste à configurer une fonctionnalité de service principal qui sert d'intermédiaire entre le client de jeu et le service. Amazon GameLift Servers Cette approche protège vos ressources d'hébergement et vous permet de mieux contrôler la manière dont les joueurs sont placés dans les sessions de jeu.

- Développez des fonctionnalités de service principal pour l'hébergement. Le service principal communique avec le Amazon GameLift Servers service et fournit des informations de connexion à un client de jeu. Cette fonctionnalité inclut le démarrage de sessions de jeu, le placement de joueurs dans des jeux et la récupération des informations de session de jeu. Pour de plus amples

informations, consultez [Intégrez des jeux à des serveurs de jeux personnalisés](#). Procédez au minimum comme suit :

- Procurez-vous le AWS SDK pour Amazon GameLift Servers et ajoutez-le à votre projet de service principal. Consultez les [ressources du Amazon GameLift Servers SDK pour les services clients](#).
- Ajoutez du code pour initialiser un Amazon GameLift Servers client et enregistrer les paramètres clés. Consultez [Configuration Amazon GameLift Servers sur un service principal](#).
- Ajoutez une fonctionnalité permettant d'appeler l'action du AWS SDK `CreateGameSession()` et de fournir des informations de connexion à une session de jeu à un client de jeu. Consultez la section [Créer une session de jeu sur une flotte spécifique](#).

`CreateGameSession()` L'appel est un point de départ pratique pour demander de nouvelles sessions de jeu. Une fois que vous aurez mis en place un système de placement de sessions de jeu (voir étape 3), vous remplacerez ce code par un appel à `StartGameSessionPlacement()` (ou `StartMatchmaking()` si vous utilisez `FlexMatch`).

Pour obtenir des conseils sur la conception de votre service principal, consultez [Concevez le service client de votre jeu](#).

- Ajoutez des fonctionnalités à votre client de jeu qui permettent aux joueurs de rejoindre une session de jeu hébergée. Le client du jeu adresse des requêtes à votre service principal, et non directement à Amazon GameLift Servers. Une fois que le service principal a fourni les informations de connexion à la session de jeu, le client du jeu se connecte directement à la session de jeu pour jouer au jeu.
- Testez l'intégration de votre client de jeu. Vous pouvez utiliser le même parc Amazon GameLift Servers Anywhere avec un poste de travail local à des fins de test.

Pendant la phase de développement, si vous souhaitez tester le comportement de votre build de jeu dans une flotte Amazon GameLift Servers gérée, nous vous recommandons de configurer également un [environnement de test basé sur le cloud](#). Cette solution Amazon GameLift Servers Toolkit imite le comportement d'une flotte gérée mais vous permet de mettre à jour les versions des serveurs de jeu avec un délai d'exécution minimal.

Étape 3 : configurer le placement des sessions de jeu

Personnalisez la façon dont vous Amazon GameLift Servers souhaitez traiter les demandes de nouvelle session de jeu et localisez les serveurs de jeu disponibles pour les héberger. Amazon

GameLift Servers suit automatiquement la disponibilité de tous les serveurs de jeu sur toutes les flottes. Lorsqu'un client de jeu envoie une demande pour rejoindre une session de jeu, Amazon GameLift Servers recherche le « meilleur emplacement possible » en fonction d'un ensemble de priorités définies telles que le minimum de latence, de coût et de disponibilité.

- Créez une file d'attente de session de jeu pour placer une nouvelle session de jeu parmi les serveurs de jeu disponibles. Les files d'attente sont le principal mécanisme de placement des sessions de jeu. Pour de plus amples informations, consultez [Création d'une file d'attente de session de jeu](#).
- Ajoutez au minimum vos flottes Anywhere en tant que destinations dans votre file d'attente. Tous les autres paramètres sont des personnalisations facultatives.
- Dans votre code de service principal, convertissez l'`CreateGameSession()` appel en `StartGameSessionPlacement()`. Consultez la section [Création d'une session de jeu dans une file d'attente multisite](#).
- Créez un mécanisme pour avertir un client de jeu lorsqu'une session de jeu est prête à être ouverte. Pendant le développement, vous pouvez demander l'état d'une session de jeu en appelant `DescribeGameSessionPlacement`. Avant d'utiliser une file d'attente pour traiter des volumes importants, vous devez toutefois activer les notifications d'événements. Consultez [Configurer la notification des événements pour le placement des sessions de jeu](#).
- (Facultatif) Ajoutez des composants FlexMatch de matchmaking. Pour obtenir des conseils, consultez le [guide du Amazon GameLift Servers FlexMatch développeur](#).

Étape 4 : Configuration d'une flotte Anywhere avec l'Amazon GameLift Serversagent

Jusqu'à présent, vous avez travaillé avec des appareils locaux (enregistrés sous le nom de Anywhere fleet computes) pour tester et itérer les composants de votre jeu. L'étape suivante consiste à configurer le type de flottes dont vous aurez besoin pour un système de production. Commencez par une flotte Anywhere, puis ajoutez l'Amazon GameLift Serversagent pour gérer certaines tâches clés de gestion des hôtes sur ordinateur. Pour en savoir plus, consultez [Collaborez avec l'Amazon GameLift Serversagent](#).

- Procurez-vous l'Amazon GameLift Serversagent et ajoutez-le au package d'installation de votre serveur de jeu. Obtenez et compilez le code source de l'agent, disponible dans le [référentiel Github de l'Amazon GameLift Serversagent](#). Placez le fichier exécutable JAR obtenu dans le même répertoire que le fichier exécutable de votre build de jeu.

- Modifiez le script de démarrage de l'agent selon vos besoins. Assurez-vous que le fichier exécutable de l'agent est lancé dès qu'un calcul commence à s'exécuter. Consultez le fichier readme dans le dépôt de l'agent pour obtenir de l'aide concernant l'installation et l'exécution de l'agent sur vos ordinateurs d'hébergement. Votre commande de lancement doit inclure des options permettant de spécifier, au minimum, l'ID de flotte Anywhere Région AWS, un emplacement personnalisé et un nom de calcul.

L'agent gère automatiquement les tâches suivantes pour vous. Par conséquent, si vous les avez gérées à l'aide de scripts, vous pouvez les supprimer :

- Appels `RegisterCompute()` pour ajouter le calcul à une flotte Anywhere.
- Appels `GetComputeAuthToken()` pour authentifier les serveurs de jeu lorsqu'ils se connectent au Amazon GameLift Servers service. L'agent gère l'obtention et l'actualisation du jeton d'authentification, qui peut être utilisé par tous les processus du serveur de jeu exécutés sur l'ordinateur.
- Démarre de nouveaux processus serveur sur le calcul en fonction d'un ensemble d'instructions d'exécution.
- Créez une configuration d'exécution pour les calculs de votre flotte Anywhere. Spécifiez au minimum le chemin de lancement de l'exécutable de votre serveur de jeu. Vous pouvez utiliser la Amazon GameLift Servers console ou la AWS CLI pour créer ou modifier des instructions d'exécution pour le parc. L'agent exécute ces instructions et demande régulièrement des mises à jour au Amazon GameLift Servers service.
- Configurez ou modifiez la file d'attente de votre session de jeu selon vos besoins. Créez une nouvelle file d'attente (ou mettez à jour une file existante) et désignez une destination pour le parc Anywhere déployé avec l'Amazon GameLift Serversagent.
- Testez l'intégration de l'agent à vos flottes Anywhere. Vérifiez que l'agent démarre correctement les processus du serveur en fonction de la configuration d'exécution.

Étape 5 : créer une flotte gérée dans le cloud

Créez une EC2 flotte Amazon GameLift Servers gérée pour compléter votre flotte Anywhere. Si vous avez configuré un environnement de test basé sur le cloud à l'étape 2 pour accélérer le développement, prévoyez de créer une flotte gérée une fois que vous aurez terminé la majeure partie du développement et des tests de votre jeu. Vous avez besoin d'un parc entièrement géré pour configurer et tester des paramètres supplémentaires tels que le dimensionnement automatique de la capacité.

- Package de la compilation de votre serveur de jeu et de son téléchargement sur Amazon GameLift Servers. Créez un script d'installation avec vos fichiers de compilation, vos dépendances et les logiciels de support. Vous pouvez utiliser le même logiciel de compilation avec vos flottes Anywhere et gérées. Consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#). Vous pouvez télécharger votre build à Amazon GameLift Servers l'aide de la console ou de la AWS CLI.

Avant de télécharger votre build, déterminez de quelle manière Région AWS vous souhaitez créer le parc géré. Vous devez télécharger le build dans la même région. Pour en savoir plus sur le choix de l'emplacement d'une flotte, voir [Emplacement de la flotte](#).

- Créez une EC2 flotte gérée. Vous pouvez utiliser la Amazon GameLift Servers console ou la AWS CLI pour créer un parc géré. Lorsque vous créez une flotte, commencez Amazon GameLift Servers immédiatement à déployer la version de votre serveur de jeu pour l'hébergement. Vous pouvez configurer de nombreux aspects d'un parc géré. Pour de plus amples informations, consultez [Création d'une EC2 flotte Amazon GameLift Servers gérée](#). Procédez au minimum comme suit :
 - Donnez un nom à la flotte et spécifiez la version de jeu téléchargée à déployer.
 - Choisissez des instances à la demande pour votre flotte et sélectionnez un type d'instance disponible dans l'emplacement de votre flotte. Les flottes Spot constituent une option intéressante, mais nécessitent une conception et une configuration supplémentaires.
 - Créez une configuration d'exécution avec des paramètres similaires à ceux que vous avez utilisés avec le parc Anywhere. Spécifiez au minimum le chemin de lancement de l'exécutable de votre serveur de jeu.
 - Spécifiez les paramètres de port pour autoriser le trafic entrant à accéder à vos serveurs de jeu.
- Ajoutez le parc géré à votre file d'attente de session de jeu partagée. Mettez à jour la file d'attente à partir de l'étape 4 afin qu'elle inclue les destinations de la flotte gérée et de la flotte Anywhere déployée avec l'Amazon GameLift Serversagent.
- Testez l'hébergement de jeux avec vos flottes gérées. À ce stade, vous devriez être en mesure de tester l'ensemble du cycle d'hébergement, avec un client de jeu demandant une session de jeu, obtenant des informations de connexion et se connectant avec succès à une session de jeu.

Étape 6 : Personnalisez vos flottes

Alors que vous vous préparez au lancement du jeu, vous devrez peaufiner vos solutions d'hébergement. Parmi les décisions à prendre en compte, mentionnons les suivantes :

- Pour les flottes Anywhere, automatisez le processus de démarrage et d'arrêt des ordinateurs selon les besoins, y compris l'installation et l'exécution du logiciel du serveur de jeu. Le recyclage des ordinateurs est utile pour garantir leur mise à jour régulière, et leur arrêt peut permettre de réduire les coûts lorsqu'ils ne sont pas nécessaires.
- Si votre serveur de jeu doit communiquer avec d'autres AWS ressources, configurez des rôles IAM pour gérer l'accès. Consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
- Déterminez l'emplacement géographique où vous souhaitez positionner les serveurs de jeu. Ajoutez des sites distants à vos flottes gérées. Consultez [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#).
- Pour les flottes gérées, pensez à utiliser des flottes Spot pour réaliser des économies. Consultez [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#).
- Optimisez les performances du parc en sélectionnant des configurations de ressources de calcul, puis en configurant vos instructions d'exécution pour exécuter le nombre optimal de processus serveur par calcul. Procédez ainsi pour les flottes Anywhere et les flottes gérées. Consultez [Gérez comment Amazon GameLift Servers lance des serveurs de jeu](#).
- Testez les options de placement des sessions de jeu pour les flottes gérées, notamment en personnalisant les paramètres de priorisation. Consultez [Personnaliser une file d'attente de session de jeu](#).
- Pour les flottes gérées, configurez le dimensionnement automatique de la capacité pour répondre à la demande attendue des joueurs. Consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).
- Pour les flottes Anywhere, créez des mécanismes permettant de gérer la mise à l'échelle manuelle ou automatisée des capacités afin de répondre à la demande attendue des joueurs.
- Concevez et implémentez le basculement vers d'autres ressources si nécessaire. Configurez des flottes de réserve dans d'autres, modifiez les files d'attente Régions AWS et le dimensionnement automatique pour gérer les basculements si nécessaire.
- Configurez les outils d'observabilité de l'hébergement, notamment les outils d'analyse et de journalisation. Consultez [Surveillance des Amazon GameLift Servers](#). Créez des groupes de mesures pour agréger les analyses de toutes vos ressources d'hébergement.
- Automatisez votre déploiement à l'aide de l'[infrastructure en tant que code \(laC\)](#). Consultez [Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation](#).

Amazon GameLift Servers prend en charge l'utilisation de AWS CloudFormation modèles pour toutes les configurations spécifiques au déploiement. Vous pouvez également utiliser le AWS Cloud Development Kit (AWS CDK) pour définir vos Amazon GameLift Servers ressources. Pour

plus d'informations à ce sujet AWS CDK, consultez le [guide du AWS Cloud Development Kit \(AWS CDK\) développeur](#).

Pour gérer le déploiement de vos AWS CloudFormation stacks, nous vous recommandons d'utiliser des outils et services d'intégration continue et de livraison continue (CI/CD) tels que. AWS CodePipeline Ces outils vous permettent de déployer automatiquement ou avec approbation chaque fois que vous créez le binaire d'un serveur de jeu. Avec un CI/CD outil ou un service, le déploiement des ressources pour une nouvelle version de serveur de jeu peut ressembler à ceci :

- Créez et testez le binaire de votre serveur de jeu.
- Téléchargez le fichier binaire sur Amazon GameLift Servers.
- Déployez de nouvelles flottes avec la nouvelle version.
- Ajoutez les nouvelles flottes à votre file d'attente de session de jeu et supprimez les flottes de la version précédente.
- Lorsque les flottes de la version précédente n'hébergent plus de sessions de jeu actives, supprimez les AWS CloudFormation piles de ces flottes.

Préparation de jeux pour Amazon GameLift Servers

Préparez vos jeux multijoueurs pour l'hébergement sur Amazon GameLift Servers. Intégrer Amazon GameLift Servers héberger des fonctionnalités dans vos projets de jeu et créer votre serveur de jeu et vos serveurs clients. Configurez un environnement de test hébergé pour prendre en charge le développement et les tests itératifs rapides de jeux.

Rubriques

- [Intégrez des jeux à des serveurs de jeux personnalisés](#)
- [Concevez le service client de votre jeu](#)
- [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#)
- [Testez votre intégration à l'aide de Amazon GameLift Servers Local](#)
- [Ajouter le FlexMatch matchmaking](#)
- [Obtenir des données de flotte pour une Amazon GameLift Servers instance](#)
- [Intégrer des jeux avec Amazon GameLift ServersRealtime](#)

Intégrez des jeux à des serveurs de jeux personnalisés

Amazon GameLift Servers fournit un ensemble complet d'outils pour préparer vos jeux multijoueurs et des serveurs de jeux personnalisés pour qu'ils s'exécutent sur Amazon GameLift Servers. Le Amazon GameLift Servers SDKs contiennent les bibliothèques nécessaires aux clients et serveurs de jeux pour communiquer avec Amazon GameLift Servers. Pour plus d'informations sur SDKs et où les obtenir, consultez [Obtenez des outils Amazon GameLift Servers de développement](#).

Les rubriques de cette section contiennent des instructions détaillées sur la façon d'ajouter Amazon GameLift Servers fonctionnalité sur votre client de jeu et votre serveur de jeu avant le déploiement sur Amazon GameLift Servers.

Rubriques

- [Interactions entre le client et le serveur du jeu avec Amazon GameLift Servers](#)
- [Intégrez votre serveur de jeu avec Amazon GameLift Servers](#)
- [Intégrez votre client de jeu à Amazon GameLift Servers](#)
- [Moteurs de jeu et Amazon GameLift Servers](#)

Interactions entre le client et le serveur du jeu avec Amazon GameLift Servers

Les composants de votre Amazon GameLift Servers les solutions d'hébergement interagissent les unes avec les autres de manière spécifique pour exécuter des sessions de jeu en réponse à la demande des joueurs. Cette rubrique décrit comment les composants communiquent entre eux lorsque votre serveur de jeu est hébergé sur Amazon GameLift Servers EC2 flottes gérées, autogérées Amazon GameLift Servers Des flottes n'importe où ou une solution hybride.

Les composants de la solution d'hébergement incluent un serveur de jeu, Amazon GameLift Servers service, un service principal côté client et un client de jeu. Le serveur de jeu utilise [Amazon GameLift Servers SDK du serveur](#) pour interagir avec Amazon GameLift Servers service. Le service principal utilise [Amazon GameLift Servers API de service](#) (partie du AWS SDK) pour interagir avec le service pour le compte du client du jeu. Lorsque vous rejoignez une session de jeu, le client de jeu se connecte directement à une session de jeu en utilisant l'adresse IP et le numéro de port uniques de la session de jeu.

Rubriques

- [Schéma des interactions](#)
- [Comportements d'interaction](#)

Schéma des interactions

Le schéma suivant illustre la façon dont les composants de votre hébergement de jeux interagissent afin que Amazon GameLift Servers le service peut suivre l'état de disponibilité du serveur de jeu et démarrer des sessions de jeu en réponse aux demandes des joueurs.

Comportements d'interaction

Les sections suivantes décrivent la séquence des événements dans chacune des interactions clés.

Initialisation d'un processus de serveur de jeu

Au démarrage, un processus de serveur de jeu établit une communication avec Amazon GameLift Servers service et indique qu'il est prêt à héberger une session de jeu.

1. Un nouveau processus de l'exécutable du serveur de jeu commence à s'exécuter sur une ressource d'hébergement.

2. Le processus du serveur de jeu appelle les opérations suivantes du SDK du serveur dans l'ordre :
 - a. `InitSDK()` pour initialiser le SDK du serveur, authentifier le processus du serveur et établir une communication avec Amazon GameLift Servers service.
 - b. `ProcessReady()` pour indiquer si vous êtes prêt à organiser une session de jeu. Cet appel indique également les informations de connexion du processus, que les clients du jeu utilisent pour se connecter à la session de jeu, ainsi que d'autres informations.

Le processus du serveur attend ensuite les instructions du Amazon GameLift Servers service.

3. Amazon GameLift Servers met à jour l'état du processus du serveur `ACTIVE` et indique qu'il est disponible pour héberger une nouvelle session de jeu.
4. Amazon GameLift Servers commence à appeler périodiquement le `onHealthCheck` rappel pour demander un état de santé aux processus du serveur. Ces appels se poursuivent tant que le processus du serveur reste actif. Le processus du serveur doit répondre correctement ou non dans un délai d'une minute. Si le processus du serveur ne répond pas correctement ou ne répond pas, à un moment donné, Amazon GameLift Servers le service modifie l'état actif du processus du serveur et arrête d'envoyer des demandes pour démarrer une session de jeu.

Création d'une session de jeu

Le Amazon GameLift Servers le service démarre une nouvelle session de jeu en réponse à une demande d'un joueur de jouer au jeu.

1. Un joueur utilisant le client du jeu demande à rejoindre une session de jeu. En fonction de la façon dont votre jeu gère le processus d'adhésion des joueurs, le client du jeu envoie une demande au service principal.
2. Si le processus d'inscription d'un joueur nécessite le démarrage d'une nouvelle session de jeu, le service principal envoie une demande de nouvelle session de jeu au Amazon GameLift Servers service. Cette demande appelle l'opération de l'API du `serviceStartGameSessionPlacement()`. (Comme alternative, le service principal peut appeler `startMatchmaking()`, ou `createGameSession()`.)
3. Le Amazon GameLift Servers le service répond en créant un nouveau `GameSessionPlacement` ticket avec un statut `PENDING`. Il renvoie les informations des tickets au service principal, afin qu'il puisse suivre l'état du ticket de placement et déterminer quand la

session de jeu est prête pour les joueurs. Pour de plus amples informations, veuillez consulter [Configurer la notification des événements pour le placement des sessions de jeu](#).

4. Le Amazon GameLift Servers le service lance le processus de placement de session de jeu. Il identifie les flottes à examiner et recherche dans ces flottes un processus serveur actif qui n'héberge pas de session de jeu. Lors de la localisation d'un processus serveur disponible, Amazon GameLift Servers le service effectue les opérations suivantes :
 - a. Crée un `GameSession` objet avec les paramètres de session de jeu et les données du joueur provenant de la demande de placement, et définit le statut sur `ACTIVATING`.
 - b. Invite le processus serveur à démarrer une session de jeu. Le service invoque le `onStartGameSession` rappel du processus du serveur et transmet l'`GameSession` objet.
 - c. Modifie le nombre de sessions de jeu du processus serveur en 1.
5. Le processus du serveur exécute sa fonction de `onStartGameSession` rappel. Lorsque le processus serveur est prêt à accepter les connexions des joueurs, il appelle le SDK du serveur à fonctionner `ActivateGameSession()` et attend les connexions des joueurs.
6. Le Amazon GameLift Servers le service met à jour l'`GameSession` objet avec les informations de connexion pour le processus serveur (comme indiqué dans l'appel à `ProcessReady()`) et définit l'état de la session de jeu sur `ACTIVE`. Il met également à jour le statut du `GameSessionPlacement` ticket en `FULFILLED`.
7. Le service principal appelle `DescribeGameSessionPlacement()` pour vérifier l'état des tickets et obtenir des informations sur les sessions de jeu. Lorsque la session de jeu est active, le service principal avertit le client du jeu et transmet les informations de connexion à la session de jeu.
8. Le client du jeu utilise les informations de connexion pour se connecter directement au processus du serveur de jeu et rejoindre la session de jeu.

Ajouter un joueur à une partie

Un jeu peut éventuellement utiliser les sessions des joueurs pour suivre les connexions des joueurs aux sessions de jeu. Les sessions des joueurs peuvent être créées individuellement ou dans le cadre d'une demande de placement de session de jeu.

1. Le service principal appelle l'opération de l'API du service `CreatePlayerSession()` avec un identifiant de session de jeu.

2. Le Amazon GameLift Servers le service vérifie l'état de la session de jeu (ce doit être le casACTIVE) et recherche un emplacement de joueur ouvert pendant la session de jeu. Si un emplacement est disponible, le service effectue les opérations suivantes :
 - a. Crée un nouvel `PlayerSession` objet et définit son statut surRESERVED.
 - b. Répond à la demande de service principal en fournissant des informations sur la session du joueur.
3. Le service principal transmet les informations de session du joueur au client du jeu ainsi que les informations de connexion à la session de jeu.
4. Le client du jeu utilise les informations de connexion et l'identifiant de session du joueur pour se connecter directement au processus du serveur de jeu et demander à rejoindre la session de jeu.
5. En réponse à une tentative de connexion d'un client de jeu, le processus du serveur de jeu appelle l'opération API du service `AcceptPlayerSession()` pour valider l'identifiant de session du joueur. Le processus serveur accepte ou refuse la connexion.
6. Le Amazon GameLift Servers le service effectue l'une des opérations suivantes :
 - a. Si la connexion est acceptée, Amazon GameLift Servers définit le `PlayerSession` statut ACTIVE et transmet le processus `PlayerSession` au serveur de jeu.
 - b. Si le processus du serveur de jeu ne `AcceptPlayerSession()` demande pas l'identifiant de session du joueur dans un certain délai après la `CreatePlayerSession()` demande initiale, le Amazon GameLift Servers le service change le `PlayerSession` statut TIMEDOUT et rouvre l'emplacement du joueur dans la session de jeu.

Supprimer un joueur

Pour les jeux qui utilisent des sessions de jeu, le processus du serveur de jeu indique Amazon GameLift Servers service lorsqu'un joueur se déconnecte. Le service utilise ces informations pour suivre l'état des machines à sous des joueurs au cours d'une session de jeu et peut permettre aux nouveaux joueurs d'utiliser les machines à sous ouvertes.

1. Un joueur se déconnecte de la session de jeu.
2. Le processus du serveur de jeu détecte la perte de connexion et appelle le SDK du serveur à `fonctionnerRemovePlayerSession()`.
3. Le Amazon GameLift Servers le service change l'état de la session du joueur COMPLETED et rouvre l'emplacement du joueur pendant la session de jeu.

Fermeture de la session de jeu

À la fin d'une session de jeu, ou lors de la fermeture de la session de jeu, le processus du serveur notifie Amazon GameLift Servers service d'état de session de jeu.

1. Le processus du serveur de jeu met fin à la session de jeu et initie l'arrêt du processus en appelant le fonctionnement du SDK du serveur. `ProcessEnding()`
2. Le Amazon GameLift Servers le service effectue les opérations suivantes :
 - a. Télécharge les journaux des sessions de jeu sur Amazon Simple Storage Service (Amazon S3).
 - b. Modifie le statut de la session de jeu en `TERMINATED`.
 - c. Modifie l'état du processus du serveur en `TERMINATED`.
 - d. En fonction de la conception de la solution d'hébergement, les nouvelles ressources d'hébergement disponibles sont allouées pour exécuter un nouveau processus de serveur de jeu.

Intégrez votre serveur de jeu avec Amazon GameLift Servers

Une fois que votre serveur de jeu personnalisé est déployé et exécuté sur Amazon GameLift Servers instances, il doit être capable d'interagir avec Amazon GameLift Servers (et éventuellement d'autres ressources). Cette section décrit comment intégrer le logiciel de votre serveur de jeu à Amazon GameLift Servers.

Note

Ces instructions supposent que vous avez créé un Compte AWS et que vous avez un projet de serveur de jeu existant.

Les rubriques de cette section décrivent comment gérer les tâches d'intégration suivantes :

- Établissez la communication entre Amazon GameLift Servers et vos serveurs de jeu.
- Générez et utilisez un certificat TLS pour établir une connexion sécurisée entre le client de jeu et le serveur de jeu.
- Autorisez le logiciel de votre serveur de jeu à interagir avec d'autres AWS ressources.

- Autorisez les processus du serveur de jeu à obtenir des informations sur la flotte sur laquelle ils s'exécutent.

Rubriques

- [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
- [Communiquez avec les autres AWS ressources de vos flottes](#)

Ajouter Amazon GameLift Servers à votre serveur de jeu

Cette rubrique explique comment modifier le code de votre serveur de jeu afin que les processus du serveur de jeu puissent communiquer avec le Amazon GameLift Servers service. Utilisez ces instructions pour les serveurs de jeu que vous prévoyez de déployer sur des EC2 flottes Amazon GameLift Servers gérées, des flottes de conteneurs gérées ou des flottes Anywhere.

Les processus du serveur de jeu communiquent avec le Amazon GameLift Servers service pour recevoir des instructions du service et pour signaler l'état des processus du serveur et l'état de la session de jeu. Pour obtenir des informations détaillées sur les interactions entre les composants de votre solution d'hébergement de jeux (serveur de jeu, service principal, client de jeu et Amazon GameLift Servers), consultez [Interactions entre le client et le serveur du jeu avec Amazon GameLift Servers](#).

Pour préparer votre jeu à l'hébergement, ajoutez le SDK du serveur Amazon GameLift Servers à votre projet de serveur de jeu. Si vous utilisez le Amazon GameLift Servers plugin pour Unreal Engine ou Unity, le SDK du serveur est intégré et prêt à être utilisé. Le SDK du serveur est disponible en plusieurs langues. Pour plus d'informations sur la prise en charge des outils pour les serveurs de jeux, y compris le SDK du serveur, consultez [Obtenez des outils Amazon GameLift Servers de développement](#).

Références de l'API du SDK du serveur :

- [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#)
- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)
- [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
- [SDK du serveur Go pour Amazon GameLift Servers -- Actions](#)

Initialiser le processus du serveur

Ajoutez du code pour établir la communication avec le Amazon GameLift Servers service et signalez lorsque le processus du serveur de jeu est prêt à héberger une session de jeu. Ce code doit être exécuté avant tout Amazon GameLift Servers code.

1. Initialisez un client d'Amazon GameLift Servers API en appelant `InitSdk()`. Si vous préparez votre serveur de jeu à fonctionner sur des EC2 flottes Amazon GameLift Servers gérées, utilisez la valeur par défaut `InitSDK()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) sans paramètres. Le client API gère la connexion au Amazon GameLift Servers service pour vous.

 Si vous préparez votre serveur de jeu pour une utilisation sur une flotte Amazon GameLift Servers Anywhere :

Initialisez le client Amazon GameLift Servers API en l'appelant `InitSdk()` avec ce qui suit : `ServerParameters`

- URL du websocket utilisé pour se connecter à votre serveur de jeu.
- ID du processus utilisé pour héberger votre serveur de jeu.
- L'ID de l'ordinateur hébergeant les processus de votre serveur de jeu.
- L'ID du parc contenant votre ordinateur Amazon GameLift Servers Anywhere.
- Le jeton d'autorisation généré par l'Amazon GameLift Servers opération [GetComputeAuthToken](#).

2. Avertissez le service que le processus du serveur de jeu est prêt à héberger une session de jeu. Appelez `ProcessReady()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) avec ce qui suit. `ProcessParameters` Chaque processus du serveur de jeu ne doit être appelé `ProcessReady()` qu'une seule fois.
 - Numéro de port du processus serveur. Lorsque le processus serveur démarre une session de jeu, il fournit le port au Amazon GameLift Servers service, qui met à jour les informations de session de jeu. Votre jeu peut récupérer ces informations et les fournir aux clients du jeu, qui les utilisent pour se connecter au processus du serveur et rejoindre la session de jeu.
 - Les emplacements des fichiers que vous Amazon GameLift Servers souhaitez stocker pour vous. Il peut s'agir de journaux de session de jeu et d'autres fichiers générés par le processus du serveur au cours d'une session de jeu. Bien que Amazon GameLift Servers ces fichiers soient enregistrés temporairement sur l'ordinateur sur lequel le processus serveur est en cours

d'exécution, ils ne sont disponibles que jusqu'à l'arrêt de l'instance. Vous pouvez accéder aux fichiers stockés via la [Amazon GameLift Serversconsole](#) ou en appelant l'opération Amazon GameLift Servers API [GetGameSessionLogUrl\(\)](#).

 Si vous préparez votre serveur de jeu pour une utilisation sur un parc de conteneurs Amazon GameLift Servers géré :

Il n'est pas nécessaire de spécifier les paramètres de journalisation d'une flotte de conteneurs. Envoyez plutôt les données de session de jeu et les autres données du journal vers la sortie standard. Les flottes de conteneurs capturent automatiquement toutes les sorties standard de conteneurs sous forme de flux journal.

- Les fonctions de rappel suivantes permettent d'Amazon GameLift Serversenvoyer des messages ou des invites à un processus de serveur de jeu. Vous devez implémenter chacune de ces fonctions dans le code de votre serveur de jeu. [Pour plus d'informations, voir ProcessParameters\(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#)
- (Facultatif) `onHealthCheck` — Amazon GameLift Servers appelle régulièrement cette fonction pour demander un rapport d'état de santé au serveur.
- `onStartGameSession`— Amazon GameLift Servers appelle cette fonction en réponse à la demande du client [CreateGameSession\(\)](#).
- `onProcessTerminate`— Amazon GameLift Servers force le processus du serveur à s'arrêter, le laissant s'arrêter correctement.
- (Facultatif) `onUpdateGameSession` : Amazon GameLift Servers fournit un objet de session de jeu mis à jour au serveur de jeu ou fournit une mise à jour de statut suite à une demande de remplacement de match. La fonction de [FlexMatchremblayage](#) nécessite ce rappel.

Vous pouvez également configurer un serveur de jeu afin qu'il puisse accéder en toute sécurité à d'autres AWS ressources que vous possédez ou contrôlez. Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

(Facultatif) Rapport sur l'état des processus du serveur

Ajoutez du code à votre serveur de jeu pour implémenter la fonction de rappel. `onHealthCheck()` Amazon GameLift Serversinvoque périodiquement cette méthode de rappel pour collecter des indicateurs de santé. Pour implémenter cette fonction de rappel, procédez comme suit :

- Évaluez l'état de santé du processus du serveur. Par exemple, vous pouvez signaler que le processus du serveur est défectueux si des dépendances externes ont échoué.
- Terminez l'évaluation du statut et répondez au rappel dans les 60 secondes. S'il Amazon GameLift Servers ne reçoit pas de réponse dans ce délai, il considère automatiquement que le processus du serveur est défectueux.
- Renvoie une valeur booléenne : true pour sain, false pour malsain.

Si vous n'implémentez pas de rappel de contrôle de santé, Amazon GameLift Servers considère que le processus du serveur est sain à moins que le serveur ne réponde pas.

Le Amazon GameLift Servers service utilise l'état des processus du serveur pour mettre fin aux processus défaillants et libérer les ressources. Si un processus serveur continue d'être signalé comme défaillant ou ne répond pas à trois vérifications de santé consécutives, le service peut arrêter le processus et en démarrer un nouveau. Le service collecte des mesures sur l'état des processus des serveurs d'une flotte.

(Facultatif) Obtenez un certificat TLS

Si le processus serveur est exécuté sur un parc sur lequel la génération de certificats TLS est activée, vous pouvez récupérer le certificat TLS pour établir une connexion sécurisée avec un client de jeu et chiffrer les communications client-serveur. Une copie du certificat est stockée sur l'instance. [Pour obtenir l'emplacement du fichier, appelez `GetComputeCertificate\(\)`\(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#)

Démarrer une session de jeu

Ajoutez du code pour implémenter la fonction de rappel. `onStartGameSession` Amazon GameLift Servers invoque ce rappel pour démarrer une session de jeu sur le processus du serveur.

La `onStartGameSession` fonction prend un [GameSession](#) objet comme paramètre d'entrée. Cet objet inclut des informations clés de session de jeu, telles que le nombre maximum de joueurs. Il peut également inclure des données de jeu et des données de joueur. L'implémentation de la fonction doit effectuer les tâches suivantes :

- Lancez des actions pour créer une nouvelle session de jeu en fonction des `GameSession` propriétés. Au minimum, le serveur de jeu doit associer l'identifiant de session de jeu, auquel les clients du jeu font référence lorsqu'ils se connectent au processus serveur.
- Traitez les données du jeu et les données des joueurs selon vos besoins. Ces données se trouvent dans l'`GameSession` objet.

- Prévenez le Amazon GameLift Servers service lorsqu'une nouvelle session de jeu est prête à accepter des joueurs. [Appelez l'opération API du serveur `ActivateGameSession\(\)`\(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#) En réponse à un appel réussi, le service change le statut de la session de jeu en `ACTIVE`.

(Facultatif) Valider un nouveau joueur

Si vous suivez le statut des sessions des joueurs, ajoutez un code pour valider un nouveau joueur lorsqu'il se connecte à un serveur de jeu. Amazon GameLift Servers suit les joueurs actuels et les créneaux de session de jeu disponibles.

À des fins de validation, un client de jeu qui tente de rejoindre une session de jeu doit inclure un identifiant de session de joueur. Amazon GameLift Servers génère cet identifiant lorsque votre jeu démarre de nouvelles sessions de jeu en appelant [StartGameSessionPlacement\(\)](#) ou [StartMatchmaking\(\)](#). Sur ces demandes, un créneau libre dans une session de jeu est réservé à la session du joueur.

Lorsque le processus du serveur de jeu reçoit une demande de connexion d'un client de jeu, il appelle `AcceptPlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) avec l'identifiant de session du joueur. En réponse, Amazon GameLift Servers vérifie que l'identifiant de session du joueur correspond à un emplacement libre réservé pendant la session de jeu. Après avoir Amazon GameLift Servers validé l'identifiant de session du joueur, le processus serveur accepte la connexion. Le joueur peut ensuite rejoindre la session de jeu. Si l'identifiant de session du joueur Amazon GameLift Servers n'est pas validé, le processus du serveur refuse la connexion.

(Facultatif) Signaler la fin d'une session de joueur

Si vous suivez le statut des sessions des joueurs, ajoutez un code pour avertir Amazon GameLift Servers lorsqu'un joueur quitte la session de jeu. Ce code doit être exécuté chaque fois que le processus serveur détecte une interruption de connexion. Amazon GameLift Servers utilise cette notification pour suivre les joueurs actuels et les emplacements disponibles pendant la session de jeu.

Pour gérer les connexions supprimées dans votre code, ajoutez un appel à l'opération d'API du serveur `RemovePlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) avec l'ID de session de joueur correspondant.

Mettre fin à une session de jeu

Ajoutez un code à la séquence d'arrêt du processus du serveur pour Amazon GameLift Servers avertir de la fin d'une session de jeu. Pour recycler et actualiser les ressources d'hébergement, arrêtez chaque processus du serveur une fois la session de jeu terminée.

[Au début du code d'arrêt du processus serveur, appelez l'opération API du serveur](#)

[ProcessEnding\(\) \(C++\) \(C#\) \(Unreal\) \(Go\) g \(C++\)](#) Cet appel indique Amazon GameLift Servers que le processus du serveur est en train de s'arrêter. Amazon GameLift Servers change l'état de la session de jeu et le statut du processus du serveur en `TERMINATED`. Après l'appel `ProcessEnding()`, le processus peut être arrêté en toute sécurité.

Répondre à une notification d'arrêt du processus du serveur

Ajoutez du code pour arrêter le processus du serveur en réponse à une notification du Amazon GameLift Servers service. Le service envoie cette notification lorsque le processus serveur signale régulièrement un dysfonctionnement ou si l'instance sur laquelle le processus serveur est en cours d'exécution est arrêtée. Amazon GameLift Servers peut arrêter une instance dans le cadre d'un événement de réduction de capacité ou en réponse à une interruption d'une instance Spot. Une interruption d'une instance Spot donne un préavis de deux minutes, ce qui donne au serveur le temps de déconnecter les joueurs en douceur, de préserver les données relatives à l'état du jeu et d'effectuer d'autres tâches de nettoyage.

Pour gérer une notification d'arrêt, apportez les modifications suivantes au code de votre serveur de jeu :

- [Implémentez la fonction de rappel `onProcessTerminate\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#) Cette fonction doit appeler le code qui arrête le processus du serveur.
- Appelez l'opération API du serveur `GetTerminationTime()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) ([C++](#)) à partir du code d'arrêt de votre serveur de jeu. S'il Amazon GameLift Servers a émis un appel pour arrêter le processus du serveur, `GetTerminationTime()` renvoie l'heure de fin estimée.
- [Au début du code d'arrêt de votre serveur de jeu, appelez l'opération API du serveur `ProcessEnding\(\)` \(C++\) \(C#\) \(Unreal\) \(Go\) \(C++\)](#) Cet appel informe le service Amazon GameLift Servers que le processus serveur est en cours d'arrêt. Le service change ensuite l'état du processus du serveur en `TERMINATED`. Après l'appel `ProcessEnding()`, le processus peut être arrêté en toute sécurité.

Communiquez avec les autres AWS ressources de vos flottes

Lorsque vous créez une version de serveur de jeu destinée à être déployée sur Amazon GameLift Servers des flottes, vous souhaitez peut-être que les applications de votre version de jeu communiquent directement et en toute sécurité avec les autres AWS ressources que vous possédez. Étant donné Amazon GameLift Servers que vous gérez vos flottes d'hébergement de jeux, vous devez accorder un accès Amazon GameLift Servers limité à ces ressources et services.

Voici quelques exemples de scénarios :

- Utilisez un CloudWatch agent Amazon pour collecter des métriques, des journaux et des traces à partir de EC2 flottes gérées et de flottes Anywhere.
- Envoyez les données du journal de l'instance à Amazon CloudWatch Logs.
- Obtenez des fichiers de jeu stockés dans un bucket Amazon Simple Storage Service (Amazon S3).
- Lisez et écrivez des données de jeu (telles que les modes de jeu ou l'inventaire) stockées dans une base de données Amazon DynamoDB ou dans un autre service de stockage de données.
- Envoyez des signaux directement à une instance à l'aide d'Amazon Simple Queue Service (Amazon SQS).
- Accédez à des ressources personnalisées déployées et exécutées sur Amazon Elastic Compute Cloud (Amazon EC2).

Amazon GameLift Servers prend en charge les méthodes suivantes pour établir l'accès :

- [Accédez aux AWS ressources avec un rôle IAM](#)
- [Accédez aux AWS ressources grâce au peering VPC](#)

Accédez aux AWS ressources avec un rôle IAM

Utilisez un rôle IAM pour spécifier qui peut accéder à vos ressources et définir des limites à cet accès. Les parties de confiance peuvent « assumer » un rôle et obtenir des informations de sécurité temporaires les autorisant à interagir avec les ressources. Lorsque les parties font des demandes d'API liées à la ressource, elles doivent inclure les informations d'identification.

Pour configurer le contrôle d'accès par un rôle IAM, effectuez les tâches suivantes :

1. [Création du rôle IAM](#)
2. [Modifier les applications pour obtenir des informations d'identification](#)

3. [Associer une flotte au rôle IAM](#)

Création du rôle IAM

Au cours de cette étape, vous créez un rôle IAM, avec un ensemble d'autorisations pour contrôler l'accès à vos AWS ressources et une politique de confiance qui donne le Amazon GameLift Servers droit d'utiliser les autorisations du rôle.

Pour obtenir des instructions sur la façon de configurer le rôle IAM, consultez [Configurer un rôle de service IAM pour Amazon GameLift Servers](#). Lors de la création de la politique d'autorisations, choisissez les services, les ressources et les actions spécifiques que vos applications doivent utiliser. Il est recommandé de limiter autant que possible l'étendue des autorisations.

Après avoir créé le rôle, prenez note de son Amazon Resource Name (ARN). Vous avez besoin du rôle ARN lors de la création de la flotte.

Modifier les applications pour obtenir des informations d'identification

Au cours de cette étape, vous configurez vos applications pour obtenir des informations d'identification de sécurité pour le rôle IAM et les utiliser lors de l'interaction avec vos AWS ressources. Consultez le tableau suivant pour déterminer comment modifier vos applications en fonction (1) du type d'application et (2) de la version du SDK du serveur avec Amazon GameLift Servers laquelle votre jeu communique.

	Applications pour serveurs de jeux	Autres applications
Utilisation du SDK du serveur version 5.x	Appelez la méthode SDK du serveur <code>GetFleetRoleCredentials()</code> à partir du code de votre serveur de jeu.	Ajoutez du code à l'application pour extraire les informations d'identification d'un fichier partagé sur l'instance de flotte.
Utilisation du SDK du serveur version 4 ou antérieure	Appelez AWS Security Token Service (AWS STS) AssumeRole avec le rôle ARN.	Appelez AWS Security Token Service (AWS STS) AssumeRole avec le rôle ARN.

Pour les jeux intégrés au SDK 5.x du serveur, ce schéma montre comment les applications de votre version de jeu déployée peuvent acquérir des informations d'identification pour le rôle IAM.

Appel **GetFleetRoleCredentials()** (SDK 5.x du serveur)

Dans le code de votre serveur de jeu, qui devrait déjà être intégré au SDK 5.x Amazon GameLift Servers du serveur, appelez `GetFleetRoleCredentials` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) pour récupérer un ensemble d'informations d'identification temporaires. Lorsque les informations d'identification expirent, vous pouvez les actualiser avec un autre appel à `GetFleetRoleCredentials`.

Utiliser des informations d'identification partagées (SDK du serveur 5.x)

Pour les applications non serveur déployées avec des versions de serveurs de jeu à l'aide du SDK 5.x du serveur, ajoutez du code pour obtenir et utiliser les informations d'identification stockées dans un fichier partagé. Amazon GameLift Servers génère un profil d'informations d'identification pour chaque instance de flotte. Les informations d'identification peuvent être utilisées par toutes les applications de l'instance. Amazon GameLift Servers actualise en permanence les informations d'identification temporaires.

Vous devez configurer une flotte pour générer le fichier d'informations d'identification partagé lors de la création de la flotte.

Dans chaque application qui doit utiliser le fichier d'informations d'identification partagé, spécifiez l'emplacement du fichier et le nom du profil, comme suit :

Windows :

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "C:\\Credentials\\credentials"
```

Linux :

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "/local/credentials/credentials"
```

Exemple : configurer un CloudWatch agent pour collecter des métriques pour les instances Amazon GameLift Servers de flotte

Si vous souhaitez utiliser un CloudWatch agent Amazon pour collecter des métriques, des journaux et des traces à partir de vos Amazon GameLift Servers flottes, utilisez cette méthode pour autoriser l'agent à émettre les données vers votre compte. Dans ce scénario, suivez les étapes suivantes :

1. Récupérez ou écrivez le `config.json` fichier de CloudWatch l'agent.
2. Mettez à jour le `common-config.toml` fichier pour que l'agent identifie le nom du fichier d'informations d'identification et le nom du profil, comme décrit ci-dessus.
3. Configurez le script d'installation de build de votre serveur de jeu pour installer et démarrer l'CloudWatch agent.

Utilisation **AssumeRole()** (SDK 4 du serveur)

Ajoutez du code à vos applications pour assumer le rôle IAM et obtenir des informations d'identification pour interagir avec vos AWS ressources. Toute application qui s'exécute sur une instance de Amazon GameLift Servers flotte avec le SDK de serveur 4 ou une version antérieure peut assumer le rôle IAM.

Dans le code de l'application, avant d'accéder à une AWS ressource, l'application doit appeler l'opération [AssumeRole](#) API AWS Security Token Service (AWS STS) et spécifier l'ARN du rôle. Cette opération renvoie un ensemble d'informations d'identification temporaires qui autorisent l'application à accéder à la AWS ressource. Pour plus d'informations, consultez la section [Utilisation d'informations d'identification temporaires avec AWS des ressources](#) dans le Guide de l'utilisateur IAM.

Associer une flotte au rôle IAM

Après avoir créé le rôle IAM et mis à jour les applications de la version de votre serveur de jeu pour obtenir et utiliser les informations d'accès, vous pouvez déployer une flotte. Lorsque vous configurez la nouvelle flotte, définissez les paramètres suivants :

- [InstanceRoleArn](#)— Définissez ce paramètre sur l'ARN du rôle IAM.
- [InstanceRoleCredentialsProvider](#)— Pour demander Amazon GameLift Servers à générer un fichier d'informations d'identification partagé pour chaque instance de flotte, définissez ce paramètre sur `SHARED_CREDENTIAL_FILE`.

Vous devez définir ces valeurs lors de la création de la flotte. Ils ne pourront pas être mis à jour ultérieurement.

Accédez aux AWS ressources grâce au peering VPC

Vous pouvez utiliser le peering Amazon Virtual Private Cloud (Amazon VPC) pour communiquer entre les applications exécutées sur une Amazon GameLift Servers instance et une autre ressource. AWS Un VPC est un réseau privé virtuel que vous définissez et qui inclut un ensemble de ressources gérées via votre. Compte AWS Chaque flotte Amazon GameLift Servers est dotée de son propre VPC. Avec le peering VPC, vous pouvez établir une connexion réseau directe entre le VPC de votre flotte et celui de vos autres ressources. AWS

Amazon GameLift Servers rationalise le processus de configuration des connexions d'appairage de VPC pour vos serveurs de jeu. Il gère les demandes d'appairage, met à jour les tables de routage et configure les connexions requises. Pour obtenir des instructions sur la façon de configurer le peering VPC pour vos serveurs de jeu, consultez. [Peering VPC pour Amazon GameLift Servers](#)

Intégrez votre client de jeu à Amazon GameLift Servers

Les rubriques de cette section décrivent les Amazon GameLift Servers fonctionnalité que vous pouvez ajouter à un service principal. Un service principal gère les tâches suivantes :

- Demande des informations sur les sessions de jeu actives à Amazon GameLift Servers.
- Connecte un joueur à une session de jeu existante.
- Crée une nouvelle session de jeu et y associe les joueurs.
- Modifie les métadonnées d'une session de jeu existante.

Pour plus d'informations sur la manière dont les clients du jeu interagissent avec Amazon GameLift Servers et des serveurs de jeux fonctionnant sur Amazon GameLift Servers, voir [Interactions entre le client et le serveur du jeu avec Amazon GameLift Servers](#).

Prérequis

- Un Compte AWS.
- Une version de serveur de jeu téléchargée sur Amazon GameLift Servers.
- Une flotte pour héberger vos jeux.

Rubriques

- [Ajoutez Amazon GameLift Servers à votre client de jeu](#)

- [Générer un joueur IDs](#)

Ajoutez Amazon GameLift Servers à votre client de jeu

Amazon GameLift ServersIntégrez-les aux composants de jeu qui nécessitent des informations sur les sessions de jeu, créez de nouvelles sessions de jeu et ajoutez des joueurs aux parties. Selon l'architecture de votre jeu, cette fonctionnalité se trouve dans les services principaux qui gèrent des tâches telles que l'authentification des joueurs, le matchmaking ou le placement des sessions de jeu.

Note

Pour des informations détaillées sur la façon de configurer le matchmaking pour votre jeu, consultez le [guide du Amazon GameLift ServersFlexMatch développeur](#).

Configuration Amazon GameLift Servers sur un service principal

Ajoutez du code pour initialiser un Amazon GameLift Servers client et enregistrer les paramètres clés. Ce code doit être exécuté avant tout code dépendant deAmazon GameLift Servers.

1. Configurez une configuration client. Utilisez la configuration client par défaut ou créez un objet de configuration client personnalisé. Pour plus d'informations, consultez [AWS::Client::ClientConfiguration](#)(C++) ou [AmazonGameLiftConfig](#)(C#).

Une configuration client spécifie une région cible et un point de terminaison à utiliser lors de la prise de contactAmazon GameLift Servers. La région identifie l'ensemble des ressources déployées (flottes, files d'attente et entremetteurs) à utiliser. La configuration du client par défaut définit l'emplacement sur la région USA Est (Virginie du Nord). Pour utiliser une autre région, créez une configuration personnalisée.

2. Initialisez un client Amazon GameLift Servers. Utilisez [Aws : GameLift : : GameLiftClient \(\)](#) (C++) ou [AmazonGameLiftClient\(\)](#) (C#) avec une configuration client par défaut ou une configuration client personnalisée.
3. Ajoutez un mécanisme permettant de générer un identifiant unique pour chaque joueur. Pour de plus amples informations, veuillez consulter [Générer un joueur IDs](#).
4. Collectez et stockez les informations suivantes :
 - Parc cible : de nombreuses demandes d'Amazon GameLift ServersAPI doivent spécifier un parc. Pour ce faire, utilisez un identifiant de flotte ou un identifiant d'alias pointant vers le parc

cible. Il est recommandé d'utiliser des alias de flotte afin de pouvoir changer de joueur d'une flotte à l'autre sans avoir à mettre à jour vos services principaux.

- File d'attente cible : pour les jeux qui utilisent des files d'attente impliquant plusieurs flottes pour créer de nouvelles sessions de jeu, spécifiez le nom de la file d'attente à utiliser.
- AWS informations d'identification — Tous les appels vers Amazon GameLift Servers doivent fournir les informations d'identification de l'hôte du jeu. Compte AWS Vous obtenez ces informations d'identification en créant un utilisateur joueur, comme décrit dans [Configurez l'accès programmatique à votre jeu](#). En fonction de la façon dont vous gérez l'accès de l'utilisateur du joueur, procédez comme suit :
 - Si vous utilisez un rôle pour gérer les autorisations des utilisateurs des joueurs, ajoutez du code pour assumer le rôle avant d'appeler une Amazon GameLift Servers API. La demande d'attribution du rôle renvoie un ensemble d'informations d'identification de sécurité temporaires. Pour plus d'informations, consultez la section [Passage à un rôle IAM \(AWS API\)](#) dans le guide de l'utilisateur IAM.
 - Si vous disposez d'informations d'identification de sécurité à long terme, configurez votre code pour localiser et utiliser les informations d'identification stockées. Voir [Authentifier à l'aide d'informations d'identification à long terme](#) dans le guide de référence AWS SDKs et Tools. Pour plus d'informations sur le stockage des informations d'identification, consultez les références d'AWS API pour [\(C++\)](#) et [\(.NET\)](#).
 - Si vous disposez d'informations d'identification de sécurité temporaires, ajoutez du code pour actualiser régulièrement les informations d'identification à l'aide du AWS Security Token Service (AWS STS), comme décrit dans la section [Utilisation d'informations d'identification de sécurité temporaires avec le AWS SDKs](#) guide de l'utilisateur IAM. Le code doit demander de nouvelles informations d'identification avant que les anciennes n'expirent.

Obtenez des sessions de jeu

Ajoutez du code pour découvrir les sessions de jeu disponibles et gérer les paramètres et les métadonnées des sessions de jeu.

Rechercher des sessions de jeu actives

[SearchGameSessions](#) Utilisez-le pour obtenir des informations sur une session de jeu spécifique, toutes les sessions actives ou les sessions répondant à un ensemble de critères de recherche. Cet appel renvoie un [GameSession](#) objet pour chaque session de jeu active correspondant à votre demande de recherche.

Utilisez des critères de recherche pour obtenir une liste filtrée des sessions de jeu actives auxquelles les joueurs peuvent se connecter. Par exemple, vous pouvez filtrer les sessions de la façon suivante :

- Exclure les sessions de jeu complètes : `CurrentPlayerSessionCount = MaximumPlayerSessionCount`.
- Choisissez les sessions de jeu en fonction de la durée pendant laquelle la session est en cours : `ÉvaluezCreationTime`.
- Trouvez des sessions de jeu en fonction d'une propriété de jeu personnalisée : `gameSessionProperties.gameMode = "brawl"`.

Gérer les sessions de jeu

Utilisez les opérations suivantes pour récupérer ou mettre à jour des informations sur des sessions de jeu.

- [DescribeGameSessionDetails\(\)](#) — Obtenez le statut de protection d'une session de jeu en plus des informations de session de jeu.
- [UpdateGameSession\(\)](#) — Modifiez les métadonnées et les paramètres d'une session de jeu selon vos besoins.
- [GetGameSessionLogUrl](#) — Accédez aux journaux de session de jeu stockés.

Créez des sessions de jeu

Ajoutez du code pour démarrer de nouvelles sessions de jeu sur vos flottes déployées et les mettre à disposition des joueurs. Il existe deux options pour créer des sessions de jeu, selon que vous déployez votre jeu dans plusieurs régions AWS ou dans une seule.

Créez une session de jeu dans une file d'attente multisite

[StartGameSessionPlacement](#) À utiliser pour placer une demande de nouvelle session de jeu dans une file d'attente. Pour utiliser cette opération, créez une file d'attente. Cela détermine où se Amazon GameLift Servers situe la nouvelle session de jeu. Pour plus d'informations sur les files d'attente et sur leur utilisation, consultez [Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente](#).

Lorsque vous créez un emplacement de session de jeu, spécifiez le nom de la file d'attente à utiliser, un nom de session de jeu, un nombre maximum de joueurs simultanés et un ensemble facultatif de propriétés de jeu. Vous pouvez également éventuellement fournir une liste de joueurs qui rejoindront

automatiquement la session de jeu. Si vous incluez des données de latence des joueurs pour les régions concernées, utilisez Amazon GameLift Servers ces informations pour placer la nouvelle session de jeu dans une flotte offrant une expérience de jeu idéale aux joueurs.

Pour obtenir des mesures de latence précises, utilisez Amazon GameLift Servers les balises ping UDP. Ces points de terminaison vous permettent de mesurer la latence réelle du réseau UDP entre les appareils des joueurs et les sites d'hébergement potentiels, ce qui permet de prendre des décisions de placement plus précises qu'avec des pings ICMP. Pour plus d'informations sur l'utilisation de balises ping UDP pour mesurer la latence, reportez-vous à [Balises ping UDP](#)

Le placement de session de jeu est un processus asynchrone. Une fois que vous avez fait une demande, vous pouvez la laisser aboutir ou expirer. Vous pouvez également annuler la demande à tout moment en utilisant [StopGameSessionPlacement](#). Pour vérifier le statut de votre demande de placement, appelez [DescribeGameSessionPlacement](#).

Créez une session de jeu sur une flotte spécifique

[CreateGameSession](#) À utiliser pour créer une nouvelle session sur un parc spécifique. Cette opération synchrone réussit ou échoue selon que la flotte dispose de ressources disponibles pour héberger une nouvelle session de jeu. Après avoir Amazon GameLift Servers créé la nouvelle session de jeu et renvoyé un [GameSession](#) objet, vous pouvez y rejoindre des joueurs.

Lorsque vous utilisez cette opération, fournissez un identifiant de flotte ou un identifiant d'alias, un nom de session et le nombre maximum de joueurs simultanés pour le jeu. Vous pouvez également inclure un ensemble de propriétés de jeu. Les propriétés du jeu sont définies dans un tableau de paires clé-valeur.

Si vous utilisez la fonction de protection des Amazon GameLift Servers ressources pour limiter le nombre de sessions de jeu qu'un joueur peut créer, fournissez l'identifiant de joueur du créateur de la session de jeu.

Joindre un joueur à une session de jeu

Ajoutez un code pour réserver une place de joueur dans une session de jeu active et connecter les clients du jeu aux sessions de jeu.

1. Réservez une place de joueur lors d'une session de jeu

Pour réserver un emplacement de joueur, créez une session de joueur pour la session de jeu. Pour plus d'informations sur les sessions des joueurs, consultez [Comment les joueurs se connectent aux jeux](#).

Il existe deux manières de créer de nouvelles sessions pour les joueurs :

- [StartGameSessionPlacement](#) À utiliser pour réserver des places à un ou plusieurs joueurs lors de la nouvelle session de jeu.
- Réservez des places pour un ou plusieurs joueurs en utilisant [CreatePlayerSession](#) ou [CreatePlayerSessions](#) avec un identifiant de session de jeu.

Amazon GameLift Servers vérifie d'abord que la session de jeu accepte de nouveaux joueurs et qu'il y a des emplacements disponibles pour les joueurs. En cas de succès, Amazon GameLift Servers réserve une place au joueur, crée une nouvelle session de joueur et renvoie un [PlayerSession](#) objet. Cet objet contient le nom DNS, l'adresse IP et le port dont un client de jeu a besoin pour se connecter à la session de jeu.

Une demande de session de joueur doit inclure un ID unique pour chaque joueur. Pour de plus amples informations, veuillez consulter [Générer un joueur IDs](#).

Une session de joueur peut inclure un ensemble de données de joueur personnalisées. Ces données sont stockées dans le nouvel objet de session du joueur, que vous pouvez récupérer en appelant [DescribePlayerSessions\(\)](#). Amazon GameLift Server transmet également cet objet au serveur de jeu lorsque le joueur se connecte directement à la session de jeu. Lorsque vous demandez des sessions à plusieurs joueurs, fournissez une chaîne de données de joueur pour chaque joueur mappée à l'identifiant du joueur indiqué dans la demande.

2. Se connecter à une session de jeu

Ajoutez du code au client de jeu pour récupérer l'objet `PlayerSession`, qui contient les informations de connexion de la session de jeu. Utilisez ces informations pour établir une connexion directe avec le serveur.

- Vous pouvez vous connecter à l'aide du port spécifié et du nom DNS ou de l'adresse IP attribués au processus du serveur.
- Si la génération de certificats TLS est activée dans vos flottes, connectez-vous à l'aide du nom et du port DNS.
- Si votre serveur de jeu valide les connexions entrantes entre joueurs, faites référence à l'identifiant de session du joueur.

Une fois la connexion établie, le client du jeu et le processus serveur communiquent directement sans intervention Amazon GameLift Servers. Le serveur maintient la communication avec Amazon GameLift Servers le joueur pour signaler l'état de connexion, l'état de santé, etc. Si le serveur de jeu valide les joueurs entrants, il vérifie que l'identifiant de session du joueur correspond à un emplacement réservé dans la session de jeu et accepte ou refuse la connexion du joueur. Lorsque le joueur se déconnecte, le processus serveur signale l'interruption de la connexion.

Utiliser les propriétés des sessions de jeu

Votre client de jeu peut transmettre des données à une session de jeu en utilisant une propriété de jeu. Les propriétés du jeu sont des paires clé-valeur que votre serveur de jeu peut ajouter, lire, répertorier et modifier. Vous pouvez transmettre une propriété de jeu lorsque vous créez une nouvelle session de jeu, ou plus tard lorsque la session de jeu est active. Une session de jeu peut contenir jusqu'à 16 propriétés de jeu. Vous ne pouvez pas supprimer les propriétés du jeu.

Par exemple, votre jeu propose les niveaux de difficulté suivants : `NoviceEasy`, `Intermediate`, et `Expert`. Un joueur choisit `Easy`, puis commence la partie. Votre client de jeu demande une nouvelle session de jeu Amazon GameLift Servers en utilisant l'un `StartGameSessionPlacement` ou l'autre ou `CreateGameSession` comme expliqué dans les sections précédentes. Dans la demande, le client transmet ceci : `{"Key": "Difficulty", "Value": "Easy"}`.

En réponse à la demande, Amazon GameLift Servers crée un `GameSession` objet contenant la propriété de jeu spécifiée. Amazon GameLift Servers demande ensuite à un serveur de jeu disponible de démarrer la nouvelle session de jeu et transmet l'`GameSession` objet. Le serveur de jeu démarre une session de jeu avec un `Difficulty` de `Easy`.

En savoir plus

- [GameProperty type de données](#)
- [SearchGameSessions\(\) exemples](#)
- [UpdateGameSession GameProperties paramètre \(\)](#)

Générer un joueur IDs

Amazon GameLift Servers utilise une session de joueur pour représenter un joueur connecté à une session de jeu. Amazon GameLift Servers crée une session de joueur chaque fois qu'un joueur

se connecte à une session de jeu à l'aide d'un client de jeu intégré à Amazon GameLift Servers. Lorsqu'un joueur quitte une partie, la session du joueur se termine. Amazon GameLift Servers ne réutilise pas les sessions des joueurs.

L'exemple de code suivant génère aléatoirement un joueur unique IDs :

```
bool includeBrackets = false;
bool includeDashes = true;
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
    includeDashes);
```

Pour plus d'informations sur les sessions des joueurs, consultez [Sessions de jeu et de jeu dans la Amazon GameLift Servers console](#).

Moteurs de jeu et Amazon GameLift Servers

Vous pouvez utiliser le gestionnaire Amazon GameLift Servers service avec la plupart des principaux moteurs de jeu compatibles avec les bibliothèques C++ ou C#, notamment O3DE, Unreal Engine et Unity. Générez la version dont vous avez besoin pour votre jeu. Pour en savoir plus sur les instructions de génération et les exigences minimales, consultez les fichiers README de chaque version. Pour plus d'informations sur les produits disponibles Amazon GameLift Servers SDKs, plateformes de développement et systèmes d'exploitation pris en charge, voir [Obtenez des outils Amazon GameLift Servers de développement](#) pour les serveurs de jeux.

Outre les informations spécifiques au moteur fournies dans cette rubrique, trouvez de l'aide supplémentaire sur l'intégration Amazon GameLift Servers dans vos serveurs, clients et services de jeu dans les rubriques suivantes :

- [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)— Instructions détaillées sur l'intégration Amazon GameLift Servers dans un serveur de jeu.
- [Ajoutez Amazon GameLift Servers à votre client de jeu](#) : instructions détaillées sur l'intégration à un client de jeu ou un service, y compris la création de sessions de jeu et la participation de joueurs aux jeux.

O3DE

Serveurs de jeux

Préparez vos serveurs de jeu pour l'hébergement sur Amazon GameLift Servers à l'aide du SDK [du serveur pour Amazon GameLift Servers pour C++](#). Consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#) pour obtenir de l'aide sur l'intégration des fonctionnalités requises à votre serveur de jeux.

Clients et services de jeux

Permettez à vos clients de jeu et/ou à vos services de jeu d'interagir avec Amazon GameLift Servers service, par exemple pour rechercher des sessions de jeu disponibles ou en créer de nouvelles, et ajouter des joueurs aux parties. Les fonctionnalités principales du client sont fournies dans le [AWS SDK for C++](#). Pour intégrer Amazon GameLift Servers dans votre projet de jeu O3DE, voir [Addition Amazon GameLift Servers vers un client et un serveur de jeu O3DE](#) et [Ajoutez Amazon GameLift Servers à votre client de jeu](#)

Unreal Engine

Serveurs de jeux

Préparez vos serveurs de jeu pour l'hébergement sur Amazon GameLift Servers en ajoutant le SDK [du serveur pour Amazon GameLift Servers pour Unreal Engine](#) dans votre projet et en implémentant les fonctionnalités de serveur requises. Pour vous aider à configurer le plugin Unreal Engine et à ajouter Amazon GameLift Servers code, voir [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#).

Clients et services de jeux

Permettez à vos clients de jeu et/ou à vos services de jeu d'interagir avec Amazon GameLift Servers service, par exemple pour rechercher des sessions de jeu disponibles ou en créer de nouvelles, et ajouter des joueurs aux parties. Les fonctionnalités principales du client sont fournies dans le [AWS SDK for C++](#). Pour intégrer Amazon GameLift Servers dans votre projet de jeu Unreal Engine, voir [Ajoutez Amazon GameLift Servers à votre client de jeu](#).

Unity

Serveurs de jeux

Préparez vos serveurs de jeu pour l'hébergement sur Amazon GameLift Servers en ajoutant le SDK [du serveur pour Amazon GameLift Servers pour C#](#) dans votre projet et en implémentant les fonctionnalités de serveur requises. Pour obtenir de l'aide lors de la configuration avec Unity et de l'ajout Amazon GameLift Servers code, voir [Intégrer Amazon GameLift Servers dans un projet Unity](#).

Clients et services de jeux

Permettez à vos clients de jeu et/ou à vos services de jeu d'interagir avec Amazon GameLift Servers service, par exemple pour rechercher des sessions de jeu disponibles ou en créer de nouvelles, et ajouter des joueurs aux parties. Les principales fonctionnalités client sont fournies dans le kit [AWS SDK pour .NET](#). Pour intégrer Amazon GameLift Servers dans votre projet de jeu Unity, voir [Ajoutez Amazon GameLift Servers à votre client de jeu](#).

Autres moteurs

Pour une liste complète des Amazon GameLift Servers SDKs disponible pour les serveurs et les clients de jeu, voir [the section called “Obtenez des outils de développement”](#).

Addition Amazon GameLift Servers vers un client et un serveur de jeu O3DE

Vous pouvez utiliser O3DE, un moteur 3D open source, multiplateforme et en temps réel, pour créer des expériences interactives hautes performances, notamment des jeux et des simulations. Le moteur de rendu et les outils O3DE sont intégrés dans un cadre modulaire que vous pouvez modifier et étendre avec vos outils de développement préférés.

Le framework modulaire utilise des Gems qui contiennent des bibliothèques avec des interfaces et des actifs standard. Sélectionnez vos propres gemmes pour choisir les fonctionnalités à ajouter en fonction de vos besoins.

Le Amazon GameLift Servers Gem fournit les fonctionnalités suivantes :

Amazon GameLift Servers intégration

Un framework pour étendre la couche réseau O3DE et permettre à la gemme multijoueur de fonctionner avec Amazon GameLift Servers solution de serveur dédié. The Gem fournit des intégrations à la fois avec le SDK [du serveur pour Amazon GameLift Servers](#) et le client AWS SDK (pour appeler le Amazon GameLift Servers service lui-même).

Gestion des compilations et des packages

Instructions pour emballer et éventuellement télécharger la version du serveur dédié et une application AWS Cloud Development Kit (AWS CDK) (AWS CDK) pour configurer et mettre à jour les ressources.

Amazon GameLift Servers Configuration de la gemme

Suivez les procédures décrites dans cette section pour configurer le Amazon GameLift Servers Gem dans O3DE.

Prérequis

- Configurez votre AWS compte pour Amazon GameLift Servers. Pour plus d'informations, consultez [Configurez un Compte AWS](#).
- Configurez les AWS informations d'identification pour O3DE. Pour plus d'informations, voir [Configuration des AWS informations d'identification](#).
- Configurez le AWS CLI terrain AWS CDK. Pour plus d'informations, [AWS Command Line Interface](#) et [AWS Cloud Development Kit \(AWS CDK\)](#).

Activez le Amazon GameLift Servers Gem et ses dépendances

1. Ouvrez le gestionnaire de projet.
2. Ouvrez le menu situé sous votre projet et choisissez Modifier les paramètres du projet... .
3. Choisissez Configurer les gemmes.
4. Activez le Amazon GameLift Servers Gemme et les gemmes dépendantes suivantes :
 - [AWS Core Gem](#) — Fournit le cadre à utiliser Services AWS dans O3DE.
 - [Gemme multijoueur](#) — Fournit des fonctionnalités multijoueurs en étendant le cadre réseau.

Incluez le Amazon GameLift Servers bibliothèque statique Gem

1. Incluez l'`Gem::AWSGameLift.Server.Static` annonce `BUILD_DEPENDENCIES` pour la cible de votre serveur de projet.

```
ly_add_target(  
    NAME YourProject.Server.Static STATIC  
    ...  
    BUILD DEPENDENCIES  
        PUBLIC  
        ...  
        PRIVATE  
        ...  
        Gem::AWSGameLift.Server.Static
```

```
)
```

2. Définissez `AWSGameLiftService` comme obligatoire pour le composant système de votre serveur de projet.

```
void
YourProjectServerSystemComponent::GetRequiredServices(AZ::ComponentDescriptor::DependencyArray&
required)
{
    ...
    required.push_back(AZ_CRC_CE("AWSGameLiftServerService"));
    ...
}
```

3. (Facultatif) À créer Amazon GameLift Servers demandes de service en C++, incluez-les `Gem::AWSGameLift.Client.Static` dans celles `BUILD_DEPENDENCIES` destinées à votre client cible.

```
ly_add_target(
    NAME YourProject.Client.Static STATIC
    ...
    BUILD_DEPENDENCIES
    PUBLIC
    ...
    PRIVATE
    ...
    Gem::AWSCore.Static
    Gem::AWSGameLift.Client.Static
}
```

Intégrez votre jeu et votre serveur dédié

Gérez les sessions de jeu au sein de votre jeu et de votre serveur de jeu dédié grâce à l'[intégration de la gestion des sessions](#). Pour soutenir FlexMatch, voir [FlexMatch Intégration](#).

Intégrer Amazon GameLift Servers dans un projet Unreal Engine

Découvrez comment intégrer le Amazon GameLift Servers SDK pour Unreal Engine dans vos projets de jeu afin d'accéder à l'ensemble des fonctionnalités du SDK pour serveurs.

 Conseil

Pour un déploiement plus rapide, essayez le plugin Amazon GameLift Servers autonome pour Unreal Engine. Il fournit des flux de travail d'interface utilisateur guidés pour déployer rapidement votre serveur de jeu avec une configuration minimale, afin que vous puissiez tester les composants de votre jeu en action. Consultez [Amazon GameLift Serversplugin pour Unreal Engine](#).

Ressources supplémentaires :

- [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
- [Obtenez des outils Amazon GameLift Servers de développement](#)

Installez le SDK du serveur pour Unreal

Téléchargez le Amazon GameLift Servers SDK open source pour Unreal Engine sur [GitHub](#). Les fichiers readme du référentiel contiennent les prérequis et les instructions d'installation.

Configurer les cibles de construction et les règles du module

Modifiez les fichiers de votre projet de jeu pour générer correctement les composants de build à utiliser avec Amazon GameLift Servers.

Pour ajouter des cibles de build pour le client et le serveur :

1. Ouvrez les fichiers de code de votre projet de jeu et localisez le `.../Games/[your application name]Source/[your application name]Target.cs` fichier. Exemple: `.../Source/GameLiftUnrealAppTarget.cs`. (Si vous utilisez Visual Studio, ouvrez le `.sln` fichier du projet.)
2. Copiez ce fichier pour créer deux nouveaux fichiers cibles dans le `Source/` répertoire.
 - Client cible : renommez le nouveau fichier en `[your application name]Client.Target.cs`. Modifiez le contenu pour mettre à jour le nom de classe et les valeurs du type de cible, comme illustré dans l'exemple de code suivant :

```
using UnrealBuildTool;
using System.Collections.Generic;
```

```

public class GameLiftUnrealAppClientTarget : TargetRules
{
    public GameLiftUnrealAppClientTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Client;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp");
    }
}

```

- Cible du serveur : renommez le nouveau fichier en *[your application name]*Server.Target.cs. Modifiez le contenu pour mettre à jour le nom de classe et les valeurs du type de cible, comme illustré dans l'exemple de code suivant :

```

using UnrealBuildTool;
using System.Collections.Generic;

public class GameLiftUnrealAppServerTarget : TargetRules
{
    public GameLiftUnrealAppServerTarget ( TargetInfo Target ) : base
    ( Target )
    {
        Type = TargetType.Server;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        IncludeOrderVersion = EngineIncludeOrderVersion.Unreal5_1;
        ExtraModuleNames.Add( "GameLiftUnrealApp");
    }
}

```

3. Régénérez vos fichiers de projet. Si vous utilisez Visual Studio, vous pouvez cliquer avec le bouton droit sur le .uproject fichier de votre projet de jeu et sélectionner Générer des fichiers de projet Visual Studio.

Pour mettre à jour les règles du module de projet de jeu :

Mettez à jour les règles du module du projet de jeu pour qu'il devienne dépendant du plugin.

1. Ouvrez les fichiers de code de votre projet de jeu et localisez le .../Games/*[your application name]*Source/*[your application name]*.Build.cs fichier. Exemple:

.../Source/GameLiftUnrealApp.Build.cs. (Si vous utilisez Visual Studio, ouvrez le .sln fichier du projet.)

2. Localisez la `ModuleRules` classe et mettez-la à jour comme illustré dans l'exemple de code suivant :

```
using UnrealBuildTool;

public class GameLiftUnrealApp : ModuleRules
{
    public GameLiftUnrealApp ( ReadOnlyTargetRules Target ) : base ( Target )
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
        PublicDependencyModuleNames.AddRange( new string[] { "Core",
"CoreUObject", "Engine", "InputCore", "HeadMountedDisplay",
"EnhancedInput" });
        // Add the following section
        if (Target.Type == TargetType.Server)
        {
            PublicDependencyModuleNames.Add("GameLiftServerSDK");
        }
        else
        {
            PublicDefinitions.Add("WITH_GAMELIFT=0");
        }
        bEnableExceptions = true;
    }
}
```

3. Après avoir créé les nouveaux fichiers cibles et modifié les règles du module, reconstruisez votre projet de jeu.

Ajoutez la fonctionnalité d'hébergement de jeux au code de votre serveur

Après l'installation et la configuration du SDK du serveur, l'étape suivante consiste à intégrer la fonctionnalité d'hébergement de jeux dans le code de votre serveur. Le SDK du serveur permet à votre serveur de jeu de communiquer avec le Amazon GameLift Servers service, de recevoir des instructions pour les sessions de jeu, de signaler l'état et l'état de santé et d'effectuer d'autres actions.

Cette rubrique fournit un exemple de code qui ajoute les fonctionnalités minimales requises pour héberger votre jeuAmazon GameLift Servers.

Étape 1 : mise à jour du fichier **GameMode** d'en-tête

1. Ouvrez les fichiers de code de votre projet de jeu et localisez le *Your-application-name*GameMode.h fichier. Exemple: GameLiftUnrealAppGameMode.h. Si vous utilisez Visual Studio, ouvrez le .sln fichier correspondant à votre projet de jeu.
2. Modifiez le fichier d'en-tête pour inclure l'exemple de code suivant. Assurez-vous de remplacer « GameLiftUnrealApp » par le nom de votre propre application.

Exemple de code pour GameMode.h

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "GameLiftUnrealAppGameMode.generated.h"

struct FProcessParameters;

DECLARE_LOG_CATEGORY_EXTERN(GameServerLog, Log, All);

UCLASS(minimalapi)
class AGameLiftUnrealAppGameMode : public AGameModeBase
{
    GENERATED_BODY()

public:
    AGameLiftUnrealAppGameMode();

protected:
    virtual void BeginPlay() override;

private:
    void InitGameLift();

private:
    TSharedPtr<FProcessParameters> ProcessParameters;
};
```

Étape 2 : ajouter les appels du SDK du serveur requis au code de votre serveur de jeu

Utilisez l'exemple de code présenté dans cette section pour intégrer le code de votre serveur de jeu afin de l'utiliser avec Amazon GameLift Servers. Pour plus de détails sur le fonctionnement du code, reportez-vous [Initialiser le processus du serveur](#) aux sections et [SDK 5.x du serveur C++ \(Unreal\)](#) pour Amazon GameLift Servers -- Actions.

Note

L'indicateur `WITH_GAMELIFT` du préprocesseur a deux objectifs :

- Limite les appels d'API Amazon GameLift Servers du backend aux versions du serveur Unreal uniquement
- Garantit la compatibilité entre les différentes cibles de build d'Unreal

1. Ouvrez le *Your-application-name*GameMode.cpp fichier source correspondant. Dans notre exemple :GameLiftUnrealAppGameMode.cpp.
2. Modifiez le code pour l'aligner sur l'exemple de code suivant. Assurez-vous de remplacer toute instance de « GameLiftUnrealApp » par le nom de votre propre application.

L'exemple de code fourni montre comment ajouter les éléments requis pour l'intégration avec Amazon GameLift Servers. Il s'agit des licences suivantes :

- Initialisez un client d'Amazon GameLift Servers API.
- Implémentez des fonctions de rappel pour répondre aux demandes du Amazon GameLift Servers service, notamment `OnStartGameSession`, `OnProcessTerminate`, et `onHealthCheck`.
- Appelez `ProcessReady()` pour informer le Amazon GameLift Servers service lorsque vous êtes prêt à organiser des sessions de jeu.

Exemple de code pour GameMode.cpp

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
  
#include "GameLiftUnrealAppGameMode.h"
```



```

#include "UObject/ConstructorHelpers.h"
#include "Kismet/GameplayStatics.h"

#ifdef WITH_GAMELIFT
#include "GameLiftServerSDK.h"
#include "GameLiftServerSDKModels.h"
#endif

#include "GenericPlatform/GenericPlatformOutputDevices.h"

DEFINE_LOG_CATEGORY(GameServerLog);

AGameLiftUnrealAppGameMode::AGameLiftUnrealAppGameMode() :
    ProcessParameters(nullptr)
{
    // Set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/
ThirdPerson/Blueprints/BP_ThirdPersonCharacter"));

    if (PlayerPawnBPClass.Class != NULL)
    {
        DefaultPawnClass = PlayerPawnBPClass.Class;
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing AGameLiftUnrealAppGameMode..."));
}

void AGameLiftUnrealAppGameMode::BeginPlay()
{
    Super::BeginPlay();

#ifdef WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
#ifdef WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Calling InitGameLift..."));

    // Getting the module first.
    FGameLiftServerSDKModule* GameLiftSdkModule =
        &FModuleManager::LoadModuleChecked<FGameLiftServerSDKModule>(FName("GameLiftServerSDK"));

```

```
//Define the server parameters for a GameLift Anywhere fleet. These are not needed
for a GameLift managed EC2 fleet.
FServerParameters ServerParametersForAnywhere;

bool bIsAnywhereActive = false;
if (FParse::Param(FCommandLine::Get(), TEXT("glAnywhere")))
{
    bIsAnywhereActive = true;
}

if (bIsAnywhereActive)
{
    UE_LOG(GameServerLog, Log, TEXT("Configuring server parameters for
Anywhere..."));

    // If GameLift Anywhere is enabled, parse command line arguments and pass them
in the ServerParameters object.
    FString glAnywhereWebSocketUrl = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereWebSocketUrl="),
glAnywhereWebSocketUrl))
    {
        ServerParametersForAnywhere.m_webSocketUrl =
TCHAR_TO_UTF8(*glAnywhereWebSocketUrl);
    }

    FString glAnywhereFleetId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereFleetId="),
glAnywhereFleetId))
    {
        ServerParametersForAnywhere.m_fleetId = TCHAR_TO_UTF8(*glAnywhereFleetId);
    }

    FString glAnywhereProcessId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereProcessId="),
glAnywhereProcessId))
    {
        ServerParametersForAnywhere.m_processId =
TCHAR_TO_UTF8(*glAnywhereProcessId);
    }
    else
    {
        // If no ProcessId is passed as a command line argument, generate a
randomized unique string.
```

```
        FString TimeString = FString::FromInt(std::time(nullptr));
        FString ProcessId = "ProcessId_" + TimeString;
        ServerParametersForAnywhere.m_processId = TCHAR_TO_UTF8(*ProcessId);
    }

    FString glAnywhereHostId = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereHostId="),
glAnywhereHostId))
    {
        ServerParametersForAnywhere.m_hostId = TCHAR_TO_UTF8(*glAnywhereHostId);
    }

    FString glAnywhereAuthToken = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAuthToken="),
glAnywhereAuthToken))
    {
        ServerParametersForAnywhere.m_authToken =
TCHAR_TO_UTF8(*glAnywhereAuthToken);
    }

    FString glAnywhereAwsRegion = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAwsRegion="),
glAnywhereAwsRegion))
    {
        ServerParametersForAnywhere.m_awsRegion =
TCHAR_TO_UTF8(*glAnywhereAwsRegion);
    }

    FString glAnywhereAccessKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereAccessKey="),
glAnywhereAccessKey))
    {
        ServerParametersForAnywhere.m_accessKey =
TCHAR_TO_UTF8(*glAnywhereAccessKey);
    }

    FString glAnywhereSecretKey = "";
    if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSecretKey="),
glAnywhereSecretKey))
    {
        ServerParametersForAnywhere.m_secretKey =
TCHAR_TO_UTF8(*glAnywhereSecretKey);
    }
```

```

        FString glAnywhereSessionToken = "";
        if (FParse::Value(FCommandLine::Get(), TEXT("glAnywhereSessionToken="),
glAnywhereSessionToken))
        {
            ServerParametersForAnywhere.m_sessionToken =
TCHAR_TO_UTF8(*glAnywhereSessionToken);
        }

        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_YELLOW);
        UE_LOG(GameServerLog, Log, TEXT(">>>> WebSocket URL: %s"),
*ServerParametersForAnywhere.m_webSocketUrl);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Fleet ID: %s"),
*ServerParametersForAnywhere.m_fleetId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Process ID: %s"),
*ServerParametersForAnywhere.m_processId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Host ID (Compute Name): %s"),
*ServerParametersForAnywhere.m_hostId);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Auth Token: %s"),
*ServerParametersForAnywhere.m_authToken);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Aws Region: %s"),
*ServerParametersForAnywhere.m_awsRegion);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Access Key: %s"),
*ServerParametersForAnywhere.m_accessKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Secret Key: %s"),
*ServerParametersForAnywhere.m_secretKey);
        UE_LOG(GameServerLog, Log, TEXT(">>>> Session Token: %s"),
*ServerParametersForAnywhere.m_sessionToken);
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }

    UE_LOG(GameServerLog, Log, TEXT("Initializing the GameLift Server..."));

    //InitSDK will establish a local connection with GameLift's agent to enable further
communication.
    FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule-
>InitSDK(ServerParametersForAnywhere);
    if (InitSdkOutcome.IsSuccess())
    {
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
        UE_LOG(GameServerLog, Log, TEXT("GameLift InitSDK succeeded!"));
        UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    }
    else
    {

```

```

    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: InitSDK failed : ("));
    FGameLiftError GameLiftError = InitSdkOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"), *GameLiftError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
    return;
}

ProcessParameters = MakeShared<FProcessParameters>();

//When a game session is created, Amazon GameLift Servers sends an activation
request to the game server and passes along the game session object containing game
properties and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
invoke GameLiftServerAPI.ActivateGameSession()
ProcessParameters->OnStartGameSession.BindLambda([=]
(Aws::GameLift::Server::Model::GameSession InGameSession)
{
    FString GameSessionId = FString(InGameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"),
*GameSessionId);
    GameLiftSdkModule->ActivateGameSession();
});

//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback
before shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to
shutdown.
ProcessParameters->OnTerminate.BindLambda([=]()
{
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    // First call ProcessEnding()
    FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule-
>ProcessEnding();
    // Then call Destroy() to free the SDK from memory
    FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();
    // Exit the process with success or failure
    if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess()) {
        UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
    }
    else {

```

```

        if (!processEndingOutcome.IsSuccess()) {
            const FGameLiftError& error = processEndingOutcome.GetError();
            UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error:
%s"),
                error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
                *error.m_errorMessage);
        }
        if (!destroyOutcome.IsSuccess()) {
            const FGameLiftError& error = destroyOutcome.GetError();
            UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
                error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
                *error.m_errorMessage);
        }
    }
});

//This is the HealthCheck callback.
//Amazon GameLift Servers will invoke this callback every 60 seconds or so.
//Here, a game server might want to check the health of dependencies and such.
//Simply return true if healthy, false otherwise.
//The game server has 60 seconds to respond with its health status. Amazon GameLift
Servers will default to 'false' if the game server doesn't respond in time.
//In this case, we're always healthy!
ProcessParameters->OnHealthCheck.BindLambda([]()
{
    UE_LOG(GameServerLog, Log, TEXT("Performing Health Check"));
    return true;
});

//GameServer.exe -port=7777 LOG=server.mylog
ProcessParameters->port = FURL::UrlConfig.DefaultPort;
TArray<FString> CommandLineTokens;
TArray<FString> CommandLineSwitches;

FCommandLine::Parse(FCommandLine::Get(), CommandLineTokens, CommandLineSwitches);

for (FString SwitchStr : CommandLineSwitches)
{
    FString Key;
    FString Value;

    if (SwitchStr.Split("=", &Key, &Value))
    {
        if (Key.Equals("port"))

```

```

        {
            ProcessParameters->port = FString::Atoi(*Value);
        }
    }
}

//Here, the game server tells Amazon GameLift Servers where to find game session
log files.
//At the end of a game session, Amazon GameLift Servers uploads everything in the
specified
//location and stores it in the cloud for access later.
TArray<FString> Logfiles;
Logfiles.Add(TEXT("GameLiftUnrealApp/Saved/Logs/server.log"));
ProcessParameters->logParameters = Logfiles;

//The game server calls ProcessReady() to tell Amazon GameLift Servers it's ready
to host game sessions.
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready..."));
FGameLiftGenericOutcome ProcessReadyOutcome = GameLiftSdkModule-
>ProcessReady(*ProcessParameters);

if (ProcessReadyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_GREEN);
    UE_LOG(GameServerLog, Log, TEXT("Process Ready!"));
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}
else
{
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_RED);
    UE_LOG(GameServerLog, Log, TEXT("ERROR: Process Ready Failed!"));
    FGameLiftError ProcessReadyError = ProcessReadyOutcome.GetError();
    UE_LOG(GameServerLog, Log, TEXT("ERROR: %s"),
*ProcessReadyError.m_errorMessage);
    UE_LOG(GameServerLog, SetColor, TEXT("%s"), COLOR_NONE);
}

UE_LOG(GameServerLog, Log, TEXT("InitGameLift completed!"));
#endif
}

```

Étape 3 : Reconstruire le projet de jeu

- Créez un projet de jeu pour les deux types de cibles suivants : éditeur de développement et serveur de développement.

Note

Il n'est pas nécessaire de reconstruire la solution. Créez plutôt le projet dans le `/Games/` dossier de votre application. Dans le cas contraire, Visual Studio reconstruira l'intégralité UE5 du projet, ce qui peut prendre jusqu'à une heure.

Package de votre serveur de jeu pour l'hébergement

Le code de votre serveur de jeu étant désormais intégré aux fonctionnalités minimales requises du SDK de serveur, vous êtes prêt à emballer la version de votre serveur de jeu à l'aide de l'éditeur Unreal.

Pour emballer le build du serveur de jeu

1. Ouvrez le projet de jeu dans l'éditeur Unreal.
2. Suivez les étapes de l'éditeur Unreal pour emballer votre serveur de jeu :
 - Choisissez votre plateforme cible (Windows ou Linux).
 - Sélectionnez la cible de compilation de votre serveur (*[your application name]*Server).

Le processus d'emballage génère le fichier exécutable de votre serveur de jeu : *[your application name]*Server.exe.

3. Préparez la version de votre serveur de jeu pour le déploiement sur les ressources d'hébergement. Le build doit inclure les fichiers suivants :
 - Le fichier exécutable de votre serveur de jeu
 - Si vous utilisez la version 5.5 ou antérieure d'Unreal Engine, incluez les fichiers suivants pour les versions de Windows. Vous pouvez les trouver dans votre version source de l'Unreal Engine :
 - `VC_redist.x64.exe` (UnrealEngine\Engine\Source\Programs\PrereqInstaller\Resources\VCRedist\)

- UEPrereqSetup_x64.exe or UE5PrereqSetup_x64.exe (UnrealEngine\Engine\Extras\Redist\en-us\)
- Toutes les autres dépendances requises pour votre serveur de jeu.
- bibliothèques OpenSSL, si nécessaire. Vous pouvez ignorer cette étape si votre serveur de jeu est intégré à la version 5.3 ou ultérieure Amazon GameLift Servers du SDK du serveur. [La dernière version du SDK du serveur est disponible ici.](#)

Bibliothèques OpenSSL requises pour les serveurs de jeux créés avec le SDK de serveur pour Unreal, versions 5.2 et antérieures

Vous devez inclure la même version des bibliothèques OpenSSL que celle utilisée lors de l'empaquetage du serveur de jeu dans Unreal. Ces bibliothèques se trouvent dans la source de votre moteur de jeu. L'emplacement varie en fonction de votre environnement de développement :

Sous Windows :

- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libssl-1_1-x64.dll
- [ENGINE_ROOT_DIR]\Engine\Extras\ThirdPartyNotUE\libimobiledevice\x64\libcrypto-1_1-x64.dll

Sous Linux :

- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libssl.so.1.1
- Engine/Source/Thirdparty/OpenSSL/1.1.1n/include/libcrypto.so.1.1

Copiez les bibliothèques OpenSSL dans le package de compilation de votre jeu, dans le même répertoire que le fichier exécutable du serveur de jeu.

Étapes suivantes

Maintenant que vous avez préparé une version de serveur de jeu avec les fonctionnalités minimales requises pour l'hébergement Amazon GameLift Servers, considérez les prochaines étapes potentielles suivantes :

- Déployez votre serveur de jeu intégré pour les tests et le développement. Avec une flotte Anywhere, vous pouvez configurer votre machine locale comme ressource d'hébergement et

l'utiliser pour tester les connexions entre votre serveur de jeu et vos clients de jeu. Pour un hébergement basé sur le cloud, déployez votre serveur de jeu sur un parc de conteneurs géré EC2 ou géré. Consultez les rubriques suivantes pour obtenir des conseils :

- [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#)
- [Amazon GameLift Servers Des flottes partout](#)
- [Amazon GameLift Servers EC2 flottes gérées](#)
- [Amazon GameLift Servers flottes de conteneurs gérées](#)
- Personnalisez l'intégration de votre serveur de jeu en ajoutant des fonctionnalités optionnelles. Par exemple, vous souhaitez peut-être ajouter des sessions de joueur avec un joueur unique IDs, configurer le matchmaking backfill ou gérer l'accès du serveur de jeu à vos autres AWS ressources (comme une base de données ou un service de stockage de contenu). Consultez les rubriques suivantes pour obtenir des conseils :
 - [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
 - [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
- Personnalisez le composant client de votre jeu pour demander des sessions de jeu, recevoir des informations de connexion et vous connecter directement à un serveur de jeu pour jouer à un jeu. Consultez les rubriques suivantes pour obtenir des conseils :
 - [Intégrez votre client de jeu à Amazon GameLift Servers](#)

Intégrer Amazon GameLift Servers dans un projet Unity

Découvrez comment intégrer le Amazon GameLift Servers SDK pour Unity dans vos projets de jeu afin d'accéder à l'ensemble des fonctionnalités du SDK pour serveurs.

Conseil

Pour un déploiement plus rapide, essayez le plugin Amazon GameLift Servers autonome pour Unity. Il fournit des flux de travail d'interface utilisateur guidés pour déployer rapidement votre serveur de jeu avec une configuration minimale, afin que vous puissiez tester les composants de votre jeu en action. Consultez [Amazon GameLift Servers plugin pour Unity \(SDK serveur 5.x\)](#).

Ressources supplémentaires :

- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)

- [the section called “Obtenez des outils de développement”](#)

Installation du SDK du serveur pour Unity

Téléchargez l'open source Amazon GameLift Servers pour Unity auprès de [GitHub](#). Les fichiers readme du référentiel contiennent les prérequis et les instructions d'installation.

Configurez une flotte Amazon GameLift Servers Anywhere à des fins de test

Vous pouvez configurer votre station de développement en tant que parc d'hébergement Amazon GameLift Servers Anywhere pour tester votre Amazon GameLift Servers intégration de manière itérative. Avec cette configuration, vous pouvez démarrer les processus du serveur de jeu sur votre poste de travail, envoyer des demandes d'adhésion à des joueurs ou de matchmaking Amazon GameLift Servers pour démarrer des sessions de jeu, et connecter les clients aux nouvelles sessions de jeu. Avec votre propre station de travail configurée en tant que serveur d'hébergement, vous pouvez surveiller tous les aspects de l'intégration de votre jeu Amazon GameLift Servers.

Pour obtenir des instructions sur la configuration de votre poste de travail, reportez-vous [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#) à la section pour effectuer les étapes suivantes :

1. Créez un emplacement personnalisé pour votre poste de travail.
2. Créez une flotte Amazon GameLift Servers Anywhere avec votre nouvel emplacement personnalisé. En cas de succès, cette demande renvoie un numéro de flotte. Notez cette valeur, car vous en aurez besoin plus tard.
3. Enregistrez votre station de travail comme ordinateur dans le nouveau parc Anywhere. Entrez un nom de calcul unique et spécifiez l'adresse IP de votre station de travail. En cas de succès, cette demande renvoie un point de terminaison du SDK de service, sous la forme d'une WebSocket URL. Notez cette valeur, car vous en aurez besoin plus tard.
4. Générez un jeton d'authentification pour le calcul de votre station de travail. Cette authentification de courte durée inclut le jeton et une date d'expiration. Votre serveur de jeu l'utilise pour authentifier les communications avec le Amazon GameLift Servers service. Stockez l'authentification sur l'ordinateur de votre poste de travail afin que les processus de votre serveur de jeu en cours puissent y accéder.

Ajoutez Amazon GameLift Servers du code serveur à votre projet Unity

Votre serveur de jeu communique avec le Amazon GameLift Servers service pour recevoir des instructions et signaler l'état actuel. Pour ce faire, vous devez ajouter du code de serveur de jeu qui utilise le SDK Amazon GameLift Servers du serveur.

L'exemple de code fourni illustre les éléments d'intégration de base requis. Il utilise un `MonoBehavior` pour illustrer une simple initialisation d'un serveur de jeu avec Amazon GameLift Servers. L'exemple suppose que le serveur de jeu fonctionne sur une flotte Amazon GameLift Servers Anywhere à des fins de test. Il inclut un code pour :

- Initialisez un client d'Amazon GameLift Servers API. L'exemple utilise la version de `InitSDK()` avec les paramètres du serveur pour votre parc et vos calculs Anywhere. Utilisez l' `WebSocket` URL, l'ID de flotte, le nom de calcul (ID d'hôte) et le jeton d'authentification, tels que définis dans la rubrique précédente [Configurez une flotte Amazon GameLift Servers Anywhere à des fins de test](#).
- Implémentez des fonctions de rappel pour répondre aux demandes du Amazon GameLift Servers service, notamment `OnStartGameSession`, `OnProcessTerminate`, et `onHealthCheck`.
- Appelez `ProcessReady()` avec un port désigné pour avertir le Amazon GameLift Servers service lorsque le processus est prêt à héberger des sessions de jeu.

L'exemple de code fourni établit la communication avec le Amazon GameLift Servers service. Il implémente également un ensemble de fonctions de rappel qui répondent aux demandes du Amazon GameLift Servers service. Pour plus d'informations sur chaque fonction et sur le rôle du code, voir [Initialiser le processus du serveur](#). Pour plus d'informations sur les actions du SDK et les types de données utilisés dans ce code, consultez [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#).

L'exemple de code montre comment ajouter les fonctionnalités requises, comme décrit dans la section [Ajouter Amazon GameLift Servers à votre serveur de jeu](#). Pour plus d'informations sur les actions du SDK du serveur, consultez le [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#).

Exemple de code d'intégration

```
using System.Collections.Generic;
using Aws.GameLift.Server;
using UnityEngine;

public class ServerSDKManualTest : MonoBehaviour
```

```
{
    //This example is a simple integration that initializes a game server process
    //that is running on an Amazon GameLift Servers Anywhere fleet.
    void Start()
    {
        //Identify port number (hard coded here for simplicity) the game server is
        //listening on for player connections
        var listeningPort = 7777;

        //WebSocketUrl from RegisterHost call
        var websocketUrl = "wss://us-west-2.api.amazongamelift.com";

        //Unique identifier for this process
        var processId = "myProcess";

        //Unique identifier for your host that this process belongs to
        var hostId = "myHost";

        //Unique identifier for your fleet that this host belongs to
        var fleetId = "myFleet";

        //Authorization token for this host process
        var authToken = "myAuthToken";

        //Server parameters are required for an Amazon GameLift Servers Anywhere fleet.
        //They are not required for an Amazon GameLift Servers managed EC2 fleet.
        ServerParameters serverParameters = new ServerParameters(
            websocketUrl,
            processId,
            hostId,
            fleetId,
            authToken);

        //InitSDK establishes a local connection with an Amazon GameLift Servers agent
        //to enable further communication.
        var initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
        if (initSDKOutcome.Success)
        {
            //Implement callback functions
            ProcessParameters processParameters = new ProcessParameters(
                //Implement OnStartGameSession callback
                (gameSession) => {
                    //Amazon GameLift Servers sends a game session activation request
                    //to the game server
                }
            );
        }
    }
}
```

```

        //with game session object containing game properties and other
settings.

        //Here is where a game server takes action based on the game
session object.

        //When the game server is ready to receive incoming player
connections,

        //it invokes the server SDK call ActivateGameSession().
        GameLiftServerAPI.ActivateGameSession();
    },
    (updateGameSession) => {
        //Amazon GameLift Servers sends a request when a game session is
updated (such as for
        //FlexMatch backfill) with an updated game session object.
        //The game server can examine matchmakerData and handle new
incoming players.
        //updateReason explains the purpose of the update.
    },
    () => {
        //Implement callback function OnProcessTerminate
        //Amazon GameLift Servers invokes this callback before shutting
down the instance hosting this game server.
        //It gives the game server a chance to save its state, communicate
with services, etc.,
        //and initiate shut down. When the game server is ready to shut
down, it invokes the
        //server SDK call ProcessEnding() to tell Amazon GameLift Servers
it is shutting down.
        GameLiftServerAPI.ProcessEnding();
    },
    () => {
        //Implement callback function OnHealthCheck
        //Amazon GameLift Servers invokes this callback approximately every
60 seconds.
        //A game server might want to check the health of dependencies,
etc.

        //Then it returns health status true if healthy, false otherwise.
        //The game server must respond within 60 seconds, or Amazon
GameLift Servers records 'false'.
        //In this example, the game server always reports healthy.
        return true;
    },
    //The game server gets ready to report that it is ready to host game
sessions

    //and that it will listen on port 7777 for incoming player connections.

```

```

        listeningPort,
        new LogParameters(new List<string>()
        {
            //Here, the game server tells Amazon GameLift Servers where to find
game session log files.
            //At the end of a game session, Amazon GameLift Servers uploads
everything in the specified
            //location and stores it in the cloud for access later.
            "/local/game/logs/myserver.log"
        }));

        //The game server calls ProcessReady() to tell Amazon GameLift Servers it's
ready to host game sessions.
        var processReadyOutcome =
GameLiftServerAPI.ProcessReady(processParameters);
        if (processReadyOutcome.Success)
        {
            print("ProcessReady success.");
        }
        else
        {
            print("ProcessReady failure : " +
processReadyOutcome.Error.ToString());
        }
    }
    else
    {
        print("InitSDK failure : " + initSDKOutcome.Error.ToString());
    }
}

void OnApplicationQuit()
{
    //Make sure to call GameLiftServerAPI.ProcessEnding() and
GameLiftServerAPI.Destroy() before terminating the server process.
    //These actions notify Amazon GameLift Servers that the process is terminating
and frees the API client from memory.
    GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
    GameLiftServerAPI.Destroy();
    if (processEndingOutcome.Success)
    {
        Environment.Exit(0);
    }
    else

```

```
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
}
```

Étapes suivantes

Maintenant que vous avez préparé une version de serveur de jeu avec les fonctionnalités minimales requises pour l'hébergement Amazon GameLift Servers, considérez les prochaines étapes potentielles suivantes :

- Déployez votre serveur de jeu intégré pour les tests et le développement. Avec une flotte Anywhere, vous pouvez configurer votre machine locale comme ressource d'hébergement et l'utiliser pour tester les connexions entre votre serveur de jeu et vos clients de jeu. Pour un hébergement basé sur le cloud, déployez votre serveur de jeu sur un parc de conteneurs géré EC2 ou géré. Consultez les rubriques suivantes pour obtenir des conseils :
 - [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#)
 - [Amazon GameLift Servers Des flottes partout](#)
 - [Amazon GameLift Servers EC2 flottes gérées](#)
 - [Amazon GameLift Servers flottes de conteneurs gérées](#)
- Personnalisez l'intégration de votre serveur de jeu en ajoutant des fonctionnalités optionnelles. Par exemple, vous souhaitez peut-être ajouter des sessions de joueur avec un joueur unique IDs, configurer le matchmaking backfill ou gérer l'accès du serveur de jeu à vos autres AWS ressources (comme une base de données ou un service de stockage de contenu). Consultez les rubriques suivantes pour obtenir des conseils :
 - [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
 - [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
- Personnalisez le composant client de votre jeu pour demander des sessions de jeu, recevoir des informations de connexion et vous connecter directement à un serveur de jeu pour jouer à un jeu. Consultez les rubriques suivantes pour obtenir des conseils :
 - [Intégrez votre client de jeu à Amazon GameLift Servers](#)

Concevez le service client de votre jeu

Nous vous recommandons de mettre en place un service client de jeu qui authentifie vos joueurs et communique avec Amazon GameLift Servers API. En mettant en place un service client de jeu personnalisé, vous pouvez :

- Personnalisez l'authentification pour vos joueurs.
- Contrôlez comment Amazon GameLift Servers correspond et démarre des sessions de jeu.
- Utilisez votre base de données de joueurs pour les attributs des joueurs tels que l'évaluation des compétences pour le matchmaking au lieu de faire confiance au client.

L'utilisation d'un service client de jeu réduit également les risques de sécurité introduits par les clients du jeu qui interagissent directement avec votre Amazon GameLift Servers API.

Authentification de vos joueurs

Vous pouvez utiliser Amazon Cognito et la session de joueur IDs pour authentifier les clients de votre jeu. Pour gérer le cycle de vie et les propriétés de l'identité de vos joueurs, utilisez les groupes d'utilisateurs Amazon Cognito.

Si vous préférez, créez une solution d'identité personnalisée et hébergez-la sur AWS. Vous pouvez également utiliser les autorisateurs Lambda pour une logique d'autorisation personnalisée avec API Gateway.

Ressources supplémentaires :

- [Utilisation de groupes d'identités \(identités fédérées\)](#) (Amazon Cognito Developer Guide)
- [Commencer à utiliser les groupes d'utilisateurs](#) (Guide du développeur Amazon Cognito)
- [Comment configurer l'authentification des joueurs avec Amazon Cognito](#) (AWS pour le blog sur les jeux)

Serveurs de session de jeu autonomes avec un backend sans serveur

À l'aide d'une architecture de service client sans serveur, le backend peut consulter l'état des tickets de matchmaking à partir d'une base de données hautement évolutive au lieu d'accéder directement au Amazon GameLift Servers API.

Le schéma suivant montre un backend sans serveur construit avec Services AWS qui associe les joueurs à des jeux exécutés sur Amazon GameLift Servers flottes. La liste suivante fournit une description de chaque légende numérotée du diagramme. Pour essayer cet exemple, voir [Activation de l'hébergement de jeux basé sur une session multijoueur](#). AWS GitHub

1. Le client du jeu demande une identité d'utilisateur Amazon Cognito à un pool d'identités Amazon Cognito.
2. Le client du jeu reçoit des informations d'accès temporaires et demande une session de jeu via une API Amazon API Gateway.
3. API Gateway invoque une AWS Lambda fonction.
4. La fonction Lambda demande les données du joueur à partir d'une table NoSQL Amazon DynamoDB. La fonction fournit l'identité Amazon Cognito dans les données contextuelles de la demande.
5. La fonction Lambda demande une correspondance via Amazon GameLift Servers FlexMatch matchmaking.
6. FlexMatch associe un groupe de joueurs avec une latence appropriée, puis demande le placement d'une session de jeu par le biais d'un Amazon GameLift Servers queue. La file d'attente contient des flottes contenant un ou plusieurs Région AWS emplacements.
7. Après Amazon GameLift Servers place la session sur l'un des sites de la flotte, Amazon GameLift Servers envoie une notification d'événement à une rubrique Amazon Simple Notification Service (Amazon SNS).
8. Une fonction Lambda reçoit l'événement Amazon SNS et le traite.
9. Si le ticket de matchmaking est un MatchmakingSucceeded événement, la fonction Lambda écrit le résultat, ainsi que le port et l'adresse IP du serveur de jeu, dans une table DynamoDB.
- 10 Le client du jeu envoie une demande signée à API Gateway pour consulter l'état du ticket de matchmaking à un intervalle spécifique.
- 11 API Gateway utilise une fonction Lambda qui vérifie l'état du ticket de matchmaking.
- 12 La fonction Lambda vérifie dans la table DynamoDB si le ticket est valide. En cas de succès, la fonction renvoie le port et l'adresse IP du serveur de jeu, ainsi que l'identifiant de session du joueur, au client. Si le ticket n'aboutit pas, la fonction envoie une réponse confirmant que le match n'est pas encore prêt.

13 Le client de jeu se connecte au serveur de jeu via TCP ou UDP en utilisant le port et l'adresse IP fournis par le service principal. Le client du jeu envoie ensuite l'identifiant de session du joueur au serveur de jeu, qui le valide ensuite à l'aide du SDK du serveur pour Amazon GameLift Servers.

Serveurs de session de jeu autonomes avec un backend WebSocket basé

À l'aide d'une architecture WebSocket basée sur Amazon API Gateway, vous pouvez effectuer des demandes de matchmaking WebSockets et envoyer des notifications push pour la fin du matchmaking à l'aide de messages initiés par le serveur. Cette architecture améliore les performances grâce à une communication bidirectionnelle entre le client et le serveur.

Pour plus d'informations sur l'utilisation d'API Gateway WebSock APIs, consultez la section [Travailler avec WebSocket APIs](#).

Le schéma suivant montre une architecture de backend WebSocket basée sur API Gateway et d'autres Services AWS pour associer les joueurs aux jeux exécutés sur Amazon GameLift Servers flottes. La liste suivante fournit une description de chaque légende numérotée du diagramme.

1. Le client du jeu demande une identité d'utilisateur Amazon Cognito à un pool d'identités Amazon Cognito.
2. Le client du jeu signe une WebSocket connexion à une API API Gateway avec les informations d'identification Amazon Cognito.
3. API Gateway appelle une AWS Lambda fonction sur la connexion. La fonction stocke les informations de connexion dans une table Amazon DynamoDB.
4. Le client du jeu envoie un message à une fonction Lambda, via l'API API Gateway via la WebSocket connexion, pour demander une session.
5. Une fonction Lambda reçoit le message puis demande une correspondance via Amazon GameLift Servers FlexMatch matchmaking.
6. Après FlexMatch correspond à un groupe de joueurs, FlexMatch demande le placement d'une session de jeu par le biais d'un Amazon GameLift Servers queue.
7. Après Amazon GameLift Servers place la session sur l'un des sites de la flotte, Amazon GameLift Servers envoie une notification d'événement à une rubrique Amazon Simple Notification Service (Amazon SNS).
8. Une fonction Lambda reçoit l'événement Amazon SNS et le traite.

9. Si le ticket de matchmaking est un MatchmakingSucceeded événement, la fonction Lambda demande à DynamoDB de connecter correctement le joueur. La fonction envoie ensuite un message au client du jeu via l'API API Gateway via la WebSocket connexion. Dans cette architecture, le client du jeu n'interroge pas activement l'état du matchmaking.
- 10 Le client du jeu reçoit le port et l'adresse IP du serveur de jeu, ainsi que l'identifiant de session du joueur, via la WebSocket connexion.
- 11 Le client de jeu se connecte au serveur de jeu via TCP ou UDP en utilisant le port et l'adresse IP fournis par le service principal. Le client du jeu envoie également l'identifiant de session du joueur au serveur de jeu, qui le valide ensuite à l'aide du SDK du serveur pour Amazon GameLift Servers.

Configurez le développement itératif avec Amazon GameLift Servers N'importe où

Amazon GameLift Servers fournit des outils et des solutions pour vous aider à configurer un environnement de test hébergé à utiliser pendant le développement de jeux. Grâce à ces outils, vous pouvez créer un environnement qui reflète l'expérience réelle des joueurs en matière d'hébergement géré avec Amazon GameLift Servers et soutient un processus de développement rapide et itératif.

Avec un environnement de test distinct, vous éliminez la surcharge d'un Amazon GameLift Servers flotte gérée pendant les tests. Vous n'avez plus besoin de télécharger chaque nouvelle itération de build du serveur de jeu, de créer une nouvelle flotte pour celle-ci, puis d'attendre plus de 15 minutes avant de l'activer. Au lieu de cela, vous pouvez créer une nouvelle version, mettre rapidement à jour le parc de tests avec la nouvelle version, la démarrer et commencer les tests.

À l'aide d'un Amazon GameLift Servers Quelle que soit la flotte, vous pouvez configurer un environnement de test à l'aide d'un appareil local, tel que votre poste de développement. Vous pouvez également configurer un environnement de test à l'aide d'une ressource d'hébergement basée sur le cloud.

Configurez un environnement de test Anywhere pour développer et tester une gamme de scénarios, notamment les suivants :

- Testez l'intégration de votre serveur de jeu avec Amazon GameLift Servers SDK du serveur. Vous pouvez effectuer des tests même sans client de jeu fonctionnel en utilisant des appels AWS CLI pour démarrer de nouvelles sessions de jeu et suivre les événements des sessions de jeu.

- Testez les interactions entre le client de votre jeu, le service principal et Amazon GameLift Servers service pendant que vous développez des composants pour votre jeu. Ajustez l'expérience des joueurs lorsqu'ils rejoignent une partie.
- Expérimentez avec votre FlexMatch design entremetteur. Essayez des variantes d'ensembles de règles et d'autres implémentations de fonctionnalités de matchmaking. Configurez et testez le matchmaking backfill.
- Essayez-en d'autres Amazon GameLift Servers les fonctionnalités d'hébergement, telles que les paramètres de configuration d'exécution (avec Amazon GameLift Servers Agent) pour la gestion du cycle de vie des serveurs de jeux.
- Créez, testez et répétez rapidement pour valider tous les aspects de l'expérience de joueur de votre jeu, y compris les interactions multijoueurs, dans un environnement hébergé en direct.

Plus tard, au fur et à mesure que vous préparerez le lancement de votre jeu, vous souhaiterez ajouter Amazon GameLift Servers des flottes gérées pour affiner vos configurations d'hébergement et tester des scénarios supplémentaires, notamment les suivants :

- Expérimentez et testez des modèles de files d'attente pour les sessions de jeu, notamment l'utilisation de flottes multi-sites, de flottes Spot et On-Demand et de plusieurs types d'instances.
- Essayez les options de placement des sessions de jeu avec des flottes gérées, notamment l'utilisation de politiques de latence facultatives et de paramètres de priorisation des flottes.
- Configurez le dimensionnement des capacités pour répondre à la demande des joueurs, à l'aide d'options de dimensionnement automatique ou manuel.
- Configurez AWS CloudFormation avec Amazon GameLift Servers des flottes gérées pour gérer vos ressources d'hébergement à long terme.

Outil de mise à jour rapide (pour le développement uniquement)

Dans le cas EC2 des flottes gérées, pour déployer une mise à jour de version de serveur de jeu, vous devez télécharger chaque nouvelle version sur Amazon GameLift Servers et créez une nouvelle flotte pour cela.

L'outil de mise à jour rapide des versions vous permet de contourner ces étapes pendant le développement, ce qui vous permet de gagner du temps et d'accélérer les itérations de développement. Avec cet outil, vous pouvez rapidement mettre à jour les fichiers de build de votre jeu sur tous les ordinateurs d'une flotte existante. L'outil propose plusieurs options ; vous pouvez remplacer une version complète d'un jeu ou modifier 6 fichiers spécifiques, et

vous pouvez gérer le redémarrage des processus du serveur de jeu après les mises à jour. Vous pouvez également l'utiliser pour mettre à jour les calculs individuels d'une flotte. Pour obtenir l'outil de mise à jour rapide des versions et en savoir plus sur son utilisation, consultez le Amazon GameLift Servers Référentiel de boîtes à outils pour [l'outil Fast Build Update](#) sur Github.

Rubriques

- [Créez un environnement de test basé sur le cloud](#)
- [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#)
- [Collaborez avec l'Amazon GameLift Serversagent](#)

Créez un environnement de test basé sur le cloud

Note

Cette rubrique couvre les tests itératifs pour les jeux intégrés au SDK du serveur pour Amazon GameLift Servers version 5.x. Si votre jeu utilise la version 4.x ou antérieure du SDK du serveur, consultez. [Testez votre intégration à l'aide de Amazon GameLift Servers Local](#)

Utilisez un Amazon GameLift Servers Une flotte n'importe où pour créer et tester de manière itérative les composants de votre jeu dans un environnement hébergé dans le cloud. Créez une flotte Anywhere avec des ressources d'hébergement et une connexion au Amazon GameLift Servers service, exécutez vos serveurs de jeu dessus et testez les fonctionnalités du jeu selon les besoins.

Déployez une flotte n'importe où avec le Amazon GameLift Servers Agent

Si la version de votre serveur de jeu est intégrée à Amazon GameLift Servers SDK 5.x ou version ultérieure, vous pouvez le déployer sur une flotte Anywhere basée sur le cloud avec Amazon GameLift Servers Agent. L'agent est un processus d'arrière-plan qui gère le cycle de vie des serveurs de jeu et les autres tâches associées à chaque ordinateur d'une flotte. Ces tâches incluent l'enregistrement du calcul auprès d'une flotte Anywhere, l'acquisition d'un jeton d'authentification et le démarrage/arrêt des processus du serveur de jeu en fonction d'un ensemble d'instructions. L'agent est contrôlé par la configuration d'exécution d'une flotte, que vous pouvez mettre à jour à tout moment pendant le cycle de vie de la flotte. (L'agent est automatiquement déployé dans les EC2

flottes gérées.) Pour plus d'informations et pour télécharger l'agent, consultez le [Amazon GameLift Servers GitHub référentiel](#).

Configurez des tests itératifs avec Amazon EC2

Utilisez le flux de travail guidé dans ce [Amazon GameLift Servers solution](#) de boîte à outils pour configurer un environnement d'hébergement basé sur le cloud qui reflète l'expérience d'hébergement géré avec Amazon GameLift Servers.

Le GitHub référentiel fournit un ensemble de scripts qui automatisent la plupart des processus de configuration d'un environnement de test avec Amazon GameLift Servers N'importe où et Amazon GameLift Servers Agent. Il fournit également des conseils pour mettre à jour l'environnement chaque fois que vous devez tester une nouvelle version de serveur de jeu. Vous pouvez exécuter un script unique qui déploie un environnement de test avec un exemple de version de serveur de jeu, ou vous pouvez suivre chaque étape pour le configurer avec votre propre version de serveur de jeu.

Dans ce flux de travail, vous travaillerez entièrement dans le AWS Management Console, en utilisant AWS CloudShell pour exécuter des scripts et effectuer des tâches en ligne de commande.

Note

Pour les tâches décrites dans ce didacticiel, vous avez besoin d'un utilisateur du AWS compte autorisé à accéder aux services suivants : Amazon GameLift Servers AWS CloudShell, Amazon S3 AWS Systems Manager EC2, Amazon et AWS Identity and Access Management. Les utilisateurs disposant d'un accès de niveau administrateur au AWS compte disposent déjà des autorisations requises.

Le flux de travail couvre les tâches suivantes :

- Package d'un serveur de jeu développé pour Amazon GameLift Servers. Le flux de travail fournit un script pour créer un exemple de serveur de jeu C++, qui a déjà été intégré au SDK du serveur pour Amazon GameLift Servers version 5.x et est prête à être hébergée. Vous pouvez également travailler sur votre propre projet de jeu si vous avez terminé l'intégration.
- Configurez un bucket Amazon Simple Storage Service pour stocker les versions et les dépendances des serveurs de jeu. Au fur et à mesure que vous produisez de nouvelles versions de vos builds de jeu, vous pouvez les stocker dans S3 et utiliser les scripts pour mettre à jour la flotte Anywhere à des fins de test de jeux.

- Obtenez et construisez le Amazon GameLift Servers Agent. L'agent gère les processus du serveur de jeu sur une ressource d'hébergement en fonction de votre configuration. Il utilise la même logique et se comporte de la même manière que Amazon GameLift Servers EC2 hébergement géré.
- Configurez une flotte Anywhere pour vos ressources d'hébergement. Avec une flotte Anywhere, vous pouvez utiliser le Amazon GameLift Servers service d'hébergement de ressources qui ne sont pas gérées par Amazon GameLift Servers. Au cours de cette étape, vous allez également configurer la configuration d'exécution, qui indique Amazon GameLift Servers Agent quand et comment démarrer les processus du serveur de jeu.
- Configurez une EC2 instance Amazon. Il s'agit de votre environnement de test pour les tests itératifs. Il est beaucoup plus rapide d'utiliser une EC2 instance standard plutôt qu'une instance entièrement gérée Amazon GameLift Servers instance (optimisée pour une utilisation au niveau de la production). Avec une EC2 instance standard, vous pouvez rapidement et continuellement mettre à jour le serveur de jeu selon vos besoins.
- Déployez la version de votre serveur de jeu et Amazon GameLift Servers Agent de l' EC2 instance Amazon. Le flux de travail fournit un script qui récupère la dernière version de votre build de jeu et toutes les dépendances et l'installe sur votre EC2 instance. Dans ce flux de travail, les dépendances incluent Amazon GameLift Servers L'agent et l' CloudWatch agent.
- Démarrez le Amazon GameLift Servers Agent. Une fois installé, l'agent démarre automatiquement et commence à exécuter les instructions. Il s'agit des licences suivantes :
 - Enregistrez l' EC2 instance en tant que calcul dans le Amazon GameLift Servers Une flotte n'importe où.
 - Établissez une WebSocket connexion avec le Amazon GameLift Servers service et obtenez la dernière configuration d'exécution.
 - Démarrez les processus du serveur de jeu en fonction des instructions de la configuration d'exécution. Dans ce flux de travail, l'agent est chargé de démarrer un seul processus de l'exécutable du serveur de jeu.
- Testez vos scénarios de jeu. Une fois l'environnement de test configuré et la dernière version de votre serveur de jeu installée, vous pouvez commencer les tests. Le flux de travail passe par plusieurs étapes de test, notamment le démarrage d'une session de jeu. Accédez aux journaux du serveur de CloudWatch jeu pour suivre la progression au fur et à mesure que la session de jeu démarre et se prépare à accepter des joueurs.

Au fur et à mesure que vous développez les composants de votre jeu, y compris un client de jeu et un service principal côté client, vous pouvez les inclure dans vos scénarios de test. Utilisez un

client de jeu pour demander une session de jeu, récupérer les informations de connexion depuis Amazon GameLift Servers service, puis connectez-vous directement à la session de jeu.

- Déployez une nouvelle version de serveur de jeu et répétez les tests. Au fur et à mesure que vous développez votre jeu, vous pouvez générer de nouvelles versions de serveurs de jeu, puis les déployer rapidement dans l'environnement de EC2 test pour les tester. Téléchargez-les dans le compartiment Amazon S3, puis utilisez les scripts de flux de travail pour mettre à jour l'environnement de test.

Transférez votre jeu vers Amazon GameLift Servers flottes gérées

Une fois que vous avez terminé les tests de développement et que vous êtes prêt à préparer le lancement, c'est le bon moment pour passer à Amazon GameLift Servers flottes gérées. Utilisez des flottes gérées pour affiner et tester les ressources d'hébergement de vos jeux. Mettez en œuvre votre solution de placement de sessions de jeu (files d'attente et entremetteurs), sélectionnez le matériel d'hébergement (y compris les flottes Spot) et les emplacements optimaux, et choisissez une stratégie pour augmenter la capacité. Vous pouvez également commencer à les utiliser AWS CloudFormation pour gérer plus efficacement le cycle de vie de toutes vos ressources d'hébergement de jeux, y compris les flottes, les files d'attente et les entremetteurs.

La transition d'un parc de tests Anywhere basé sur le cloud à un Amazon GameLift Servers flotte gérée. Vous n'avez pas besoin de modifier le code du jeu et vous pouvez réutiliser les mêmes files d'attente et les mêmes matchmakers. Effectuez les tâches suivantes :

- Créez un Amazon GameLift Servers créer une ressource. Avec une flotte de test Anywhere, vous devez déployer manuellement la version de votre serveur de jeu et ses dépendances sur le calcul de chaque flotte. Avec une flotte gérée, téléchargez votre package de construction de jeu sur Amazon GameLift Servers, qui le déploie automatiquement sur tous les ordinateurs du parc. Consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#) pour en savoir plus sur l'emballage des fichiers de compilation de votre jeu et la création d'une ressource de compilation avec des fichiers dans un compartiment Amazon S3.
- Créez une flotte gérée. Créez un parc à l'aide de la console ou de la AWS CLI, en spécifiant un parc EC2 géré. Ce type de parc nécessite des paramètres de configuration supplémentaires, notamment la spécification des ressources de construction et des types d'instances. Vous pouvez utiliser la même configuration d'exécution pour gérer le cycle de vie du serveur de jeu sur chaque ordinateur de flotte. Consultez [Création d'une EC2 flotte Amazon GameLift Servers gérée](#) pour plus de détails sur la création d'une flotte gérée.

- Redirigez les alias de flotte (facultatif). Si vous configurez des alias à utiliser avec vos flottes Anywhere, vous pouvez réutiliser les mêmes alias pour vos flottes gérées. Consultez [Créez un Amazon GameLift Servers alias](#) pour plus de détails sur la création ou la mise à jour d'un alias.

Configurez des tests locaux avec Amazon GameLift Servers Anywhere

Note

Cette rubrique couvre les tests locaux pour les jeux intégrés au SDK du serveur pour la Amazon GameLift Servers version 5.x. Si votre jeu utilise la version 4.x ou antérieure du SDK du serveur, consultez. [Testez votre intégration à l'aide de Amazon GameLift Servers Local](#)

Utilisez une flotte Amazon GameLift Servers Anywhere et votre propre matériel pour créer et tester de manière itérative les composants de votre jeu dans un environnement hébergé simulé. Configurez une flotte Anywhere et enregistrez un appareil local pour établir une connexion au Amazon GameLift Servers service. Installez la version intégrée de votre serveur de jeu sur l'appareil, lancez un processus de serveur de jeu et testez les fonctionnalités du jeu selon vos besoins. Vous pouvez mettre à jour la version de votre serveur de jeu aussi souvent que nécessaire pour tester chaque nouvelle itération de version.

Avec une flotte Anywhere, vous pouvez effectuer des tests à l'aide de la AWS CLI ou de scripts de test. Si vous avez intégré un client de jeu Amazon GameLift Servers, vous pouvez exécuter le client sur le même appareil local ou sur un autre appareil.

Les tests locaux avec une flotte Anywhere sont particulièrement utiles pour tester l'intégration de votre serveur de jeu avec Amazon GameLift Servers. Vous avez une visibilité complète sur toutes les activités d'hébergement sur la machine locale, ainsi que sur les événements et les données de journalisation.

Note

Utilisez-vous le Amazon GameLift Servers plugin pour Unreal Engine ou Unity ? Ces outils incluent des flux de travail guidés pour configurer des tests locaux avec une flotte Anywhere. Suivez la documentation pour [Plugin pour Unity : configurez des tests locaux avec Amazon GameLift Servers Anywhere](#) ou [Plugin pour Unreal : hébergez votre jeu localement avec Anywhere Amazon GameLift Servers](#).

Rubriques

- [Configurez une flotte Anywhere locale](#)
- [Mettez à jour et installez votre serveur de jeu](#)
- [Testez l'activité des sessions de jeu](#)
- [Itérez sur votre serveur de jeu](#)
- [Passez de votre jeu à des flottes Amazon GameLift Servers gérées](#)

Configurez une flotte Anywhere locale

Suivez ces étapes pour créer un parc Anywhere pour votre poste de travail local. Pour obtenir des instructions détaillées à l'aide de la AWS CLI ou du AWS Management Console for Amazon GameLift Servers, consultez [Créez un Amazon GameLift Servers Flotte Anywhere](#).

Pour créer la flotte Anywhere

1. Créez un emplacement personnalisé pour votre poste de travail local. (AWS CLI ou console). Un emplacement personnalisé est simplement une étiquette pour la ressource de calcul que vous prévoyez d'inclure dans votre flotte Anywhere. Les noms de lieux personnalisés doivent commencer par custom-. Par exemple : custom-my_laptop. Consultez [Créez un emplacement personnalisé](#).
2. Créez une flotte Anywhere (AWS CLI ou console). Au cours de cette étape, créez la ressource de flotte avec l'emplacement personnalisé de votre poste de travail local. Consultez [Créez une flotte Anywhere](#).

Prenez note de l'ID ou de la valeur ARN de la nouvelle flotte. Vous aurez besoin de cette valeur à l'étape suivante.

3. Enregistrez votre station de travail locale en tant que parc informatique (AWS CLI uniquement). Une flotte Anywhere doit disposer d'au moins une ressource de calcul pour héberger vos serveurs de jeu. Consultez [Ajouter un ordinateur à la flotte](#). Pour ajouter un ordinateur à la flotte, vous avez besoin des informations suivantes :
- Un nom de calcul. Chaque ordinateur d'un parc doit avoir un nom unique.
 - L'identifiant de flotte Anywhere. Vous pouvez utiliser le FleetID ou FleetArn.
 - Les informations de connexion de l'ordinateur. Spécifiez une IpAddress ou DnsName. C'est ainsi que Amazon GameLift Servers les clients du jeu se connecteront aux serveurs de jeu.
 - Un emplacement personnalisé dans la flotte Anywhere.

Notez la valeur de `GameLiftServiceSdkEndpoint` retour. Vous aurez besoin de cette valeur lorsque vous mettrez à jour votre serveur de jeu pour qu'il fonctionne sur une flotte n'importe où.

Mettez à jour et installez votre serveur de jeu

Cette tâche suppose que vous avez déjà intégré une version de serveur de jeu au SDK 5.x Amazon GameLift Servers du serveur. Le processus d'intégration consiste à ajouter du code à votre serveur de jeu afin qu'il puisse interagir avec le Amazon GameLift Servers service pour démarrer et gérer des sessions de jeu.

Pour une flotte Anywhere, vous devez configurer manuellement certains paramètres du serveur de jeu. Sur un parc Amazon GameLift Servers géré, ces paramètres sont configurés automatiquement.

Pour préparer votre serveur de jeu à une flotte Anywhere

1. Obtenez un jeton d'authentification. Votre serveur de jeu doit inclure un jeton d'authentification à chaque communication avec le Amazon GameLift Servers service. Amazon GameLift Servers les jetons d'authentification sont de courte durée et doivent être régulièrement actualisés.

Il est recommandé de créer un script pour effectuer les tâches suivantes :

- Appelez l'action AWS CLI `get-compute-auth-token`.
- Stockez la valeur du jeton renvoyée là où les processus du serveur de jeu peuvent la récupérer, par exemple dans une variable d'environnement sur le calcul local.

Installez le script avec votre serveur de jeu sur l'ordinateur. Configurez le script pour qu'il s'exécute avant de démarrer le premier processus du serveur de jeu. Lorsque les processus du serveur de jeu sont actifs, exécutez régulièrement le script pour conserver un jeton d'authentification valide. Tous les processus du serveur de jeu présents sur l'ordinateur peuvent utiliser le même jeton d'authentification.

2. Mettez à jour le code de votre serveur de Amazon GameLift Servers jeu. Lorsque vous avez intégré le code de votre serveur de jeu au SDK du serveur pour Amazon GameLift Servers, vous avez ajouté un appel à l'action `InitSdk()`. Lorsque le serveur de jeu fonctionne sur une flotte Anywhere, cet appel nécessite des paramètres de serveur supplémentaires. Pour plus d'informations, consultez [Initialiser le processus du serveur](#) et le correspondant [SDK de serveur](#)

[5.x pour Amazon GameLift Servers](#) à votre langage de développement. Les paramètres du serveur sont les suivants :

- `websocketUrl`— Définissez ce paramètre sur la `GameLiftServiceSdkEndpoint` valeur renvoyée lorsque vous enregistrez un calcul auprès de la flotte.
- `hostId`— Définissez ce paramètre sur le nom de l'ordinateur que vous spécifiez lorsque vous enregistrez un ordinateur dans le parc Anywhere.
- `fleetId`— Définissez ce paramètre sur l'ID de la flotte Anywhere.
- `authToken`— Définissez ce paramètre sur le jeton renvoyé en réponse à une demande de récupération d'un jeton d'authentification pour un calcul.
- `processId`— Définissez ce paramètre pour identifier un processus de serveur de jeu en cours d'exécution sur l'ordinateur local. Chaque processus de serveur de jeu simultané doit avoir un identifiant de processus unique.

Les valeurs des paramètres du serveur utilisées par chaque processus du serveur de jeu doivent être spécifiques au calcul de la flotte Anywhere sur lequel le processus est exécuté. Pour plus de détails sur la façon d'obtenir les valeurs appropriées pour un calcul, consultez [Ajouter un ordinateur à la flotte](#). En tant que bonne pratique, définissez `websocketUrl`, `hostId`, `fleetId`, et `authToken` en tant que variables d'environnement sur le calcul local. Tous les processus du serveur exécutés sur le calcul utiliseront ces valeurs.

3. Installez la version du serveur de jeu sur l'ordinateur local. Incluez toutes les dépendances nécessaires au fonctionnement du serveur de jeu.
4. Démarrez un ou plusieurs processus du serveur de jeu exécutés sur l'ordinateur local. Lorsque le processus du serveur de jeu appelle l'action du SDK du serveur `ProcessReady()`, le processus est prêt à héberger une session de jeu.

Testez l'activité des sessions de jeu

Testez l'intégration de votre serveur de jeu en utilisant des sessions de jeu. Si aucune Amazon GameLift Servers fonctionnalité de client de jeu n'est intégrée, vous pouvez utiliser la AWS CLI pour démarrer des sessions de jeu. Essayez les scénarios suivants :

- Créez une session de jeu. [create-game-session](#) Commande d'appel (ou opération [CreateGameSession](#) d'API). Spécifiez l'identifiant et l'emplacement personnalisé de votre flotte Anywhere. Cet appel renvoie un identifiant unique pour la nouvelle session de jeu.

- Vérifiez l'état de la session de jeu. [describe-game-sessions](#) Commande d'appel (ou action d'[DescribeGameSessions](#) API). Spécifiez l'identifiant de session de jeu. Cet appel renvoie des informations détaillées sur la session de jeu, y compris le statut de la session de jeu. Les sessions de jeu en statut Actif sont prêtes à être connectées par les joueurs. Pour obtenir la liste de toutes les sessions de jeu de la flotte, appelez la [list-game-sessions](#) commande (ou l'action de l'[ListGameSessions](#) API).
- Connectez-vous à la session de jeu. Si votre client de jeu est en mesure de rejoindre une session de jeu, utilisez les informations de connexion incluses dans les informations de session de jeu.

Itérez sur votre serveur de jeu

Vous pouvez utiliser la même Anywhere flotte et les mêmes calculs pour tester d'autres versions de la version de votre serveur de jeu.

1. Nettoyez votre existant **GameSession**. Si le processus du serveur de jeu plante ou refuse d'appeler `ProcessEnding()`, Amazon GameLift Servers nettoie le système une `GameSession` fois que le serveur de jeu a cessé d'envoyer des bilans de santé.
2. Générez une nouvelle version de serveur de jeu. Apportez des modifications à votre serveur de jeu et empaquetez une version révisée.
3. Mettez à jour la version du serveur de jeu sur votre ordinateur local. Votre ancienne flotte Anywhere est toujours active et votre ordinateur portable est toujours enregistré en tant que ressource informatique dans la flotte.
4. Obtenez un jeton d'autorisation mis à jour. Appelez la commande [get-compute-auth-token](#) CLI et stockez le jeton sur l'ordinateur local.
5. Démarrez un ou plusieurs processus du serveur de jeu exécutés sur l'ordinateur local. Lorsque le serveur de jeu traite les appels `ProcessReady()`, il est prêt à être utilisé pour les tests.

Passez de votre jeu à des flottes Amazon GameLift Servers gérées

Une fois que vous avez terminé les tests de développement et que vous êtes prêt à préparer le lancement, c'est le bon moment pour passer aux flottes Amazon GameLift Servers gérées. Utilisez des flottes gérées pour affiner et tester les ressources d'hébergement de vos jeux. Mettez en œuvre votre solution de placement de sessions de jeu (files d'attente et entremetteurs), sélectionnez le matériel d'hébergement (y compris les flottes Spot) et les emplacements optimaux, et choisissez une stratégie pour augmenter la capacité. Vous pouvez également commencer à les utiliser AWS

CloudFormation pour gérer plus efficacement le cycle de vie de toutes vos ressources d'hébergement de jeux, y compris les flottes, les files d'attente et les entremetteurs.

Vous devez apporter quelques modifications mineures pour passer d'une flotte de test Anywhere locale à une flotte Amazon GameLift Servers gérée. Vous pouvez réutiliser les mêmes files d'attente et les mêmes entremetteurs. Effectuez les tâches suivantes :

- Changez l'appel du code du serveur de jeu en **InitSdk()**. Supprimez les paramètres du serveur. Pour une flotte gérée, suit Amazon GameLift Servers automatiquement ces informations.
- Créez une ressource Amazon GameLift Servers de construction. Avec une flotte de test Anywhere, vous devez déployer manuellement la version de votre serveur de jeu et ses dépendances sur le calcul de chaque flotte. Avec une flotte gérée, vous pouvez créer et télécharger votre package de build de jeu Amazon GameLift Servers, qui le déploie automatiquement sur tous les ordinateurs de la flotte. Consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#) pour en savoir plus sur l'emballage des fichiers de compilation de votre jeu et la création d'une ressource de compilation avec des fichiers dans un compartiment Amazon S3. N'incluez pas de scripts qui enregistrent un calcul et obtiennent un jeton d'authentification, car ces tâches sont Amazon GameLift Servers automatiquement gérées avec des flottes gérées.
- Créez une flotte gérée. Créez un parc à l'aide de la console ou de la AWS CLI, en spécifiant un parc EC2 géré. Ce type de parc nécessite des paramètres de configuration supplémentaires, notamment la spécification des ressources de construction et des types d'instances. Vous devez tous configurer une configuration d'exécution pour gérer le cycle de vie du serveur de jeu sur chaque parc informatique. Consultez [Création d'une EC2 flotte Amazon GameLift Servers gérée](#) pour plus de détails sur la création d'une flotte gérée.
- Redirigez les alias de flotte (facultatif). Si vous configurez des alias à utiliser avec vos flottes Anywhere, vous pouvez réutiliser les mêmes alias pour vos flottes gérées. Consultez [Créez un Amazon GameLift Servers alias](#) pour plus de détails sur la création ou la mise à jour d'un alias.

Collaborez avec l'Amazon GameLift Serversagent

L'Amazon GameLift Serversagent supervise le fonctionnement des processus du serveur de jeu sur vos Amazon GameLift Servers flottes. L'agent est déployé sur chaque ordinateur d'un parc et assure la gestion automatisée des processus, la gestion de l'hébergement et la journalisation du calcul. Pour utiliser l'agent, la version de votre serveur de jeu doit être intégrée au SDK du serveur pour la Amazon GameLift Servers version 5.x ou ultérieure.

L'Amazon GameLift Serversagent est disponible en externe pour être utilisé avec des Amazon GameLift Servers flottes qui ne sont pas des EC2 flottes gérées. (EC2 Les flottes gérées gèrent automatiquement les tâches de l'agent.) Vous pouvez choisir de gérer Amazon GameLift Servers des flottes, y compris des flottes Anywhere, avec ou sans l'agent. Sans l'agent, vous devez proposer une solution alternative pour effectuer les tâches requises.

Lorsqu'il est déployé sur un ordinateur, l'Amazon GameLift Serversagent doit être lancé avant le démarrage des processus du serveur de jeu. Au lancement, l'agent exécute les tâches suivantes :

- Enregistre le calcul auprès d'une flotte Amazon GameLift Servers Anywhere à l'aide de l'[RegisterComputeAPI](#).
- Appelle l'[GetComputeAuthTokenAPI](#) pour récupérer un jeton d'autorisation et le stocke pour qu'il soit utilisé par les processus du serveur qui s'exécutent sur le calcul.
- Définit la variable d'environnement WebSocket URL pour le calcul et établit une WebSocket connexion au Amazon GameLift Servers service.
- Demande au Amazon GameLift Servers service la dernière version de la configuration d'exécution de la flotte.
- Démarre et arrête les processus du serveur conformément aux instructions de configuration d'exécution.

Le code source et les instructions de compilation de l'Amazon GameLift Serversagent sont disponibles dans l'[Amazon GameLift Serversagent](#) GitHub.

À propos de l'agent

L'Amazon GameLift Serversagent est conçu pour effectuer les tâches suivantes pour vos flottes :

Gestion des processus

- Démarre de nouveaux processus serveur tels que définis dans les instructions d'exécution. L'agent peut utiliser une configuration d'exécution personnalisée déployée avec l'agent. Vous pouvez également fournir un dans `RuntimeConfiguration` le cadre de la définition de votre flotte. Cette approche présente l'avantage de pouvoir modifier la configuration d'exécution de la flotte à tout moment. L'agent demande régulièrement au Amazon GameLift Servers service des configurations d'exécution mises à jour.
- Surveille les activations des processus du serveur et met fin aux processus lorsqu'ils ne s'activent pas à temps.

- Envoie des battements de cœur à Amazon GameLift Servers. Si l'agent ne parvient pas à envoyer les pulsations cardiaques, le calcul peut être marqué comme obsolète.
- Signale Amazon GameLift Servers la fin d'un processus serveur. Amazon GameLift Servers utilise ces informations pour surveiller la disponibilité du serveur de jeu pour le placement des sessions de jeu.
- Émet des événements de flotte pour les processus du serveur, notamment :
 - `SERVER_PROCESS_INVALID_PATH`: Les paramètres de lancement du processus du serveur de jeu n'étaient pas correctement configurés.
 - `SERVER_PROCESS_TERMINATED_UNHEALTHY`: Le processus du serveur de jeu n'a pas signalé de bilan de santé valide dans les 3 minutes suivant l'activation et a donc été interrompu.
 - `SERVER_PROCESS_FORCE_TERMINATED`: Le processus du serveur de jeu ne s'est pas correctement arrêté après avoir `OnProcessTerminate()` été envoyé dans les 30 secondes.
 - `SERVER_PROCESS_CRASHED`: Le processus d'un serveur de jeu s'est bloqué pour une raison ou une autre.

Gestion du calcul

- Reçoit des messages du Amazon GameLift Servers service pour arrêter le calcul.
- Demande de terminer le calcul par Amazon GameLift Servers.

Journalisation

- Télécharge les journaux dans un compartiment Amazon S3 de votre AWS compte.

Testez votre intégration à l'aide de Amazon GameLift Servers Local

Note

Cette rubrique couvre les tests pour les jeux intégrés au SDK du serveur pour Amazon GameLift Servers version 4.x ou antérieure uniquement. Le package SDK de votre serveur inclut une version compatible de Amazon GameLift Servers Local. Si vous utilisez la version 5.x du SDK du serveur, consultez [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#) les tests locaux avec un Amazon GameLift Servers. Une flotte n'importe où.

Utiliser Amazon GameLift Servers Local pour exécuter une version limitée du managé Amazon GameLift Servers service sur un appareil local et testez l'intégration de votre jeu par rapport à celui-ci. Cet outil est utile lors de l'exécution d'un développement itératif sur l'intégration de votre jeu. L'alternative consiste à télécharger chaque nouvelle version sur Amazon GameLift Servers et configurer une flotte pour héberger votre jeu, cela peut prendre plusieurs ou plus à chaque fois.

Avec Amazon GameLift Servers En local, vous pouvez vérifier les points suivants :

- Votre serveur de jeu est correctement intégré au SDK du serveur et communique correctement avec le Amazon GameLift Servers service permettant de démarrer de nouvelles sessions de jeu, d'accepter de nouveaux joueurs et de signaler l'état de santé et l'état de santé.
- Votre client de jeu est correctement intégré au AWS SDK pour Amazon GameLift Servers et est capable de récupérer des informations sur les sessions de jeu existantes, de démarrer de nouvelles sessions de jeu, de rejoindre des joueurs et de se connecter à la session de jeu.

Amazon GameLift Servers Local est un outil de ligne de commande qui démarre une version autonome du Amazon GameLift Servers service. Amazon GameLift Servers Local fournit également un journal des événements courants concernant l'initialisation des processus du serveur, les contrôles de santé, les appels d'API et les réponses. Amazon GameLift Servers Local reconnaît un sous-ensemble des actions du AWS SDK pour Amazon GameLift Servers. Vous pouvez passer des appels depuis AWS CLI ou depuis le client de jeu. Toutes les actions de l'API s'exécutent localement comme elles le font dans le Amazon GameLift Servers service Web.

Chaque processus serveur ne doit héberger qu'une seule session de jeu. La session de jeu est l'exécutable auquel vous vous connectez Amazon GameLift Servers Local. Lorsque la session de jeu est terminée, vous devez appeler `GameLiftServerSDK::ProcessEnding` puis quitter le processus. Lors d'un test local avec Amazon GameLift Servers Local, vous pouvez démarrer plusieurs processus serveur. Chaque processus se connectera à Amazon GameLift Servers Local. Vous pouvez ensuite créer une session de jeu pour chaque processus du serveur. Lorsque votre session de jeu se termine, le processus de votre serveur de jeu devrait se terminer. Vous devez ensuite démarrer manuellement un autre processus serveur.

Amazon GameLift Servers local prend en charge les éléments suivants APIs :

- `CreateGameSession`
- `CreatePlayerSession`
- `CreatePlayerSessions`

- DescribeGameSessions
- DescribePlayerSessions

Configuration Amazon GameLift Servers local

Amazon GameLift Servers Local est fourni sous forme de `.jar` fichier exécutable fourni avec le [SDK du serveur](#). Il peut être exécuté sous Windows ou Linux et utilisé avec n'importe quel Amazon GameLift Servers-langue prise en charge.

Avant d'exécuter Local, les éléments suivants doivent avoir été installés.

- Une version du SDK du serveur pour Amazon GameLift Servers versions 3.1.5 à 4.x.
- Java 8

Tester un serveur de jeu

Si vous souhaitez tester uniquement votre serveur de jeu, vous pouvez utiliser le AWS CLI pour simuler les appels des clients du jeu au Amazon GameLift Servers Service local. Vous pouvez ainsi vérifier que votre serveur de jeux se comporte comme prévu dans les circonstances suivantes :

- Le serveur de jeu démarre correctement et initialise le SDK du serveur pour Amazon GameLift Servers.
- Dans le cadre du processus de lancement, le serveur de jeu notifie Amazon GameLift Servers que le serveur est prêt à héberger des sessions de jeu.
- Le serveur de jeu envoie l'état de santé à Amazon GameLift Servers chaque minute pendant la course.
- Le serveur de jeux répond aux demandes de démarrage d'une nouvelle session de jeu.

1. Démarrer Amazon GameLift Servers Local.

Ouvrez une fenêtre d'invite de commande, accédez au répertoire contenant le fichier *GameLiftLocal.jar* et exécutez-le. Par défaut, Local écoute les demandes des clients de jeu sur le port 8080. Pour spécifier un autre numéro de port, utilisez le paramètre `-p`, comme indiqué dans l'exemple suivant:

```
java -jar GameLiftLocal.jar -p 9080
```

Une fois que Local a démarré, les journaux indiquent que deux serveurs locaux ont été lancés, l'un qui écoute votre serveur de jeux et l'autre qui écoute votre client de jeu ou l' AWS CLI. Les journaux continuent de signaler l'activité sur les deux serveurs locaux, y compris la communication vers et depuis les composants de votre jeu.

2. Démarrez votre serveur de jeux.

Démarrez votre Amazon GameLift Servers-serveur de jeu intégré localement. Il n'est pas nécessaire de modifier le point de terminaison du serveur de jeux.

Dans la fenêtre d'invite de commande locale, les messages du journal indiquent que votre serveur de jeu s'est connecté au Amazon GameLift Servers Service local. Cela signifie que votre serveur de jeu a correctement initialisé le SDK du serveur pour Amazon GameLift Servers (avec `InitSDK()`). Il a appelé `ProcessReady()` avec les chemins d'accès aux journaux affichés et, en cas de succès, est prêt à héberger une session de jeu. Pendant que le serveur de jeu fonctionne, Amazon GameLift Servers enregistre chaque rapport d'état de santé du serveur de jeu. L'exemple de message de journal suivant illustre l'intégration réussie d'un serveur de jeux :

```
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK
connected: /127.0.0.1:64247
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - SDK pid is 17040,
sdkVersion is 3.1.5 and sdkLanguage is CSharp
16:50:53,217 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - NOTE: Only SDK
versions 3.1.5 and above are supported in GameLiftLocal!
16:50:53,451 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
received from: /127.0.0.1:64247 and ackRequest requested? true
16:50:53,543 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onProcessReady
data: logPathsToUpload: "C:\\game\\logs"
logPathsToUpload: "C:\\game\\error"
port: 1935

16:50:53,544 INFO || - [HostProcessManager] nioEventLoopGroup-3-1 - Registered new
process true, true,
16:50:53,558 INFO || - [SDKListenerImpl] nioEventLoopGroup-3-1 - onReportHealth
received from /127.0.0.1:64247 with health status: healthy
```

Les messages éventuels d'erreur ou d'avertissement sont les suivants :

- Erreur : « Impossible ProcessReady de trouver un processus avec le PiD : *<process ID>* ! InitSDK () a-t-il été invoqué ? »
- Avertissement : « L'état du processus existe déjà pour le processus dont le PID est : *<process ID>* ! Est-ce que ProcessReady (...) est invoqué plusieurs fois ? »

3. Démarrez le AWS CLI.

Une fois que votre serveur de jeux a appelé ProcessReady() avec succès, vous pouvez démarrer les appels clients. Ouvrez une autre fenêtre d'invite de commande et démarrez l' AWS CLI . AWS CLI Par défaut, utilise le Amazon GameLift Servers point de terminaison du service Web. Vous devez remplacer celui-ci par le point de terminaison Local dans toutes les demandes à l'aide du paramètre `--endpoint-url`, comme indiqué dans l'exemple de demande suivant.

```
AWS gamelift describe-game-sessions --endpoint-url http://localhost:9080 --fleet-id fleet-123
```

Dans la fenêtre d'invite de commande, `AWS gamelift` les commandes donnent lieu à des réponses, comme indiqué dans le manuel de [référence des AWS CLI commandes](#).

4. Créez une session de jeu.

Avec le AWS CLI, soumettez une demande [CreateGameSession\(\)](#). La demande doit respecter la syntaxe attendue. Pour Local, le paramètre FleetId peut être défini avec toute chaîne valide (`^fleet-\S+`).

```
AWS gamelift create-game-session --endpoint-url http://localhost:9080 --maximum-player-session-count 2 --fleet-id fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
```

Dans la fenêtre d'invite de commande locale, les messages du journal indiquent que Amazon GameLift Servers Local a envoyé un `onStartGameSession` rappel à votre serveur de jeu. Si une session de jeu a été créée avec succès, votre serveur de jeux répond en appelant `ActivateGameSession`.

```
13:57:36,129 INFO || - [SDKInvokerImpl]
      Thread-2 - Finished sending event to game server to start a game session:
      arn:aws:gamelift:local::gamesession/
      fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
      aea2-54b3fa0818b6.
```

```

Waiting for ack response.13:57:36,143 INFO || - [SDKInvokerImpl]
Thread-2 - Received ack response: true13:57:36,144 INFO || -
[CreateGameSessionDispatcher] Thread-2 - GameSession with id:
arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-ab423a4b-b827-4765-
aea2-54b3fa0818b6
created13:57:36,227 INFO || - [SDKListenerImpl]
nioEventLoopGroup-3-1 - onGameSessionActivate received
from: /127.0.0.1:60020 and ackRequest
requested? true13:57:36,230 INFO || - [SDKListenerImpl]
nioEventLoopGroup-3-1 - onGameSessionActivate data: gameId:
"arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"

```

Dans la AWS CLI fenêtre, Amazon GameLift Servers répond avec un objet de session de jeu comprenant un identifiant de session de jeu. Notez que le statut de la nouvelle session de jeu est **Activating**. Le statut passe à **Actif** une fois que votre serveur de jeu l'appelle. `ActivateGameSession` Si vous souhaitez voir le statut modifié, utilisez le AWS CLI `callDescribeGameSessions()`.

```

{
  "GameSession": {
    "Status": "ACTIVATING",
    "MaximumPlayerSessionCount": 2,
    "FleetId": "fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d",
    "GameSessionId": "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890",
    "IpAddress": "127.0.0.1",
    "Port": 1935
  }
}

```

Tester un serveur et un client de jeu

Pour vérifier l'intégration complète de votre jeu, y compris la connexion de joueurs à des parties, vous pouvez exécuter votre serveur de jeux et votre client de jeu localement. Cela vous permet de tester les appels programmatiques de votre client de jeu vers le Amazon GameLift Servers Local. Vous pouvez vérifier les actions suivantes :

- Le client du jeu envoie avec succès des demandes de AWS SDK au Amazon GameLift Servers Service local, notamment pour créer des sessions de jeu, récupérer des informations sur les sessions de jeu existantes et créer des sessions pour les joueurs.
- Le serveur de jeux valide correctement les joueurs lorsqu'ils tentent de rejoindre une session de jeu. Pour les joueurs validés, le serveur de jeux peut récupérer les données des joueurs (si la fonction a été mise en œuvre).
- Le serveur de jeux signale l'abandon d'une connexion quand un joueur quitte le jeu.
- Le serveur de jeux rapporte la fin d'une session de jeu.

1. Démarrer Amazon GameLift Servers Local.

Ouvrez une fenêtre d'invite de commande, accédez au répertoire contenant le fichier *GameLiftLocal.jar* et exécutez-le. Par défaut, Local écoute les demandes des clients de jeu sur le port 8080. Pour spécifier un autre numéro de port, utilisez le paramètre `-p`, comme indiqué dans l'exemple suivant.

```
./gamelift-local -p 9080
```

Une fois que Local a démarré, vous voyez que les journaux affichent le lancement de deux serveurs locaux, l'un qui écoute votre serveur de jeux et l'autre qui écoute votre client de jeu ou l'AWS CLI.

2. Démarrez votre serveur de jeux.

Démarrez votre Amazon GameLift Servers-serveur de jeu intégré localement. Pour plus d'informations sur les journaux de messages, consultez [Tester un serveur de jeu](#).

3. Configurez votre client de jeu pour Local et démarrez-le.

Pour utiliser votre client de jeu avec Amazon GameLift Servers Service local, vous devez apporter les modifications suivantes à la configuration de votre client de jeu, comme décrit dans [Configuration Amazon GameLift Servers sur un service principal](#) :

- Modifiez l'objet `ClientConfiguration` afin de pointer vers votre point de terminaison Local, tel que `http://localhost:9080`.
- Définissez une valeur d'ID de flotte cible. Pour Local, vous n'avez pas besoin d'un ID de flotte réel ; définissez la flotte cible avec une chaîne valide de votre choix (`^fleet-\S+`), comme `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`.

- Définissez les AWS informations d'identification. Pour Local, vous n'avez pas besoin d'informations d'identification AWS réelles ; vous pouvez définir la clé d'accès et la clé secrète avec toute chaîne de votre choix.

Dans la fenêtre d'invite de commande locale, une fois que vous avez démarré le client du jeu, les messages du journal doivent indiquer qu'il a initialisé le `GameLiftClient` et qu'il a correctement communiqué avec le Amazon GameLift Servers service.

4. Testez les appels des clients du jeu vers Amazon GameLift Servers service.

Vérifiez que votre client de jeu effectue correctement tout ou partie des appels d'API suivants :

- [`CreateGameSession\(\)`](#)
- [`DescribeGameSessions\(\)`](#)
- [`CreatePlayerSession\(\)`](#)
- [`CreatePlayerSessions\(\)`](#)
- [`DescribePlayerSessions\(\)`](#)

Dans la fenêtre d'invite de commande Local, seuls les appels à `CreateGameSession()` se traduisent par des messages de journaux. Les messages du journal indiquent quand Amazon GameLift Servers Local invite votre serveur de jeu à démarrer une session de jeu (`onStartGameSession`rappel) et obtient un résultat `ActivateGameSession` lorsque votre serveur de jeu l'invoque. Dans la fenêtre de l' AWS CLI , tous les appels d'API entraînent des réponses ou des messages d'erreur tels que documentés.

5. Vérifiez que votre serveur de jeux valide les nouvelles connexions de joueur.

Après avoir créé une session de jeu et une session joueur, établissez une connexion directe à la session de jeu.

Dans la fenêtre d'invite de commande Local, les messages de journaux doivent indiquer que le serveur de jeux a envoyé une demande `AcceptPlayerSession()` pour valider la nouvelle connexion de joueur. Si vous utilisez le AWS CLI pour appeler `DescribePlayerSessions()`, le statut de la session du joueur devrait passer de `Réservée` à `Actif`.

6. Vérifiez que votre serveur de jeu communique l'état du jeu et du joueur au Amazon GameLift Servers service.

Dans Amazon GameLift Servers pour gérer la demande des joueurs et communiquer correctement les statistiques, votre serveur de jeu doit signaler les différents statuts à Amazon GameLift Servers. Vérifiez que Local enregistre les événements liés aux actions suivantes. Vous pouvez également utiliser le pour AWS CLI suivre les changements de statut.

- Le joueur se déconnecte d'une session de jeu — Amazon GameLift Servers Les messages du journal local devraient indiquer que votre serveur de jeu appelle `RemovePlayerSession()`. Un appel de l' AWS CLI à `DescribePlayerSessions()` doit refléter un changement d'état d'Active en Completed. Vous pouvez également appeler `DescribeGameSessions()` pour vérifier que le nombre de joueurs en cours de la session a diminué d'une unité.
- Fin de la session de jeu — Amazon GameLift Servers Les messages du journal local devraient indiquer que votre serveur de jeu appelle `TerminateGameSession()`.

 Note

Le conseil précédent était d'appeler à `TerminateGameSession()` la fin d'une session de jeu. Cette méthode est obsolète avec Amazon GameLift Servers SDK pour serveur v4.0.1. Consultez [Mettre fin à une session de jeu](#).

- Le processus du serveur est terminé — Amazon GameLift Servers Les messages du journal local devraient indiquer que votre serveur de jeu appelle `ProcessEnding()`. Un AWS CLI appel à `DescribeGameSessions()` doit refléter un changement de statut de Active à Terminated (ou Terminating).

Variations avec les produits locaux

Lors de l'utilisation Amazon GameLift Servers Local, gardez à l'esprit les points suivants :

- Contrairement au Amazon GameLift Servers service Web, Local ne suit pas l'état de santé d'un serveur et ne lance pas le `onProcessTerminate` rappel. Local arrête simplement la journalisation des rapports d'état pour le serveur de jeux.
- Pour les appels au AWS SDK, le parc IDs n'est pas validé et peut être n'importe quelle valeur de chaîne répondant aux exigences des paramètres (`^fleet-\S+`).
- Les sessions de jeu IDs créées avec Local ont une structure différente. Ils incluent la chaîne `local`, comme illustré ici :

```
arn:aws:gamelift:local::gamesession/fleet-123/gsess-56961f8e-  
db9c-4173-97e7-270b82f0daa6
```

Ajouter le FlexMatch matchmaking

Amazon GameLift Servers FlexMatch À utiliser pour ajouter une fonctionnalité de matchmaking des joueurs à vos jeux Amazon GameLift Servers hébergés. Vous pouvez l'utiliser FlexMatch avec des serveurs de jeu personnalisés ou Amazon GameLift Servers Realtime.

FlexMatch associe le service de mise en relation avec un moteur de règles personnalisables. Vous concevez la manière de faire correspondre les joueurs en fonction des attributs des joueurs et des modes de jeu adaptés à votre jeu. FlexMatch gère les détails de l'évaluation des joueurs à la recherche d'une partie, de l'organisation de matchs avec une ou plusieurs équipes et du lancement de sessions de jeu pour organiser les matchs.

Pour utiliser le FlexMatch service complet, vous devez configurer vos ressources d'hébergement avec des files d'attente. Amazon GameLift Servers utilise des files d'attente pour trouver les meilleurs emplacements d'hébergement possibles pour les jeux dans plusieurs régions et types d'ordinateurs. En particulier, les Amazon GameLift Servers files d'attente peuvent utiliser les données de latence, lorsqu'elles sont fournies par les clients du jeu, pour organiser des sessions de jeu afin que les joueurs aient le moins de latence possible lorsqu'ils jouent.

Pour plus d'informations sur l'FlexMatch inclusion d'une aide détaillée sur l'intégration du matchmaking dans vos jeux, consultez les rubriques suivantes [Amazon GameLift Servers FlexMatch du guide du développeur](#) :

- [Comment Amazon GameLift Servers FlexMatch fonctionne](#)
- [FlexMatch étapes d'intégration](#)

Obtenir des données de flotte pour une Amazon GameLift Servers instance

Dans certaines situations, votre build ou Amazon GameLift Servers Realtime script de jeu personnalisé peut nécessiter des informations sur la Amazon GameLift Servers flotte. Par exemple, le build ou le script de votre jeu peut inclure du code pour :

- Surveillez l'activité en fonction des données de la flotte.
- Cumulez les indicateurs pour suivre l'activité par données de flotte. (De nombreux jeux utilisent ces données pour LiveOps leurs activités.)
- Fournissez des données pertinentes aux services de jeux personnalisés, par exemple pour le matchmaking, le dimensionnement des capacités supplémentaires ou les tests.

Les informations sur le parc sont disponibles sous forme de fichier JSON sur chaque instance aux emplacements suivants :

- Windows: C:\GameMetadata\gamelift-metadata.json
- Linux : /local/gamemetadata/gamelift-metadata.json

Le `gamelift-metadata.json` fichier inclut les [attributs d'une ressource de Amazon GameLift Servers flotte](#).

Exemple de fichier JSON :

```
{
  "buildArn": "arn:aws:gamelift:us-west-2:123456789012:build/build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "buildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",
  "fleetArn": "arn:aws:gamelift:us-west-2:123456789012:fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "fleetDescription": "Test fleet for Really Fun Game v0.8",
  "fleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "name": "ReallyFunGameTestFleet08",
  "fleetType": "ON_DEMAND",
  "instanceRoleArn": "arn:aws:iam::123456789012:role/S3AccessForGameLift",
  "instanceType": "c5.large",
  "serverLaunchPath": "/local/game/reallyfungame.exe"
}
```

Intégrer des jeux avec Amazon GameLift ServersRealtime

Cette rubrique fournit une vue d'ensemble de la Amazon GameLift Servers Realtime solution gérée Amazon GameLift Servers avec. La présentation explique dans quels cas cette solution convient à votre jeu et comment elle est Amazon GameLift Servers Realtime compatible avec le jeu multijoueur.

 Tip

[Commencez dès maintenant avec Amazon GameLift Servers](#)

Que sont les Realtime serveurs ?

Realtime les serveurs sont des serveurs de ready-to-go jeu Amazon GameLift Servers légers que vous pouvez utiliser avec vos jeux multijoueurs. Realtime les serveurs suppriment le processus de développement, de test et de déploiement d'un serveur de jeu personnalisé. Cette solution peut vous aider à minimiser le temps et les efforts nécessaires pour terminer votre jeu.

Fonctions principales

- Stack réseau complet pour l'interaction entre le client et le serveur du jeu
- Fonctionnalités de base du serveur de jeu
- Logique de serveur personnalisable
- Mises à jour en direct des Realtime configurations et de la logique du serveur
- FlexMatchmaking
- Contrôle flexible des ressources d'hébergement

Configurez Realtime les serveurs en créant une flotte et en fournissant un script de configuration.

Comment Amazon GameLift ServersRealtime gère les sessions de jeu

Vous pouvez ajouter une logique personnalisée pour la gestion des sessions de jeu en l'intégrant au Realtime script. Vous pouvez écrire du code pour accéder à des objets spécifiques au serveur, ajouter une logique axée sur les événements à l'aide de rappels ou ajouter une logique basée sur des scénarios autres que les événements.

Comment Realtime les clients et les serveurs interagissent

Au cours d'une session de jeu, les clients du jeu interagissent en envoyant des messages au Realtime serveur via un service principal. Le service principal transmet ensuite les messages aux clients du jeu pour échanger l'activité, l'état du jeu et les données de jeu pertinentes.

En outre, vous pouvez personnaliser la façon dont les clients et les serveurs interagissent en ajoutant une logique de jeu au script Realtime. Avec une logique de jeu personnalisée, un Realtime serveur peut implémenter des rappels pour démarrer des réponses pilotées par des événements.

Protocole de communication

Realtime les serveurs et les clients de jeu connectés communiquent via deux canaux : une connexion TCP pour une diffusion fiable et un canal UDP pour une diffusion rapide. Lorsque vous créez des messages, les clients de jeu choisissent le protocole à utiliser en fonction de la nature du message. La livraison de message est définie sur UDP par défaut. Si aucun canal UDP n'est disponible, Amazon GameLift Servers envoie des messages en utilisant le protocole TCP comme solution de secours.

Contenu des messages

Le contenu des messages se compose de deux éléments : une opération de code obligatoire (opCode) et une charge utile facultative. L'OPcode d'un message identifie une activité du joueur ou un événement de jeu en particulier, et la charge utile fournit des données supplémentaires relatives au code d'opération. Ces deux éléments sont définis par le développeur. Votre client de jeu agit en fonction des OPcodes contenus dans les messages qu'il reçoit.

Groupes de joueurs

Amazon GameLift Servers Realtime fournit des fonctionnalités permettant de gérer des groupes de joueurs. Par défaut, Amazon GameLift Servers place tous les joueurs connectés à un jeu dans un groupe « tous les joueurs ». De plus, les développeurs peuvent configurer des groupes supplémentaires pour leurs jeux et les joueurs peuvent être membres de plusieurs groupes à la fois. Les membres du groupe peuvent envoyer des messages et partager des données de jeu avec tous les joueurs du groupe. Une utilisation possible pour les groupes consiste à configurer les équipes de joueurs et à gérer la communication de l'équipe.

Amazon GameLift Servers Realtime avec certificats TLS

Avec Amazon GameLift Servers Realtime, l'authentification du serveur et le chiffrement des paquets de données sont intégrés au service. Ces fonctionnalités de sécurité sont activées lorsque vous activez la génération de certificats TLS. Lorsqu'un client de jeu essaie de se connecter à un Realtime serveur, le serveur répond automatiquement avec le certificat TLS, que le client valide. Amazon GameLift Servers gère le chiffrement à l'aide du protocole TLS pour les communications TCP (WebSockets) et du protocole DTLS pour le trafic UDP.

Personnalisation d'un serveur Realtime

Un Realtime serveur fonctionne comme un serveur relais sans état. Le Realtime serveur transmet des paquets de messages et de données de jeu entre les clients du jeu connectés au jeu.

Cependant, le Realtime serveur n'évalue pas les messages, ne traite pas les données et n'exécute aucune logique de jeu. Ainsi utilisé, chaque client de jeu conserve sa propre vision de l'état du jeu et fournit des mises à jour aux autres joueurs via le serveur relais. Chaque client de jeu est responsable de l'intégration de ces mises à jour et de la conciliation de son propre état de jeu.

Vous pouvez personnaliser vos serveurs en ajoutant des fonctionnalités de Realtime script. Avec la logique du jeu, par exemple, vous pouvez créer un jeu dynamique avec une vue de l'état du jeu faisant autorité sur le serveur.

Amazon GameLift Servers définit un ensemble de rappels côté serveur pour les scripts. Realtime Implémentez ces rappels pour ajouter des fonctionnalités pilotées par les événements à votre serveur. Par exemple, vous pouvez :

- Authentifier un joueur lorsqu'un client de jeu essaie de se connecter au serveur.
- Validez si un joueur peut rejoindre un groupe sur demande.
- Déterminez quand envoyer les messages d'un joueur donné ou à un joueur cible, ou effectuez un traitement supplémentaire en réponse.
- Avertissez tous les joueurs lorsqu'un joueur quitte un groupe ou se déconnecte du serveur.
- Affichez le contenu des objets de session de jeu ou des objets de message, et utilisez les données.

Déploiement et mise à jour Amazon GameLift Servers Realtime

L'un des principaux avantages Amazon GameLift Servers Realtime est la possibilité de mettre à jour vos scripts à tout moment. Lorsque vous mettez à jour un script, la nouvelle version est Amazon GameLift Servers distribuée à toutes les ressources d'hébergement en quelques minutes. Une fois le nouveau script Amazon GameLift Servers déployé, toutes les nouvelles sessions de jeu créées par la suite utiliseront la nouvelle version du script. (Les sessions de jeu existantes continueront à utiliser la version originale.)

Commencez à intégrer votre jeu avec Amazon GameLift Servers Realtime :

- [Intégration d'un client de jeu pour Amazon GameLift Servers Realtime](#)
- [Création d'un Realtime script](#)

Intégration d'un client de jeu pour Amazon GameLift ServersRealtime

Cette rubrique décrit comment préparer votre client de jeu à rejoindre et participer à des sessions de jeu hébergées par Amazon GameLift Servers.

Il existe deux ensembles de tâches nécessaires pour préparer votre client de jeu :

- Configurer votre client de jeu pour obtenir plus d'informations sur des jeux existants, demander la mise en relation, démarrer de nouvelles sessions de jeu et réserver des emplacements de session de jeu pour un joueur.
- Permettez à votre client de jeu de rejoindre une session de jeu hébergée sur un serveur Realtime et d'échanger des messages.

Trouvez ou créez des sessions de jeu et des sessions de joueurs

Configurez votre client de jeu pour trouver ou démarrer des sessions de jeu, demander des mises en relation FlexMatch et réserver de l'espace pour les joueurs dans un jeu en créant des sessions de joueur. Il est recommandé de créer un service principal et de l'utiliser pour envoyer les demandes directes au Amazon GameLift Servers service lorsqu'une action du client du jeu est déclenchée par une action du client du jeu. Le service principal transmet ensuite les réponses pertinentes au client du jeu.

1. Ajoutez le AWS SDK à votre client de jeu, initialisez un Amazon GameLift Servers client et configurez-le pour utiliser les ressources d'hébergement de vos flottes et de vos files d'attente. Le AWS SDK est disponible en plusieurs langues ; consultez le Amazon GameLift Servers SDKs [Pour les services clients liés aux jeux](#).
2. Ajoutez des Amazon GameLift Servers fonctionnalités à votre service principal. Pour des instructions plus détaillées, voir [Ajoutez Amazon GameLift Servers à votre client de jeu](#) et [Ajouter le FlexMatch matchmaking](#). Une bonne pratique consiste à utiliser des placements de session de jeu pour créer de nouvelles sessions de jeu. Cette méthode vous permet de tirer pleinement parti de la Amazon GameLift Servers possibilité de placer rapidement et intelligemment de nouvelles sessions de jeu, ainsi que d'utiliser les données de latence des joueurs pour minimiser le décalage de jeu. Au minimum, votre service principal doit être en mesure de demander de nouvelles sessions de jeu et de gérer les données des sessions de jeu en réponse. Vous pouvez également ajouter des fonctionnalités pour rechercher et obtenir des informations sur les sessions de jeu existantes, et demander des sessions de joueur qui réservent de façon efficace un emplacement de joueur dans une session de jeu existante.

3. Renvoyer des informations de connexion vers le client de jeu. Le service principal reçoit les objets de session de jeu et de session de joueur en réponse aux demandes adressées au Amazon GameLift Servers service. Ces objets contiennent des informations, notamment les détails de connexion (adresse IP et port) et l'ID de session du joueur, dont le client de jeu a besoin pour se connecter à la session de jeu en cours sur un serveur en temps réel.

Connectez-vous aux jeux sur Amazon GameLift ServersRealtime

Autorisez votre client de jeu à se connecter directement à une session de jeu hébergée sur un serveur Realtime et d'échanger des messages avec le serveur et d'autres joueurs.

1. Procurez-vous le SDK client pour Amazon GameLift ServersRealtime, créez-le et ajoutez-le à votre projet client de jeu. Consultez le fichier README pour plus d'informations sur les exigences du kit SDK et des instructions sur la façon de créer les bibliothèques client.
2. Appelez [Client\(\) \(C#\)](#) avec une configuration client qui spécifie le type de client/serveur connexion à utiliser.

Note

Si vous vous connectez à un serveur en temps réel qui s'exécute sur une flotte sécurisée avec un certificat TLS, vous devez spécifier un type de connexion sécurisé.

3. Ajoutez les fonctionnalités suivantes à votre client de jeu. Consultez la [Amazon GameLift ServersRealtime référence de l'API client \(C#\)](#) pour plus d'informations.
 - Se connecter et se déconnecter d'un jeu
 - [Connect\(\)](#)
 - [Disconnect\(\)](#)
 - Envoyer des messages à des destinataires cibles
 - [SendMessage\(\)](#)
 - Recevoir et traiter des messages
 - [OnDataReceived\(\)](#)
 - Rejoindre des groupes et quitter des groupes de joueurs
 - [JoinGroup\(\)](#)
 - [RequestGroupMembership\(\)](#)
 - [LeaveGroup\(\)](#)

4. Configuration des gestionnaires d'événements pour les rappels de client en fonction de vos besoins. Consultez [Amazon GameLift ServersRealtime](#)[référence de l'API client \(C#\) : rappels asynchrones](#).

Lorsque vous travaillez avec des flottes en temps réel pour lesquelles la génération de certificats TLS est activée, le serveur est automatiquement authentifié à l'aide du certificat TLS. Le trafic TCP et UDP est chiffré en transit pour assurer la sécurité de la couche de transport. Le trafic TCP est chiffré avec TLS 1.2 et le trafic UDP avec DTLS 1.2.

Exemples de clients de jeu

Client en temps réel de base (C#)

Cet exemple illustre une intégration de base d'un client de jeu avec le SDK client (C#) pour Amazon GameLift Servers Realtime. Comme cela est illustré, l'exemple initialise un objet de client Realtime, configure des gestionnaires d'événements et met en œuvre les rappels côté client, se connecte à un serveur Realtime, envoie un message et se déconnecte.

```
using System;
using System.Text;
using Aws.GameLift.Realtime;
using Aws.GameLift.Realtime.Event;
using Aws.GameLift.Realtime.Types;

namespace Example
{
    /**
     * An example client that wraps the client SDK for Amazon GameLift Servers Realtime
     *
     * You can redirect logging from the SDK by setting up the LogHandler as such:
     * ClientLogger.LogHandler = (x) => Console.WriteLine(x);
     *
     */
    class RealTimeClient
    {
        public Aws.GameLift.Realtime.Client Client { get; private set; }

        // An opcode defined by client and your server script that represents a custom
        message type
        private const int MY_TEST_OP_CODE = 10;
    }
}
```

```

    /// Initialize a client for Amazon GameLift Servers Realtime and connect to a
    player session.
    /// <param name="endpoint">The DNS name that is assigned to Realtime server</
param>
    /// <param name="remoteTcpPort">A TCP port for the Realtime server</param>
    /// <param name="listeningUdpPort">A local port for listening to UDP traffic</
param>
    /// <param name="connectionType">Type of connection to establish between client
    and the Realtime server</param>
    /// <param name="playerSessionId">The player session ID that is assigned to the
    game client for a game session </param>
    /// <param name="connectionPayload">Developer-defined data to be used during
    client connection, such as for player authentication</param>
    public RealTimeClient(string endpoint, int remoteTcpPort, int listeningUdpPort,
    ConnectionType connectionType,
        string playerSessionId, byte[] connectionPayload)
    {
        // Create a client configuration to specify a secure or unsecure connection
    type
        // Best practice is to set up a secure connection using the connection type
    RT_OVER_WSS_DTLS_TLS12.
        ClientConfiguration clientConfiguration = new ClientConfiguration()
    {
        // C# notation to set the field ConnectionType in the new instance of
    ClientConfiguration
        ConnectionType = connectionType
    };

        // Create a Realtime client with the client configuration
        Client = new Client(clientConfiguration);

        // Initialize event handlers for the Realtime client
        Client.ConnectionOpen += OnOpenEvent;
        Client.ConnectionClose += OnCloseEvent;
        Client.GroupMembershipUpdated += OnGroupMembershipUpdate;
        Client.DataReceived += OnDataReceived;

        // Create a connection token to authenticate the client with the Realtime
    server
        // Player session IDs can be retrieved using AWS SDK for Amazon GameLift
    Servers
        ConnectionToken connectionToken = new ConnectionToken(playerSessionId,
    connectionPayload);

```

```

        // Initiate a connection with the Realtime server with the given connection
information
        Client.Connect(endpoint, remoteTcpPort, listeningUdpPort, connectionToken);
    }

    public void Disconnect()
    {
        if (Client.Connected)
        {
            Client.Disconnect();
        }
    }

    public bool IsConnected()
    {
        return Client.Connected;
    }

    /// <summary>
    /// Example of sending to a custom message to the server.
    ///
    /// Server could be replaced by known peer Id etc.
    /// </summary>
    /// <param name="intent">Choice of delivery intent i.e. Reliable, Fast etc. </
param>
    /// <param name="payload">Custom payload to send with message</param>
    public void SendMessage(DeliveryIntent intent, string payload)
    {
        Client.SendMessage(Client.NewMessage(MY_TEST_OP_CODE)
            .WithDeliveryIntent(intent)
            .WithTargetPlayer(Constants.PLAYER_ID_SERVER)
            .WithPayload(StringToBytes(payload)));
    }

    /**
     * Handle connection open events
     */
    public void OnOpenEvent(object sender, EventArgs e)
    {
    }

    /**
     * Handle connection close events
     */

```

```
public void OnCloseEvent(object sender, EventArgs e)
{
}

/**
 * Handle Group membership update events
 */
public void OnGroupMembershipUpdate(object sender, GroupMembershipEventArgs e)
{
}

/**
 * Handle data received from the Realtime server
 */
public virtual void OnDataReceived(object sender, DataReceivedEventArgs e)
{
    switch (e.OpCode)
    {
        // handle message based on OpCode
        default:
            break;
    }
}

/**
 * Helper method to simplify task of sending/receiving payloads.
 */
public static byte[] StringToBytes(string str)
{
    return Encoding.UTF8.GetBytes(str);
}

/**
 * Helper method to simplify task of sending/receiving payloads.
 */
public static string BytesToString(byte[] bytes)
{
    return Encoding.UTF8.GetString(bytes);
}
}
```

Création d'un Realtime script

Amazon GameLift ServersRealtimePour l'utiliser dans votre jeu, vous devez fournir un script (sous forme de JavaScript code) pour configurer et éventuellement personnaliser une flotte de Amazon GameLift ServersRealtime. Cette rubrique explique les principales étapes de la création d'un script Realtime. Une fois que le script est prêt, chargez-le dans le service Amazon GameLift Servers et utilisez-le pour créer une flotte (voir [Déployez un script pour Amazon GameLift ServersRealtime](#)).

Pour préparer un script à utiliser avec Amazon GameLift ServersRealtime, ajoutez les fonctionnalités suivantes à votre Realtime script.

Gérer le cycle de vie des sessions de jeu (obligatoire)

Au minimum, un script Realtime doit inclure la fonction `Init()`, qui prépare le serveur Realtime pour commencer une session de jeu. Il est également vivement recommandé de fournir un moyen de mettre fin à des sessions de jeu, afin de garantir que de nouvelles sessions de jeu pourront être encore démarrées sur votre flotte.

La fonction de rappel `Init()`, lorsqu'elle est appelée, se voit transmettre un objet de session Realtime, qui contient une interface pour le serveur Realtime. Veuillez consulter [Amazon GameLift ServersRealtimeinterface](#) pour plus de détails sur cette interface.

Pour mettre fin élégamment à une session de jeu, le script doit également appeler la fonction `session.processEnding` du serveur Realtime. Cela requiert un mécanisme pour déterminer quand mettre fin à une session. L'exemple de code de script illustre un mécanisme simple qui recherche les connexions de joueur et déclenche l'arrêt de la session de jeu quand aucun joueur ne s'est connecté à la session depuis un temps spécifié.

Amazon GameLift ServersRealtime dotés de la configuration la plus élémentaire (initialisation et terminaison des processus du serveur), agissent essentiellement comme des serveurs relais apatrides. Le serveur Realtime relaie les messages et les données de jeu entre les clients de jeu connectés au jeu, mais ne prend aucune mesure indépendante pour traiter les données ou exécuter la logique. Vous pouvez également ajouter une logique de jeu, déclenchée par des événements de jeu ou d'autres mécanismes, en fonction des besoins de votre jeu.

Ajouter une logique de jeu côté serveur (facultatif)

Si vous le souhaitez, vous pouvez ajouter une logique de jeu à votre script Realtime. Vous pouvez, par exemple, effectuer une partie ou l'ensemble des opérations suivantes. L'exemple de code de script fournit une illustration. Consultez [Référence de script pour Amazon GameLift ServersRealtime](#).

- Ajoutez une logique pilotée par les événements. Implémentez les fonctions de rappel pour répondre aux événements client-serveur. Veuillez consulter [Rappels de script pour Amazon GameLift ServersRealtime](#) pour obtenir la liste complète des rappels.
- Déclenchez la logique en envoyant des messages au serveur. Créez un ensemble de codes d'opération spéciaux pour les messages envoyés au serveur à partir des clients de jeu et ajoutez des fonctions pour gérer leur réception. Utilisez le rappel `onMessage` et analysez le contenu du message à l'aide de l'interface `gameMessage` (voir [gameMessage.opcode](#)).
- Activez la logique du jeu pour accéder à vos autres AWS ressources. Pour en savoir plus, consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
- Autorisez Game Logic à accéder aux informations de la flotte pour l'instance sur laquelle il s'exécute. Pour en savoir plus, consultez [Obtenir des données de flotte pour une Amazon GameLift Servers instance](#).

Amazon GameLift ServersRealtimeexemple de script

Cet exemple illustre un script de base nécessaire au déploiement Amazon GameLift Servers Realtime ainsi qu'une logique personnalisée. Il contient la fonction `Init()` requise et utilise une minuterie pour déclencher l'arrêt de la session de jeu en fonction du temps écoulé sans connexion de joueur. Il inclut également des hooks pour une logique personnalisée, y compris certaines implémentations de rappel.

```
// Example Realtime Server Script
'use strict';

// Example override configuration
const configuration = {
  pingIntervalTime: 30000,
  maxPlayers: 32
};

// Timing mechanism used to trigger end of game session. Defines how long, in
// milliseconds, between each tick in the example tick loop
const tickTime = 1000;

// Defines how to long to wait in Seconds before beginning early termination check in
// the example tick loop
const minimumElapsedTime = 120;

var session; // The Realtime server session object
```

```
var logger; // Log at appropriate level
  via .info(), .warn(), .error(), .debug()
var startTime; // Records the time the process started
var activePlayers = 0; // Records the number of connected players
var onProcessStartedCalled = false; // Record if onProcessStarted has been called

// Example custom op codes for user-defined messages
// Any positive op code number can be defined here. These should match your client
  code.
const OP_CODE_CUSTOM_OP1 = 111;
const OP_CODE_CUSTOM_OP1_REPLY = 112;
const OP_CODE_PLAYER_ACCEPTED = 113;
const OP_CODE_DISCONNECT_NOTIFICATION = 114;

// Example groups for user-defined groups
// Any positive group number can be defined here. These should match your client code.
// When referring to user-defined groups, "-1" represents all groups, "0" is reserved.
const RED_TEAM_GROUP = 1;
const BLUE_TEAM_GROUP = 2;

// Called when game server is initialized, passed server's object of current session
function init(rtSession) {
  session = rtSession;
  logger = session.getLogger();
}

// On Process Started is called when the process has begun and we need to perform any
// bootstrapping. This is where the developer should insert any code to prepare
// the process to be able to host a game session, for example load some settings or set
  state
//
// Return true if the process has been appropriately prepared and it is okay to invoke
  the
// GameLift ProcessReady() call.
function onProcessStarted(args) {
  onProcessStartedCalled = true;
  logger.info("Starting process with args: " + args);
  logger.info("Ready to host games...");

  return true;
}

// Called when a new game session is started on the process
function onStartGameSession(gameSession) {
```

```
// Complete any game session set-up

// Set up an example tick loop to perform server initiated actions
startTime = getTimeInS();
tickLoop();
}

// Handle process termination if the process is being terminated by Amazon GameLift
Servers
// You do not need to call ProcessEnding here
function onProcessTerminate() {
    // Perform any clean up
}

// Return true if the process is healthy
function onHealthCheck() {
    return true;
}

// On Player Connect is called when a player has passed initial validation
// Return true if player should connect, false to reject
function onPlayerConnect(connectMsg) {
    // Perform any validation needed for connectMsg.payload, connectMsg.peerId
    return true;
}

// Called when a Player is accepted into the game
function onPlayerAccepted(player) {
    // This player was accepted -- let's send them a message
    const msg = session.newTextGameMessage(OP_CODE_PLAYER_ACCEPTED, player.peerId,
                                           "Peer " + player.peerId + " accepted");
    session.sendReliableMessage(msg, player.peerId);
    activePlayers++;
}

// On Player Disconnect is called when a player has left or been forcibly terminated
// Is only called for players that actually connected to the server and not those
// rejected by validation
// This is called before the player is removed from the player list
function onPlayerDisconnect(peerId) {
    // send a message to each remaining player letting them know about the disconnect
    const outMessage = session.newTextGameMessage(OP_CODE_DISCONNECT_NOTIFICATION,
                                                    session.getServerId(),
                                                    "Peer " + peerId + " disconnected");
```



```

    session.getPlayers().forEach((player, playerId) => {
        if (playerId != peerId) {
            session.sendReliableMessage(outMessage, playerId);
        }
    });
    activePlayers--;
}

// Handle a message to the server
function onMessage(gameMessage) {
    switch (gameMessage.opCode) {
        case OP_CODE_CUSTOM_OP1: {
            // do operation 1 with gameMessage.payload for example sendToGroup
            const outMessage = session.newTextGameMessage(OP_CODE_CUSTOM_OP1_REPLY,
session.getServerId(), gameMessage.payload);
            session.sendGroupMessage(outMessage, RED_TEAM_GROUP);
            break;
        }
    }
}

// Return true if the send should be allowed
function onSendToPlayer(gameMessage) {
    // This example rejects any payloads containing "Reject"
    return (!gameMessage.getPayloadAsText().includes("Reject"));
}

// Return true if the send to group should be allowed
// Use gameMessage.getPayloadAsText() to get the message contents
function onSendToGroup(gameMessage) {
    return true;
}

// Return true if the player is allowed to join the group
function onPlayerJoinGroup(groupId, peerId) {
    return true;
}

// Return true if the player is allowed to leave the group
function onPlayerLeaveGroup(groupId, peerId) {
    return true;
}

// A simple tick loop example

```

```
// Checks to see if a minimum amount of time has passed before seeing if the game has
ended
async function tickLoop() {
  const elapsedTime = getTimeInS() - startTime;
  logger.info("Tick... " + elapsedTime + " activePlayers: " + activePlayers);

  // In Tick loop - see if all players have left early after a minimum period of time
  has passed
  // Call processEnding() to terminate the process and quit
  if ( (activePlayers == 0) && (elapsedTime > minimumElapsedTime)) {
    logger.info("All players disconnected. Ending game");
    const outcome = await session.processEnding();
    logger.info("Completed process ending with: " + outcome);
    process.exit(0);
  }
  else {
    setTimeout(tickLoop, tickTime);
  }
}

// Calculates the current time in seconds
function getTimeInS() {
  return Math.round(new Date().getTime()/1000);
}

exports.ssExports = {
  configuration: configuration,
  init: init,
  onProcessStarted: onProcessStarted,
  onMessage: onMessage,
  onPlayerConnect: onPlayerConnect,
  onPlayerAccepted: onPlayerAccepted,
  onPlayerDisconnect: onPlayerDisconnect,
  onSendToPlayer: onSendToPlayer,
  onSendToGroup: onSendToGroup,
  onPlayerJoinGroup: onPlayerJoinGroup,
  onPlayerLeaveGroup: onPlayerLeaveGroup,
  onStartGameSession: onStartGameSession,
  onProcessTerminate: onProcessTerminate,
  onHealthCheck: onHealthCheck
};
```

Déploiement d'un logiciel de serveur de jeu pour l'Amazon GameLift Servershébergement

Déployez votre logiciel de serveur de jeu multijoueur en l'installant sur vos ressources d'hébergement, en lançant les processus du serveur de jeu et en les préparant à héberger des jeux pour les joueurs. Les étapes à suivre pour préparer le logiciel du serveur de jeu en vue du déploiement dépendent du type de Amazon GameLift Servers solution que vous utilisez. Dans tous les scénarios, le logiciel du serveur de jeu déployé interagit avec le Amazon GameLift Servers service pour gérer le placement des sessions de jeu et communiquer les informations de connexion à un client de jeu.

Le logiciel de votre serveur de jeu doit d'abord être intégréAmazon GameLift Servers, comme décrit dans[Préparation de jeux pour Amazon GameLift Servers](#).

Les rubriques de cette section fournissent des conseils sur la façon de préparer votre logiciel pour le déploiement dans les scénarios suivants.

- Logiciel de serveur de jeu personnalisé
 - Pour l' EC2 hébergement Amazon GameLift Servers géré
 - Pour l'hébergement de conteneurs Amazon GameLift Servers gérés
 - Hébergement pour Amazon GameLift Servers n'importe où
- Amazon GameLift ServersRealtimescript personnalisé pour l' EC2 hébergement Amazon GameLift Servers géré

Rubriques

- [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#)
- [Créez une image de conteneur pour Amazon GameLift Servers](#)
- [Déployez un script pour Amazon GameLift ServersRealtime](#)

Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement

Après avoir intégré votre serveur de jeu à Amazon GameLift Servers (voir[Préparation de jeux pour Amazon GameLift Servers](#)), installez le logiciel du serveur de jeu sur vos ressources informatiques

pour l'hébergement. Ce processus varie en fonction du type de Amazon GameLift Servers l'hébergement que vous utilisez.

Déploiement pour un hébergement géré

Si vous utilisez Amazon GameLift Servers EC2 hébergement géré, vous devez emballer le logiciel de votre serveur de jeu et le télécharger sur Amazon GameLift Servers. Lorsque vous créez une flotte gérée, Amazon GameLift Servers le déploie automatiquement sur chaque instance de flotte.

Les rubriques de cette section décrivent comment emballer vos fichiers de build pour le téléchargement, créer un script d'installation de build facultatif, puis télécharger les fichiers à l'aide du [AWS Command Line Interface \(AWS CLI\)](#) ou du AWS SDK.

Déploiement pour un hébergement en tout lieu

Si vous utilisez Amazon GameLift Servers Anywhere Fleets Pour un hébergement autogéré, il est de votre responsabilité d'installer le logiciel du serveur de jeu sur chaque ordinateur d'une flotte et de le maintenir à jour.

Lorsqu'un processus de serveur de jeu intégré commence à s'exécuter, il s'initialise automatiquement et établit une communication avec Amazon GameLift Servers service. Le processus du serveur démarre les sessions de jeu à partir des instructions fournies par Amazon GameLift Servers et rend compte de l'activité au service.

Rubriques

- [Conditionner les fichiers de version de jeu](#)
- [Ajouter un script d'installation de build](#)
- [Créez un Amazon GameLift Servers créer une ressource pour l'hébergement géré](#)
- [Mettre à jour une version de serveur de jeu pour Amazon GameLift Servers hébergement géré](#)

Conditionner les fichiers de version de jeu

Avant de télécharger votre serveur de jeu configuré sur Amazon GameLift Servers, emballez les fichiers de compilation du jeu dans un répertoire de compilation. Ce processus est obligatoire lors de l'hébergement avec une flotte EC2 gérée, et constitue une bonne pratique lors de l'hébergement avec une flotte Anywhere. Le répertoire de compilation doit inclure tous les composants nécessaires au fonctionnement de vos serveurs de jeu et à l'hébergement de sessions de jeu. Cela peut inclure les éléments suivants :

- Fichiers binaires du serveur de jeu : fichiers binaires nécessaires au fonctionnement du serveur de jeu. Une version peut inclure des fichiers binaires pour plusieurs serveurs de jeu conçus pour fonctionner sur la même plateforme. Pour obtenir la liste des plateformes prises en charge, consultez [Obtenez des outils Amazon GameLift Servers de développement](#).
- Dépendances : tous les fichiers dépendants dont les exécutables de votre serveur de jeu ont besoin pour fonctionner. Exemples : actifs, fichiers de configuration et bibliothèques dépendantes.

Note

Pour les builds de jeux créés avec le SDK du serveur pour Amazon GameLift Servers pour le C++ (y compris ceux créés avec le plugin Unreal), incluez la DLL OpenSSL pour la même version d'OpenSSL que celle avec laquelle vous avez créé le SDK du serveur. Consultez le fichier README du SDK du serveur pour plus de détails.

- Script d'installation (facultatif) : fichier de script permettant de gérer les tâches relatives à l'installation du build de votre jeu Amazon GameLift Servers serveurs d'hébergement. Placez ce fichier à la racine du répertoire de construction. Amazon GameLift Servers exécute le script d'installation dans le cadre de la création de la flotte.

Vous pouvez configurer n'importe quelle application de votre build, y compris votre script d'installation, pour accéder à vos ressources en toute sécurité sur d'autres AWS services. Pour plus d'informations sur les méthodes à suivre, consultez [Communiquez avec les autres AWS ressources de vos flottes](#).

Après avoir empaqueté vos fichiers de compilation, assurez-vous que votre serveur de jeu peut fonctionner sur une nouvelle installation de votre système d'exploitation cible afin de vérifier que toutes les dépendances requises sont incluses et que votre script d'installation est correct.

Ajouter un script d'installation de build

Créez un script d'installation pour le système d'exploitation (OS) de votre build de jeu :

- Windows : créez un fichier batch nommé `install.bat`.
- Linux : créez un fichier de script shell nommé `install.sh`.

Lors de la création d'un script d'installation, gardez à l'esprit les informations suivantes :

- Le script ne peut accepter aucune entrée utilisateur.

- Amazon GameLift Servers installe le build et recrée les répertoires de fichiers de votre package de build sur un serveur d'hébergement aux emplacements suivants :
 - Flottes Windows : C:\game
 - Flottes Linux : /local/game
- Pendant le processus d'installation des flottes Linux, l'utilisateur run-as dispose d'un accès limité à la structure des fichiers d'instance. Cet utilisateur dispose de tous les droits sur le répertoire dans lequel vos fichiers de compilation sont installés. Si votre script d'installation exécute des actions qui nécessitent des autorisations d'administrateur, spécifiez l'accès administrateur à l'aide de `sudo`. L'utilisateur run-as pour les flottes Windows dispose d'autorisations d'administrateur par défaut. Les échecs d'autorisation liés au script d'installation génèrent un message d'événement indiquant un problème avec le script.
- Sur Linux, Amazon GameLift Servers prend en charge les langages d'interprétation shell courants tels que bash. Ajoutez un package (par exemple, `#!/bin/bash`) en haut de votre script d'installation. Pour vérifier la prise en charge de vos commandes shell préférées, accédez à distance à une instance Linux active et ouvrez une invite du shell. Pour de plus amples informations, veuillez consulter [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).
- Le script d'installation ne peut pas s'appuyer sur une connexion d'appairage VPC. Une connexion d'appairage VPC n'est disponible qu'après Amazon GameLift Servers installe le build sur les instances de flotte.

Exemple Fichier bash d'installation de Windows

Ce `install.bat` fichier d'exemple installe les composants d'exécution Visual C++ requis pour le serveur de jeu et écrit les résultats dans un fichier journal. Le script inclut le fichier de composant dans le package de construction à la racine.

```
vcxredist_x64.exe /install /quiet /norestart /log c:\game\vcxredist_2013_x64.log
```

Exemple Script shell d'installation de Linux

Ce `install.sh` fichier d'exemple utilise bash dans le script d'installation et écrit les résultats dans un fichier journal.

```
#!/bin/bash
echo 'Hello World' > install.log
```

Ce `install.sh` fichier d'exemple montre comment vous pouvez utiliser l' CloudWatch agent Amazon pour collecter des métriques personnalisées et au niveau du système, et gérer la rotation des journaux. Parce que Amazon GameLift Servers s'exécute dans un VPC de service, vous devez accorder Amazon GameLift Servers autorisations pour assumer un rôle AWS Identity and Access Management (IAM) en votre nom. Pour autoriser Amazon GameLift Servers pour assumer un rôle, créez un rôle qui inclut la politique AWS CloudWatchAgentAdminPolicy gérée et utilisez ce rôle lorsque vous créez une flotte.

```
sudo yum install -y amazon-cloudwatch-agent
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
sudo yum install -y collectd
cat <<'EOF' > /tmp/config.json
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "root",
        "credentials": {
            "role_arn": "arn:aws:iam::account#:role/rolename"
        }
    },
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/tmp/log",
                        "log_group_name": "gllog",
                        "log_stream_name": "{instance_id}"
                    }
                ]
            }
        }
    },
    "metrics": {
        "namespace": "GL_Metric",
        "append_dimensions": {
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
```

```
// Configure metrics you want to collect.
// For more information, see Manually create or edit the CloudWatch agent configuration file.
}
}
}
EOF
sudo mv /tmp/config.json /opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
sudo systemctl enable amazon-cloudwatch-agent.service
```

Créez un Amazon GameLift Servers créer une ressource pour l'hébergement géré

Lorsque vous créez une build et téléchargez vos fichiers, vous disposez de quelques options :

- [Création d'un build à partir d'un répertoire de fichiers](#). Il s'agit de l'option la plus simple et la plus couramment utilisée.
- [Créez un build avec des fichiers dans Amazon Simple Storage Service \(Amazon S3\)](#). Avec cette option, vous pouvez gérer vos versions de build dans Amazon S3.

Avec les deux méthodes, Amazon GameLift Servers crée une nouvelle ressource de build avec un ID de build unique et d'autres métadonnées. La construction démarre avec le statut Initialisé. Après Amazon GameLift Servers acquiert les fichiers du serveur de jeu, le build passe à l'état Prêt.

Lorsque le build est prêt, vous pouvez le déployer sur un nouveau Amazon GameLift Servers flotte. Pour plus d'informations, voir [Création d'une EC2 flotte Amazon GameLift Servers gérée](#). Quand Amazon GameLift Servers configure le nouveau parc, télécharge les fichiers de build sur chaque instance de flotte et installe les fichiers de build.

Création d'un build à partir d'un répertoire de fichiers

Pour créer une version de jeu stockée dans n'importe quel emplacement, y compris dans un répertoire local, utilisez la [upload-build](#) AWS CLI commande. Cette commande crée un nouvel enregistrement de build dans Amazon GameLift Servers et télécharge les fichiers à partir d'un emplacement que vous spécifiez.

Envoyez une demande de chargement. Dans une fenêtre de ligne de commande, entrez la upload-build commande et les paramètres suivants.


```
aws gamelift upload-build \  
  --name user-defined name of build \  
  --operating-system supported OS \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --build-root build path \  
  --build-version user-defined build number \  
  --region region name
```

- **operating-system**— L'environnement d'exécution du build du serveur de jeu. Vous devez spécifier une valeur de système d'exploitation. Vous ne pourrez pas le mettre à jour ultérieurement.
- **server-sdk-version**— La version du Amazon GameLift Servers SDK du serveur auquel votre serveur de jeu est intégré. Si vous ne fournissez aucune valeur, Amazon GameLift Servers utilise la valeur par défaut 4.0.2. Si vous spécifiez une version du SDK de serveur incorrecte, la compilation du serveur de jeu risque d'échouer lors de `InitSdk` l'appel pour établir une connexion au Amazon GameLift Servers service.
- **build-root**— Le chemin du répertoire de vos fichiers de compilation.
- **name**— Nom descriptif de la nouvelle version.
- **build-version**— Les détails de version des fichiers de compilation.
- **region**— La AWS région dans laquelle vous souhaitez créer votre build. Créez le build dans la région où vous prévoyez de déployer des flottes. Si vous déployez votre jeu dans plusieurs régions, créez un build dans chaque région.

Note

Consultez votre région par défaut actuelle à l'aide du [aws configure get region](#). Pour modifier votre région par défaut, utilisez la [aws configure set region **region name**](#) commande.

Exemples

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-east-1
```

```
--region us-west-2
```

```
aws gamelift upload-build \  
  --operating-system WINDOWS_2016 \  
  --server-sdk-version "5.0.0" \  
  --build-root "C:\mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

En réponse à votre demande de téléchargement, Amazon GameLift Servers indique la progression du téléchargement. En cas de téléchargement réussi, Amazon GameLift Servers renvoie le nouvel ID d'enregistrement de construction. Le temps de chargement dépend de la taille de vos fichiers de jeu et de la vitesse de connexion.

Création d'un build avec des fichiers dans Amazon S3

Vous pouvez stocker vos fichiers de compilation dans Amazon S3 et les télécharger sur Amazon GameLift Servers à partir de là. Lorsque vous créez votre build, vous spécifiez l'emplacement du compartiment S3, et Amazon GameLift Servers récupère les fichiers de compilation directement depuis Amazon S3.

Pour créer une ressource de construction

1. Stockez vos fichiers de compilation dans Amazon S3. Créez un fichier .zip contenant les fichiers de construction empaquetés et téléchargez-le dans un compartiment S3 de votre Compte AWS. Prenez note de l'étiquette du compartiment et du nom du fichier, vous en aurez besoin lors de la création d'un Amazon GameLift Servers construire.
2. Donnez Amazon GameLift Servers accès à vos fichiers de compilation. Créez un rôle IAM en suivant les instructions de [Accédez à un fichier de compilation de jeu dans Amazon S3](#). Après avoir créé le rôle, prenez note du nom de ressource Amazon (ARN) du nouveau rôle. Vous en aurez besoin lors de la création d'un build.
3. Créez un build. Utilisez la commande Amazon GameLift Servers console ou AWS CLI pour créer un nouvel enregistrement de build. Vous devez avoir l'`PassRole` autorisation, comme décrit dans [Exemples d'autorisations IAM pour Amazon GameLift Servers](#).

Console

1. Dans [Amazon GameLift Servers console](#), dans le volet de navigation, choisissez Hosting, Builds.
2. Sur la page Builds, choisissez Create build.
3. Sur la page Créer une version, sous Paramètres de génération, procédez comme suit :
 - a. Dans Nom, entrez le nom du script.
 - b. Pour Version, entrez une version. Comme vous pouvez mettre à jour le contenu d'une version, les données de version peuvent vous aider à suivre les mises à jour.
 - c. Dans Système d'exploitation (OS), choisissez le système d'exploitation de la version de votre serveur de jeu. Vous ne pourrez pas mettre à jour cette valeur ultérieurement.
 - d. Pour le build du serveur de jeu, entrez l'URI S3 de l'objet de build que vous avez chargé sur Amazon S3, puis choisissez la version de l'objet. Si vous ne vous souvenez pas de l'URI et de la version de l'objet Amazon S3, choisissez Browse S3 et recherchez l'objet de construction.
 - e. Pour le rôle IAM, choisissez le rôle que vous avez créé qui donne Amazon GameLift Servers accès à votre compartiment S3 et à votre objet de construction.
4. (Facultatif) Sous Balises, ajoutez des balises au build en saisissant des paires clé et valeur.
5. Sélectionnez Create (Créer).

Amazon GameLift Servers attribue un identifiant à la nouvelle version et télécharge le fichier .zip désigné. Vous pouvez consulter le nouveau build, y compris son statut, sur la page Builds.

AWS CLI

Pour définir le nouveau build et télécharger les fichiers de build de votre serveur, utilisez la [create-build](#) commande.

1. Ouvrez une fenêtre de ligne de commande et passez à un répertoire dans lequel vous pouvez utiliser le AWS CLI.
2. Entrez la create-build commande suivante :

```
aws gamelift create-build \  
  --name user-defined name of build \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --operating-system supported OS \  
  --s3-uri S3 URI of build object \  
  --s3-version S3 version of build object \  
  --iam-role ARN of IAM role \  
  --tags key-value pairs
```

```
--build-version user-defined build number \  
--storage-location "Bucket"=S3 bucket label,"Key"=Build .zip file  
name,"RoleArn"=Access role ARN} \  
--region region name
```

- **name**— Nom descriptif de la nouvelle version.
- **server-sdk-version**— La version du SDK du serveur pour Amazon GameLift Servers vous aviez l'habitude d'intégrer votre serveur de jeu à Amazon GameLift Servers. Si vous ne fournissez aucune valeur, Amazon GameLift Servers utilise la valeur par défaut 4.0.2.
- **operating-system**— L'environnement d'exécution du build du serveur de jeu. Vous devez spécifier une valeur de système d'exploitation. Vous ne pourrez pas le mettre à jour ultérieurement.
- **build-version**— Les détails de version des fichiers de compilation. Ces informations peuvent être utiles car chaque nouvelle version de votre serveur de jeu nécessite une nouvelle ressource de compilation.
- **storage-location**
 - **Bucket**— Le nom du compartiment S3 qui contient votre build. Par exemple, « my_build_files ».
 - **Key**— Le nom du fichier .zip qui contient vos fichiers de compilation. Par exemple, « my_game_build_7.0.1, 7.0.2 ».
 - **RoleARN**— L'ARN attribué au rôle IAM que vous avez créé. Par exemple, « arn:aws:iam : :111122223333:role/ ». GameLiftAccess Pour un exemple de politique, consultez [Accédez à un fichier de compilation de jeu dans Amazon S3](#).
- **region**— Créez le build dans la AWS région où vous prévoyez de déployer des flottes. Si vous déployez votre jeu dans plusieurs régions, créez un build dans chaque région.

Note

Nous vous recommandons de vérifier votre région par défaut actuelle à l'aide de la [configure get](#) commande . Pour modifier votre région par défaut, utilisez la [configure set](#) commande.

Exemple

```
aws gamelift create-build \  

```

```
--operating-system WINDOWS_2016 \  
--storage-location  
"Bucket"="my_game_build_files", "Key"="mygame_build_101.zip", "RoleArn"="arn:aws:iam::111  
gamelift" \  
--name "My Game Nightly Build" \  
--build-version "build 101" \  
--region us-west-2
```

3. Pour afficher la nouvelle version, utilisez la [describe-build](#) commande.

Mettre à jour une version de serveur de jeu pour Amazon GameLift Servers hébergement géré

Lorsque vous déployez votre serveur de jeu compilé pour Amazon GameLift Servers EC2 hébergement géré, vous téléchargez le logiciel de votre serveur de jeu et créez un Amazon GameLift Servers créer une ressource. Une fois que vous avez créé un Amazon GameLift Servers build, vous pouvez mettre à jour les métadonnées du build, mais vous ne pouvez pas mettre à jour les fichiers de build eux-mêmes. Pour déployer les mises à jour sur votre serveur de jeu, téléchargez les fichiers mis à jour et créez un nouveau Amazon GameLift Servers construire à l'aide de la AWS CLI commande [upload-build](#) de commande. Vous pouvez également utiliser la [create-build](#) commande pour télécharger une nouvelle version à partir d'un compartiment Amazon S3 que vous contrôlez. Déployez ensuite la nouvelle version en créant une nouvelle flotte pour celle-ci.

Vous pouvez mettre à jour les métadonnées d'une version, y compris le nom et la description. Pour ces tâches, utilisez Amazon GameLift Servers console ou [update-build](#) AWS CLI commande.

Automatisez les mises à jour de vos builds de

Suivez ces conseils pour automatiser et rationaliser le processus de mise à jour des versions des serveurs de jeu pour Amazon GameLift Servers flottes gérées :

- Utilisez les files d'attente des sessions de jeu et échangez des flottes selon vos besoins. Lorsque vous envoyez des demandes de session de jeu à Amazon GameLift Servers, spécifiez une file d'attente de session de jeu au lieu d'une flotte spécifique. Avec les files d'attente, vous pouvez ajouter des flottes avec une nouvelle version et supprimer les anciennes flottes si nécessaire. Pour de plus amples informations, veuillez consulter [Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente](#).
- Utilisez des alias pour transférer les joueurs vers une nouvelle version du jeu. Lorsque vous envoyez des demandes de session de jeu à Amazon GameLift Servers, spécifiez un alias de flotte

au lieu d'un ID de flotte. Pour de plus amples informations, veuillez consulter [Créez un Amazon GameLift Servers alias](#).

- Configuré pour le développement itératif. Pendant le développement du jeu, explorez les options permettant de configurer un environnement de test hébergé permettant un développement itératif rapide. Consultez [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#).

Testez ces ressources du [Amazon GameLift Servers Boîte à outils](#) sur Github :

Outil de mise à jour rapide des versions (pour le développement uniquement)

Cet outil vous permet de modifier les versions de serveurs de jeu déjà déployées sur les ordinateurs d'une EC2 flotte gérée, ce qui vous permet de gagner du temps lors des itérations de développement rapides. L'outil dispose de plusieurs options ; vous pouvez remplacer la version complète du jeu ou modifier des fichiers spécifiques, et vous pouvez gérer le redémarrage des processus du serveur de jeu après les mises à jour. Vous pouvez également l'utiliser pour mettre à jour tous les calculs d'un parc ou pour cibler des calculs individuels.

Visitez le Amazon GameLift Servers Repo Toolkit sur Github pour obtenir l'[outil de mise à jour rapide](#) des builds sur Github et en savoir plus sur son utilisation.

Exemple de script de déploiement en production

Ce script montre comment automatiser le processus de mise à jour des versions de serveurs de jeu déployées sur des EC2 flottes gérées en production. Pour utiliser ce script, votre Amazon GameLift Servers la solution d'hébergement doit utiliser des alias pour abstraire la flotte IDs. L'exemple de script gère les étapes de base suivantes : télécharger une version mise à jour, créer une nouvelle version et la déployer dans une nouvelle flotte, rediriger le trafic des joueurs d'une flotte existante vers la nouvelle flotte et supprimer l'ancienne flotte. Personnalisez l'exemple de script pour répondre à vos exigences de déploiement spécifiques.

Visitez le Amazon GameLift Servers Repo Toolkit sur Github pour obtenir l'[exemple de script de déploiement en production](#) dans Github et en savoir plus sur son utilisation.

Créez une image de conteneur pour Amazon GameLift Servers

Cette rubrique explique comment créer une image de conteneur à l'aide du logiciel de votre serveur de jeu Amazon GameLift Servers. Une image de conteneur de serveur de jeu inclut l'exécutable du serveur de jeu et toutes les dépendances dont il a besoin pour fonctionner. Les images de

conteneurs de serveurs de jeux sont utilisées avec une solution d'hébergement de conteneurs Amazon GameLift Servers gérés. Pour plus de détails sur la création de la solution complète, voir :

- [Feuille de route de développement pour l'hébergement avec des conteneurs Amazon GameLift Servers gérés](#)
- [Comment fonctionnent les conteneurs dans Amazon GameLift Servers](#)

Effectuez les tâches suivantes pour préparer l'image de conteneur de votre serveur de jeu à être déployée sur une flotte de Amazon GameLift Servers conteneurs. Avant de commencer ces tâches, terminez d'intégrer le code de votre serveur de jeu au SDK Amazon GameLift Servers du serveur.

Rubriques

- [Création d'une image de conteneur de serveur de jeu](#)
- [Transférer une image de conteneur vers Amazon ECR](#)

Création d'une image de conteneur de serveur de jeu

Suivez ces instructions lorsque vous travaillez sur une plate-forme Linux ou que vous utilisez le sous-système Windows pour Linux (WSL) avec Docker installé.

Pour créer une image de conteneur de serveur de jeu

1. Préparez un répertoire de travail avec le logiciel de votre serveur de jeu. Sur une machine locale, créez un répertoire de travail pour organiser les fichiers du conteneur de votre serveur de jeu. Votre image de conteneur utilise cette structure de fichier lors du déploiement du conteneur sur Amazon GameLift Servers des ressources à des fins d'hébergement. Par exemple :

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .  
.  
./glc  
./glc/gamebuild
```

Note

Si vous essayez cette fonctionnalité et que vous n'avez pas encore de version de serveur de jeu fonctionnelle, essayez notre exemple de serveur de jeu [SimpleServer](#), disponible sur GitHub.

2. Créez un nouveau Dockerfile à l'aide du modèle fourni.
3. Suivez les instructions du modèle Dockerfile pour le mettre à jour pour votre propre usage.
 - Mettez à jour l'image de base selon vos besoins.
 - Définissez des variables d'environnement pour la construction de votre serveur de jeu.
4. Créez l'image du conteneur. Exécutez le `docker build` en spécifiant le nom de votre propre dépôt. Par exemple :

```
[~/work/glc]$ docker build -t <local repository name>:<optional tag> .
```

Vous pouvez afficher vos référentiels et vos images IDs à l'aide de la `docker images` commande, comme illustré dans cet exemple :

Modèle Dockerfile pour une image de conteneur de serveur de jeu

Ce modèle contient les instructions minimales dont un conteneur de serveur de jeu a besoin pour être utilisé dans une Amazon GameLift Servers flotte. Modifiez le contenu selon les besoins de votre serveur de jeu.

```
# Base image
# -----
# Add the base image that you want to use,
# Make sure to use an image with the same architecture as the
# Instance type you are planning to use on your fleets.
FROM public.ecr.aws/amazonlinux/amazonlinux
#
# Game build directory
# -----
# Add your gamebuild directory to the env variable below.
# The game build provided here needs to be integrated with server sdk for Amazon
GameLift Servers.
ENV GAME_BUILD_DIRECTORY="<ADD_GAME_BUILD_DIRECTORY>" \
#
# Game executable and launch parameters
# -----
# Add the relative path to your executable in the 'GAME_EXECUTABLE' env variable
below.
# The game build provided over here needs to be integrated with server sdk for Amazon
GameLift Servers.
```



```

# This template assumes that the executable path is relative to the game build
directory.
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
# Add 'HOME_DIR' to identify where the game executable and logs exist.
GAME_EXECUTABLE="<ADD NAME OF EXECUTABLE WITHIN THE GAME DIRECTORY>" \
LAUNCH_PARAMS=<ADD LAUNCH PARAMETERS> \
HOME_DIR="/local/game" \

# Install dependencies as necessary
RUN yum install -y shadow-utils

RUN mkdir -p $HOME_DIR
COPY ./$GAME_BUILD_DIRECTORY/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR

# Set up for 'gamelift' user
RUN useradd -m gamelift && \
    echo "gamelift ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
    chown -R gamelift:gamelift $HOME_DIR

# Add permissions to game build
RUN chmod +x ./$GAME_EXECUTABLE

USER gamelift

# Check directory before starting the container
RUN ls -lhrt .

# Check path before starting the container
RUN echo $PATH

# Start the game build
ENTRYPOINT ["/bin/sh", "-c", "./$GAME_EXECUTABLE", "$LAUNCH_PARAMS"]

```

Transférer une image de conteneur vers Amazon ECR

Après avoir créé une image de conteneur pour le déploiement Amazon GameLift Servers, stockez-la dans un référentiel public ou privé sur Amazon ECR. Une valeur d'URI est attribuée à ce

référentiel, qui Amazon GameLift Servers permet de prendre un instantané de l'image en vue de son déploiement sur un parc de conteneurs.

 Note

Si vous ne possédez pas encore de référentiel privé Amazon ECR, [créez-en un](#).

Pour transférer l'image de votre conteneur vers Amazon ECR

1. Obtenez vos informations d'identification Amazon ECR. Avant de transférer une image de conteneur vers Amazon ECR, vous devez d'abord acquérir vos AWS informations d'identification sous forme temporaire et les fournir à Docker. Docker a besoin de ces informations d'identification pour se connecter.

```
[~/work/glc]$ aws ecr get-login-password --region us-west-2 | docker login --  
username AWS --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com  
WARNING! Your password will be stored unencrypted in  
/home/user-name/.docker/config.json.  
Configure a credential helper to remove this warning.  
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded
```

2. Copiez l'URI du [référentiel privé Amazon ECR](#) que vous souhaitez utiliser.
3. Appliquez une balise Amazon ECR à l'image de votre conteneur.

Exemple

```
[~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository  
URI>:<optional tag>
```

4. Transférez l'image de votre conteneur vers Amazon ECR

Exemple

```
[~/work/glc]$ docker image push <Amazon ECR private repository URI>
```

Déployez un script pour Amazon GameLift ServersRealtime

Lorsque vous êtes prêt à effectuer le déploiement Amazon GameLift Servers Realtime pour votre jeu, téléchargez les fichiers de script Realtime du serveur terminés sur Amazon GameLift Servers. Pour ce faire, créez une ressource de Amazon GameLift Servers script et spécifiez l'emplacement de vos fichiers de script. Vous pouvez également mettre à jour les fichiers de script du serveur déjà déployés en téléchargeant de nouveaux fichiers pour une ressource de script existante.

Lorsque vous créez une nouvelle ressource de script, Amazon GameLift Servers attribuez-lui un identifiant de script unique (par exemple, `script-1111aaaa-22bb-33cc-44dd-5555eeee66ff`) et téléchargez une copie des fichiers de script. Le temps de téléchargement dépend de la taille de vos fichiers de script et de la vitesse de votre connexion.

Après avoir créé la ressource de script, Amazon GameLift Servers déploie le script avec une nouvelle Amazon GameLift Servers Realtime flotte. Amazon GameLift Servers installe votre script de serveur sur chaque instance du parc, en y plaçant les fichiers de script. `/local/game`

Pour résoudre les problèmes d'activation du parc liés au script du serveur, voir [Déboguer les incidents de flotte Amazon GameLift Servers](#).

Fichiers de script du package

Le script de votre serveur peut inclure un ou plusieurs fichiers combinés dans un seul fichier .zip pour le téléchargement. Le fichier .zip doit contenir tous les fichiers dont votre script a besoin pour s'exécuter.

Vous pouvez stocker vos fichiers de script compressés dans un répertoire de fichiers local ou dans un bucket Amazon Simple Storage Service (Amazon S3).

Télécharger des fichiers de script depuis un répertoire local

Si vos fichiers de script sont stockés localement, vous pouvez les télécharger à Amazon GameLift Servers partir de là. Pour créer la ressource de script, utilisez la Amazon GameLift Servers console ou le [AWS Command Line Interface \(AWS CLI\)](#).

GameLift Serveurs Amazon console

Pour créer une ressource de script

1. Ouvrez la [Amazon GameLift Servers console](#).

2. Dans le volet de navigation, choisissez Hosting, Scripts.
3. Sur la page Scripts, choisissez Créer un script.
4. Sur la page Créer un script, sous Paramètres du script, procédez comme suit :
 - a. Dans Nom, entrez le nom du script.
 - b. (Facultatif) Dans Version, entrez les informations de version. Comme vous pouvez mettre à jour le contenu d'un script, les données de version peuvent être utiles pour suivre les mises à jour.
 - c. Pour Source du script, choisissez Charger un fichier .zip.
 - d. Pour les fichiers de script, choisissez Choisir un fichier, recherchez le fichier .zip contenant votre script, puis choisissez ce fichier.
5. (Facultatif) Sous Balises, ajoutez des balises au script en saisissant des paires clé et valeur.
6. Choisissez Créer.

Amazon GameLift Servers attribue un identifiant au nouveau script et télécharge le fichier .zip désigné. Vous pouvez consulter le nouveau script, y compris son statut, sur la page Scripts.

AWS CLI

Utilisez la commande [create-script](#) AWS CLI pour définir le nouveau script et charger vos fichiers de script de serveur.

Pour créer une ressource de script

1. Placez le fichier .zip dans un répertoire où vous pouvez utiliser le AWS CLI.
2. Ouvrez une fenêtre de ligne de commande et passez au répertoire dans lequel vous avez placé le fichier .zip.
3. Entrez la create-script commande et les paramètres suivants. Pour le --zip-file paramètre, veillez à ajouter la chaîne fileb:// au nom du fichier .zip. Il identifie le fichier comme étant binaire afin de Amazon GameLift Servers traiter le contenu compressé.

```
aws gamelift create-script \  
  --name user-defined name of script \  
  --script-version user-defined version info \  
  --zip-file fileb://name of zip file \  
  --region region name
```

Exemple

```
aws gamelift create-script \  
  --name "My_Realtime_Server_Script_1" \  
  --script-version "1.0.0" \  
  --zip-file fileb://myrealtime_script_1.0.0.zip \  
  --region us-west-2
```

En réponse à votre demande, Amazon GameLift Servers renvoie le nouvel objet de script.

4. Pour voir le nouveau script, appelez [describe-script](#).

Charger des fichiers de script depuis Amazon S3

Vous pouvez stocker vos fichiers de script dans un compartiment Amazon S3 et les Amazon GameLift Servers y charger. Lorsque vous créez votre script, vous spécifiez l'emplacement du compartiment S3 et vous Amazon GameLift Servers récupérez vos fichiers de script sur Amazon S3.

Pour créer une ressource de script

1. Stockez vos fichiers de script dans un compartiment S3. Créez un fichier .zip contenant les fichiers de script de votre serveur et chargez-le dans un compartiment S3 Compte AWS que vous contrôlez. Prenez note de l'URI de l'objet : vous en avez besoin lors de la création d'un Amazon GameLift Servers script.

Note

Amazon GameLift Servers ne prend pas en charge le téléchargement depuis des compartiments S3 dont les noms contiennent un point (.).

2. Accordez à Amazon GameLift Servers l'accès à vos fichiers de script. Pour créer un rôle AWS Identity and Access Management (IAM) permettant d'accéder Amazon GameLift Servers au compartiment S3 contenant votre script de serveur, suivez les instructions figurant dans [Configurer un rôle de service IAM pour Amazon GameLift Servers](#). Après avoir créé le nouveau rôle, prenez note de son nom, dont vous aurez besoin pour créer un script.
3. Créez un script. Utilisez la Amazon GameLift Servers console ou le AWS CLI pour créer un nouvel enregistrement de script. Pour effectuer cette demande, vous devez disposer de

l'`PassRole` autorisation IAM, comme décrit dans [Exemples d'autorisations IAM pour Amazon GameLift Servers](#).

GameLift Serveurs Amazon console

1. Dans le volet de navigation de la [Amazon GameLift Servers console](#), choisissez Hosting, Scripts.
2. Sur la page Scripts, choisissez Créer un script.
3. Sur la page Créer un script, sous Paramètres du script, procédez comme suit :
 - a. Dans Nom, entrez le nom du script.
 - b. (Facultatif) Dans Version, entrez les informations de version. Comme vous pouvez mettre à jour le contenu d'un script, les données de version peuvent être utiles pour suivre les mises à jour.
 - c. Pour Source du script, choisissez l'URI Amazon S3.
 - d. Entrez l'URI S3 de l'objet de script que vous avez chargé sur Amazon S3, puis choisissez la version de l'objet. Si vous ne vous souvenez pas de l'URI et de la version de l'objet Amazon S3, choisissez Browse S3, puis recherchez l'objet de script.
4. (Facultatif) Sous Balises, ajoutez des balises au script en saisissant des paires clé et valeur.
5. Choisissez Créer.

Amazon GameLift Servers attribue un identifiant au nouveau script et télécharge le fichier .zip désigné. Vous pouvez consulter le nouveau script, y compris son statut, sur la page Scripts.

AWS CLI

Utilisez la commande [create-script](#) AWS CLI pour définir le nouveau script et charger vos fichiers de script de serveur.

1. Ouvrez une fenêtre de ligne de commande et passez à un répertoire dans lequel vous pouvez utiliser le AWS CLI.
2. Entrez la create-script commande et les paramètres suivants. Le `--storage-location` paramètre indique l'emplacement du compartiment Amazon S3 de vos fichiers de script.

```
aws gamelift create-script \  
  --name [user-defined name of script] \  
  --storage-location [storage location] \  
  --script-file [script file]
```

```
--script-version [user-defined version info] \  
--storage-location "Bucket"=S3 bucket name,"Key"=name of zip file in S3 bucket,"RoleArn"=Access role ARN \  
--region region name
```

Exemple

```
aws gamelift create-script \  
  --name "My_Realtime_Server_Script_1" \  
  --script-version "1.0.0" \  
  --storage-location "Bucket"="gamelift-script","Key"="myrealtime_script_1.0.0.zip","RoleArn"="arn:aws:iam::123456789012:role/S3Access" \  
  --region us-west-2
```

En réponse à votre demande, Amazon GameLift Servers renvoie le nouvel objet de script.

3. Pour voir le nouveau script, appelez [describe-script](#).

Mettre à jour les fichiers de Amazon GameLift ServersRealtime script

Vous pouvez mettre à jour les métadonnées d'une ressource de script à l'aide de la Amazon GameLift Servers console ou de la [update-script](#) AWS CLI commande.

Vous pouvez également mettre à jour le contenu d'une ressource de script. Amazon GameLift Servers déploie le contenu du script sur toutes les instances de flotte qui utilisent la ressource de script mise à jour. Lorsque le script mis à jour est déployé, les instances l'utilisent lors du démarrage de nouvelles sessions de jeu. Les sessions de jeu déjà en cours au moment de la mise à jour n'utilisent pas le script mis à jour.

Pour mettre à jour les fichiers de script

- Pour les fichiers de script stockés localement, pour télécharger le fichier .zip de script mis à jour, utilisez la Amazon GameLift Servers console ou la `update-script` commande.
- Pour les fichiers de script stockés dans un compartiment Amazon S3, téléchargez les fichiers de script mis à jour dans le compartiment S3. Amazon GameLift Servers vérifie régulièrement les fichiers de script mis à jour et les récupère directement depuis le compartiment S3.

Configuration d'un parc d'hébergement avec Amazon GameLift Servers

Dans cette section, vous trouverez des informations sur la conception, la création et la maintenance de Amazon GameLift Servers flottes destinées à héberger vos serveurs de jeu. Consultez [Amazon GameLift Servers options d'hébergement](#) pour en savoir plus sur les solutions d'hébergement proposées, notamment celles qui utilisent des EC2 flottes gérées, des flottes Anywhere autogérées pour votre matériel sur site et une solution hybride qui utilise les deux. Amazon GameLift Servers

Rubriques

- [Caractéristiques de la flotte](#)
- [Comment fonctionne Amazon GameLift Servers la création de flottes](#)
- [Amazon GameLift Servers EC2 flottes gérées](#)
- [Amazon GameLift Servers flottes de conteneurs gérées](#)
- [Amazon GameLift Servers Des flottes partout](#)
- [Abstraire un Amazon GameLift Servers désignation de la flotte avec un alias](#)

Caractéristiques de la flotte

Une Amazon GameLift Servers flotte est un ensemble de ressources informatiques qui gère vos serveurs de jeu et héberge des sessions de jeu pour les joueurs. Les flottes peuvent varier en fonction du type de ressources informatiques que vous utilisez et de la manière dont elles sont gérées. La taille d'une flotte, c'est-à-dire le nombre de sessions de jeu et de joueurs qu'elle peut supporter, dépend du nombre de ressources informatiques que vous lui donnez. Toutes les Amazon GameLift Servers flottes présentent les caractéristiques suivantes :

- Les processus du serveur de jeu qui s'exécutent sur toutes les flottes sont intégrés au SDK du serveur Amazon GameLift Servers et communiquent avec le Amazon GameLift Servers service de la même manière. Les serveurs de jeu signalent leur disponibilité pour héberger des sessions de jeu et des joueurs, répondre aux demandes de démarrage ou d'arrêt de sessions de jeu et pour d'autres interactions.
- Amazon GameLift Servers gère le placement des sessions de jeu pour toutes les flottes de la même manière. Amazon GameLift Servers suit l'état des serveurs de jeu d'une flotte et choisit parmi les serveurs de jeu disponibles pour héberger une nouvelle session de jeu. Ce processus est utilisé

si votre jeu place les sessions de jeu sur une seule flotte ou utilise une [file d'attente de sessions de jeu](#) pour équilibrer l'hébergement entre plusieurs flottes. Avec une file d'attente, vous pouvez également personnaliser les décisions de placement en tenant compte de facteurs tels que le coût des ressources et la latence.

- Toutes les flottes prennent en charge l'utilisation d'un FlexMatch système de matchmaking en collaboration avec une file d'attente de placement pour les sessions de jeu. Le Amazon GameLift Servers service reçoit les demandes de match des joueurs, forme les matchs et les transmet à la file d'attente des sessions de jeu pour trouver les serveurs de jeu disponibles.
- Amazon GameLift Servers recueille un large éventail de mesures relatives à la flotte. Il s'agit notamment des indicateurs de statut des ordinateurs et des processus du serveur, ainsi que des indicateurs d'utilisation des sessions de jeu et de l'activité des joueurs. Consultez la liste complète des indicateurs disponibles sur [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#).

Comment fonctionne Amazon GameLift Servers la création de flottes

Lorsque vous demandez une nouvelle flotte, Amazon GameLift Servers lance un flux de travail pour créer la ressource de flotte. Au fur et à mesure qu'il termine chaque étape du flux de travail, Amazon GameLift Servers met à jour le statut de la flotte et émet une série d'événements pour communiquer les progrès réalisés en vue de la création de la flotte.

Amazon GameLift Servers utilise deux types d'événements. Les événements de transition de l'état de la flotte marquent le moment où l'état de la flotte change. Les événements de création de flotte fournissent des marqueurs supplémentaires pour aider à résoudre les problèmes de création. Vous pouvez suivre tous les événements à l'aide de la Amazon GameLift Servers console ou en appelant l'opération Amazon GameLift Servers API [DescribeFleetEvents](#). Vous pouvez également suivre l'état de la flotte et de l'emplacement à l'aide de [DescribeFleetAttributes](#) ou [DescribeFleetLocationAttributes](#).

Amazon GameLift Servers EC2 flottes gérées

Amazon GameLift Servers EC2 les flottes gérées fournissent des ressources basées sur le cloud pour l'hébergement de production. Avec une flotte gérée, vous bénéficiez de la flexibilité, de la sécurité et de la fiabilité de AWS Cloud ressources optimisées pour l'hébergement de jeux multijoueurs. Amazon GameLift Servers fournit des outils de gestion d'hôte robustes.

Un EC2 parc géré est un ensemble d'instances Amazon Elastic Compute Cloud (Amazon EC2) qui Amazon GameLift Servers possède et exploite en fonction de votre configuration. Ces instances sont physiquement situées dans des zones prises en charge Régions AWS ou des zones locales. Lorsque vous créez une flotte, vous choisissez un type d' EC2 instance qui répond aux exigences de votre serveur de jeu en termes de puissance de calcul, de mémoire, de stockage et de capacités réseau.

Au démarrage de chaque instance de la flotte, Amazon GameLift Servers déploie la version de votre serveur de jeu avec l'environnement d'exécution requis. L'environnement d'exécution utilise la dernière version d'Amazon Machine Image (AMI) disponible lors de la création du parc. Toutes les instances du parc utilisent la même version d'AMI.

 Note

À titre de bonne pratique, nous vous recommandons de remplacer vos flottes tous les 30 jours afin de garantir un environnement d' up-to-date exécution sécurisé pour vos serveurs de jeux hébergés. Cela nécessite la création d'une nouvelle flotte et la migration du trafic des joueurs vers celle-ci. Pour plus d'informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

Après avoir installé l'environnement d'exécution et créé le build de votre serveur de jeu sur une instance, Amazon GameLift Servers lance les processus du serveur de jeu. Chaque processus du serveur de jeu établit une connexion au Amazon GameLift Servers service, indique qu'il est prêt à héberger une session de jeu et commence à communiquer l'état de santé. Amazon GameLift Servers peut ensuite demander au processus du serveur de démarrer une session de jeu.

Outre le déploiement de la flotte, Amazon GameLift Servers gère les tâches de gestion des hôtes suivantes afin que vous n'ayez pas à :

- Suit l'état de tous les ordinateurs du parc et remplace les ordinateurs périmés ou défectueux.
- Gère l'authentification pour la communication entre les processus du serveur et le Amazon GameLift Servers service.
- Démarre et arrête automatiquement les processus du serveur de jeu sur chaque ordinateur, en fonction de votre configuration d'exécution.
- Propose des outils de mise à l'échelle des capacités, notamment le dimensionnement automatique qui ajuste dynamiquement la capacité de la flotte pour répondre à la demande des joueurs.
- Indique les indicateurs de performance pour les EC2 instances de la flotte.

Consultez les rubriques suivantes pour savoir comment configurer et gérer des EC2 flottes gérées :

- [Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Managed EC2](#)
- [Création d'une EC2 flotte Amazon GameLift Servers gérée](#)
- [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#)
- [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#)
- [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#)

Flux de travail EC2 de création de flotte géré

Pour les flottes gérées, Amazon GameLift Servers configure les ressources de la flotte et déploie également un ensemble de ressources informatiques avec le logiciel de votre serveur de jeu installé et en cours d'exécution. Lorsque le processus de création est terminé et réussi, la flotte dispose d'une EC2 instance active dans la région d'origine de la flotte et d'une instance dans chacune des régions éloignées de la flotte. Toutes les instances dotées d'un jeu sont prêtes à héberger des sessions de jeu.

1. Amazon GameLift Servers crée la ressource de flotte dans la région d'origine de la flotte et définit la capacité souhaitée pour chaque site à une (1) instance. Le statut de la flotte et de l'emplacement est défini sur Nouveau.
2. Amazon GameLift Servers commence à consigner les événements dans le journal des événements de la flotte.
3. Amazon GameLift Servers définit le statut de la flotte sur Téléchargement et commence à préparer le logiciel du serveur de jeu pour le déploiement.
 - a. Récupère le build du serveur de jeu téléchargé et extrait les fichiers compressés.
 - b. Exécute les scripts d'installation, s'ils sont fournis.
 - c. Définit l'état du parc sur Validation et commence à vérifier qu'aucune erreur ne s'est produite lors du téléchargement et de l'installation des fichiers de compilation.
4. Amazon GameLift Servers définit le statut de la flotte sur Building, configure le matériel de la flotte et alloue une instance pour chaque EC2 instance de flotte.
5. Amazon GameLift Servers définit le statut de la flotte sur Activation. Lance un processus de serveur de jeu sur chaque instance (en fonction des instructions d'exécution de la flotte) et teste la connectivité entre le build et le Amazon GameLift Servers service.

6. Lorsque le serveur de jeu traite chaque instance, établissez une connexion et signalez que vous êtes prêt à héberger des sessions de jeu, Amazon GameLift Servers définit le statut du parc et de l'emplacement sur Actif. À ce stade, la flotte est considérée comme prête à accueillir des sessions de jeu.

Création d'une EC2 flotte Amazon GameLift Servers gérée

Cette rubrique explique comment créer une EC2 flotte Amazon GameLift Servers gérée. Les flottes gérées utilisent des instances de calcul Amazon Elastic Compute Cloud (Amazon EC2) optimisées pour l'hébergement de jeux multijoueurs. Vous pouvez créer des flottes gérées qui déploient des ordinateurs à l'échelle mondiale vers des Zones Régions AWS Locales prises en charge par Amazon GameLift Servers

Lorsque vous créez une nouvelle ressource de EC2 flotte gérée, vous lancez immédiatement la première phase de création de flotte. La création de flottes gérées passe par plusieurs phases : Amazon GameLift Servers déploiement d'une EC2 instance, installation d'un environnement d'exécution et création de votre serveur de jeu à partir de l'instance, puis lancement des serveurs de jeu. En fonction de l'environnement d'exécution requis par la création de votre serveur de jeu, Amazon GameLift Servers déploie la dernière version d'Amazon Machine Image (AMI) au moment de la création de la flotte (et toutes les futures instances de la flotte utiliseront la même version). Vous pouvez surveiller l'état d'une flotte dans la console ou à l'aide du AWS Command Line Interface (AWS CLI). Lorsqu'une flotte est prête à accueillir des sessions de jeu, son statut passe à ACTIVE. Pour plus d'informations sur la création de flottes gérées, consultez les rubriques suivantes :

- [Comment fonctionne Amazon GameLift Servers la création de flottes](#)
- [Déboguer les incidents de flotte Amazon GameLift Servers](#)

Note

À titre de bonne pratique, nous vous recommandons de remplacer vos flottes tous les 30 jours afin de garantir un environnement d' up-to-date exécution sécurisé pour vos serveurs de jeux hébergés. Cela nécessite la création d'une nouvelle flotte et la migration du trafic des joueurs vers celle-ci. Pour plus d'informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

Pour créer une EC2 flotte gérée

Utilisez la Amazon GameLift Servers console ou le AWS Command Line Interface (AWS CLI) pour créer une EC2 flotte gérée.

Console

Dans la [Amazon GameLift Servers console](#), utilisez le volet de navigation pour ouvrir la page Fleets. Choisissez Créer une flotte pour démarrer le flux de travail de création de flotte.

Étape 1 Choisissez le type de calcul

Sélectionnez l' EC2option Géré, puis cliquez sur Suivant.

Étape 2 Définir les détails de la flotte

Au cours de cette étape, définissez certains paramètres pour l'ensemble de la flotte.

 Pour une configuration de flotte minimale :

- Entrez le nom de la flotte.
- Choisissez un type binaire et spécifiez un build ou un script téléchargé.
- Ignorez les sections sur les détails et les balises supplémentaires.

1. Remplissez la section Détails de la flotte :

- a. Entrez un nom de flotte. Nous vous recommandons d'utiliser un modèle de dénomination des flottes qui facilite l'identification des types de flotte lors de la consultation des listes de flottes.
- b. Fournissez une brève description de la flotte.
- c. Pour le type binaire, sélectionnez Build pour indiquer que vous déployez une version de serveur de jeu personnalisée, ou sélectionnez Script si vous déployez Amazon GameLift Servers Realtime sur cette flotte. Sélectionnez une version ou un script téléchargé dans la liste déroulante.

2. (Facultatif) Définissez des informations supplémentaires selon vos besoins.

- a. Si le fichier exécutable de votre serveur de jeu doit accéder à d'autres AWS ressources de votre compte, spécifiez un rôle d'instance IAM doté des autorisations nécessaires. Pour plus d'informations, notamment sur la manière d'autoriser d'autres applications côté serveur (telles que CloudWatch l'agent), consultez. [Communiquez](#)

avec les autres AWS ressources de vos flottes Ce paramètre ne peut pas être modifié une fois que vous avez créé la flotte.

Vous devez créer le rôle avant de créer une flotte qui l'utilise. En outre, pour créer une flotte avec un rôle d'instance, votre AWS utilisateur doit disposer de l'`PassRole` autorisation IAM (voir [Exemples d'autorisations IAM pour Amazon GameLift Servers](#)).

- b. Activez l'option Générer un certificat TLS pour configurer l'authentification et le cryptage de votre jeu. Les clients du jeu utilisent ce certificat pour authentifier un serveur de jeu lors de la connexion et crypter toutes les communications client/serveur. Pour chaque instance d'un parc compatible TLS, Amazon GameLift Servers crée aussi une nouvelle entrée DNS avec le certificat. Ce paramètre ne peut pas être modifié une fois que vous avez créé la flotte.
 - c. Si vous souhaitez combiner les données métriques de ce parc et d'autres, spécifiez un nom de groupe de métriques. Utilisez le même nom de groupe de mesures pour toutes les flottes que vous souhaitez combiner. Consultez les métriques du groupe de métriques pour voir les données agrégées.
3. (Facultatif) Ajoutez des balises à la ressource de flotte. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez. Attribuez des balises aux AWS ressources que vous souhaitez classer de manière utile, par exemple par objectif, propriétaire ou environnement. Choisissez Ajouter un nouveau tag pour chaque tag que vous souhaitez ajouter.
4. Choisissez Next pour continuer le flux de travail.

Étape 3 Définir les détails de l'instance

Au cours de cette étape, spécifiez le type de ressources d'hébergement à utiliser et l'endroit où vous souhaitez les déployer. En choisissant plusieurs emplacements, vous pouvez déployer votre serveur de jeu sur une zone géographique plus large, ce qui le rapproche de vos joueurs et minimise le temps de latence. Tous les types d' EC2 instances ne sont pas disponibles dans tous les emplacements.

 Pour une configuration de flotte minimale :

- N'ajoutez pas de sites distants.

- Définissez le type de flotte sur « À la demande ». Les flottes Spot nécessitent des travaux de configuration supplémentaires.
- Définissez le type d'instance sur « c5.large ». Ce type d'instance couramment utilisé est disponible partout Régions AWS.

1. Dans Déploiement de l'instance, spécifiez les emplacements et le type de flotte.

- a. Sélectionnez un ou plusieurs sites supplémentaires sur lesquels vous souhaitez déployer des instances de flotte. Ces sites distants sont ajoutés à l'emplacement d'origine de la flotte (qui est présélectionné), c'est-à-dire l' Région AWS endroit où vous créez cette flotte. Vous pouvez sélectionner des sites distants parmi tous les sites Régions AWS et les Zones Amazon GameLift Servers Locales compatibles.

Pour en savoir plus sur les emplacements pris en charge, notamment sur la façon d'utiliser un emplacement Région AWS qui n'est pas activé par défaut, consultez la section [Amazon GameLift Serverspoints de service](#) relative à l'hébergement géré. Passez également en revue les Amazon GameLift Servers [quotas](#) d'emplacements par flotte.

- b. Choisissez d'utiliser des instances à la demande ou des instances ponctuelles pour ce parc. Pour plus d'informations sur les types de flottes, consultez [Instances à la demande et instances ponctuelles](#).

2. Choisissez une configuration d' EC2 instance Amazon qui répond à vos besoins et qui est disponible dans tous les sites que vous avez sélectionnés. Cette liste est filtrée en fonction de votre position actuelle et des types de flotte sélectionnés. Vous pouvez le filtrer davantage en fonction d'autres facteurs tels que le type d'instance, la famille et l'architecture. Après avoir créé le parc, vous ne pouvez pas modifier le type d'instance.

Certains sites proposent des options de type d'instance limitées. Si le type d'instance que vous préférez n'est pas disponible pour tous les sites, choisissez la valeur de disponibilité des emplacements pour afficher tous les détails. Pour prendre en charge tous les sites, vous devrez peut-être créer des flottes distinctes avec différents types d'instances.

Pour plus d'informations sur le choix d'un type d'instance, consultez [Types d'instances](#). Pour en savoir plus sur les architectures Amazon EC2 Arm, consultez les [AWS sections Processeur Graviton](#) et [types d' EC2instances Amazon](#). Pour obtenir la liste complète des

types d'instances pris en charge par Amazon GameLift Servers, consultez la référence d'API pour [EC2InstanceType](#)(`CreateFleet()`).

 Note

Les instances Graviton Arm nécessitent un Amazon GameLift Servers serveur basé sur le système d'exploitation Linux. Le SDK Server 5.1.1 ou une version ultérieure est requis pour C++ et C#. Le SDK Server 5.0 ou une version ultérieure est requis pour Go. Ces instances ne prennent pas out-of-the-box en charge l'installation de Mono sur Amazon Linux 2023 (AL2023) ou Amazon Linux 2 (AL2).

3. Choisissez Next pour continuer le flux de travail.

Étape 4 Configuration de l'environnement d'exécution

Dans cette étape, décrivez comment vous souhaitez que chaque instance du parc exécute le logiciel de votre serveur de jeu. Définissez un élément de ligne de processus serveur distinct pour chaque exécutable à exécuter sur une instance, et déterminez le nombre de processus serveur à exécuter simultanément. Ouvrez des ports sur chaque instance pour permettre aux joueurs de se connecter directement aux serveurs de jeu. Vous pouvez mettre à jour ces paramètres de flotte à tout moment.

 Pour une configuration de flotte minimale :

- Définissez une seule ligne de processus serveur pour le fichier exécutable de votre serveur de jeu. Si votre serveur de jeu nécessite l'exécution d'autres processus, créez également une définition pour chacun d'entre eux.
- Utilisez le nombre de processus simultanés par défaut (1) pour chaque rubrique.
- Ignorez les paramètres d'activation de la session de jeu.
- Spécifiez un numéro de port unique.
- Ignorez les paramètres des ressources de la session de jeu.

1. Créez une configuration d'exécution pour indiquer Amazon GameLift Servers comment exécuter les processus du serveur sur chaque instance du parc. Vous pouvez modifier la configuration d'exécution d'une flotte à tout moment après le déploiement.

- a. Entrez le chemin de lancement d'un fichier exécutable dans votre build. Sur les instances Windows, les exécutables du serveur de jeu sont intégrés au chemin. C : \game Sur les instances Linux, les serveurs de jeu sont conçus pour /local/game. Exemples : **C:\game\MyGame\server.exe/local/game/MyGame/server.exe**, ou **MyRealtimeLaunchScript.js**.
- b. Entrez les paramètres de lancement facultatifs à transmettre au fichier exécutable de votre jeu. Exemple: **+sv_port 33435 +start_lobby**.
- c. Spécifiez le nombre de processus simultanés à exécuter sur chaque instance. Pour un exécutable sur un serveur de jeu, chaque processus peut héberger une session de jeu, de sorte que les processus simultanés déterminent le nombre de sessions de jeu que l'instance peut héberger simultanément.

Passez en revue les Amazon GameLift Servers [quotas](#) relatifs aux processus du serveur par instance. Ces quotas s'appliquent au nombre total de processus simultanés pour toutes les configurations. Si vous configurez la flotte pour les dépasser, elle ne peut pas être activée.

2. Utilisez les paramètres d'activation de session de jeu par défaut ou personnalisez-les pour votre jeu. Si la configuration d'exécution nécessite plusieurs processus de serveur de jeu simultanés par instance, ces paramètres déterminent la rapidité avec laquelle les nouvelles sessions de jeu peuvent démarrer.
 - a. Définissez le nombre maximal d'activations simultanées de session de jeu pour limiter le nombre de serveurs de jeu sur une instance qui préparent une nouvelle session de jeu. Ce paramètre est utile lorsque le lancement de plusieurs nouvelles sessions de jeu est gourmand en ressources et peut avoir un impact sur les performances des autres sessions de jeu en cours d'exécution.
 - b. Définissez le nouveau délai d'activation pour qu'il reflète le temps maximum qu'une nouvelle session de jeu doit prendre pour terminer l'activation et signaler que les joueurs sont prêts à héberger. Amazon GameLift Servers met fin à l'activation d'une session de jeu si elle dépasse cette valeur.
3. Ouvrez les paramètres de EC2 port pour permettre au trafic entrant d'accéder aux processus du serveur de la flotte. Ces paramètres ne sont pas obligatoires pour créer une flotte, mais vous devez les définir avant que les joueurs puissent se connecter aux sessions de jeu de la flotte.

Pour chaque paramètre de port, choisissez le type de protocole de transfert de données à utiliser pour la communication entre le client de jeu et le serveur de jeu. Fournissez une plage de ports (au format `nnnnn[-nnnn]`) et une plage d'adresses IP en utilisant la notation CIDR (telle que celle `0.0.0.0/0` qui autorise l'accès à n'importe qui).

Si vous devez définir plusieurs plages non consécutives, créez plusieurs paramètres de port.

4. Spécifiez les paramètres facultatifs des ressources de session de jeu. Vous pouvez mettre à jour ces paramètres à tout moment après le déploiement.
 - a. Activez ou désactivez la politique de protection Game Scaling pour toutes les instances de la flotte. Lors d'un événement de réduction, les instances de flotte protégées Amazon GameLift Servers ne seront pas résiliées si elles hébergent des sessions de jeu actives.
 - b. Définissez une limite maximale de création de ressources si vous souhaitez limiter le nombre de sessions de jeu qu'un joueur peut créer pendant une période donnée.
5. Choisissez Next pour continuer le flux de travail.

Étape 5 Réviser et créer

Vérifiez vos paramètres avant de créer la flotte. Bien que certains paramètres puissent être mis à jour ultérieurement (voir [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#)), les modifications des paramètres suivants ne sont pas autorisées une fois la flotte créée :

- Type de calcul : vous ne pouvez pas convertir un EC2 parc géré en un parc Anywhere.
- Build ou script : pour déployer une mise à jour du build ou du script de votre serveur de jeu, vous devez créer une nouvelle flotte.
- Options supplémentaires, notamment le rôle d'instance et la génération de certificats TLS.
- Détails de l'instance, y compris le type de flotte (ponctuel ou à la demande) et le type d' EC2 instance.

Lorsque vous êtes prêt à déployer la nouvelle flotte, choisissez Create. Amazon GameLift Servers lance immédiatement le processus d'activation de la flotte, en attribuant un identifiant unique et en plaçant la flotte dans NEW son statut. Suivez la progression de la flotte sur la page Flottes. Consultez la page de détails de la flotte et accédez à l'onglet Événements.

Vous pouvez ajuster la capacité d'hébergement d'une flotte une fois que celle-ci a atteint le statut ACTIF. Amazon GameLift Servers déploie initialement une flotte avec une seule instance dans chaque emplacement du parc, et vous ajustez la capacité en ajoutant des instances à chaque emplacement. Pour de plus amples informations, veuillez consulter [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).

AWS CLI

Utilisez la [create-fleet](#) commande pour créer un parc de types de calcul EC2. Amazon GameLift Servers crée la ressource de flotte dans votre valeur par défaut actuelle Région AWS (ou vous pouvez ajouter une balise `--region` pour en spécifier une autre Région AWS).

Créez une flotte gérée minimale

L'exemple de demande suivant crée une nouvelle flotte avec les paramètres minimaux requis pour déployer une flotte avec des serveurs de jeu actifs auxquels les clients du jeu peuvent se connecter. La nouvelle flotte présente les caractéristiques suivantes :

- Il spécifie une version de serveur de jeu, qui a été téléchargée vers Amazon GameLift Servers et dont le READY statut est en cours.
- Il utilise des instances à la demande `c5.large` avec un système d'exploitation correspondant à la version de jeu sélectionnée.
- Il définit le siège Région AWS de la flotte `us-west-2` et déploie les instances dans cette région uniquement.
- Selon la configuration d'exécution, chaque ordinateur du parc exécute un processus de serveur de jeu, ce qui signifie que chaque ordinateur ne peut héberger qu'une seule session de jeu à la fois. Le délai d'activation des sessions de jeu est défini sur la valeur par défaut de 300 secondes, et il n'y a aucune limite quant au nombre d'activations simultanées.
- Les joueurs peuvent se connecter aux serveurs de jeu en utilisant un paramètre de port unique défini sur 33435.
- Toutes les autres fonctionnalités sont désactivées ou utilisent les paramètres par défaut.

```
aws gamelift create-fleet \  
  --name MinimalFleet123 \  
  --description "A basic test fleet" \  
  --region us-west-2 \  
  --ec2-instance-type c5.large \  
  --game-session-length 300 \  
  --game-session-port 33435
```

```
--fleet-type ON_DEMAND \  
--build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \  
--runtime-configuration "ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated\  
\MultiplayerSampleProjectLauncher_Server.exe, ConcurrentExecutions=10}]" \  
--ec2-inbound-permissions  
"FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
```

Créez une flotte gérée entièrement configurée

L'exemple de demande suivant crée un parc de production avec des paramètres pour toutes les fonctionnalités facultatives. La nouvelle flotte présente les caractéristiques suivantes :

- Il spécifie une version de serveur de jeu, qui a été téléchargée vers Amazon GameLift Servers et dont le READY statut est en cours.
- Il utilise des instances à la demande c5.large avec le système d'exploitation correspondant à la version de jeu sélectionnée.
- Il définit le siège Région AWS de la flotte us-west-2 et déploie les instances dans la région d'origine et sur un site sa-east-1 distant.
- Sur la base de la configuration d'exécution :
 - Chaque ordinateur de la flotte exécute 10 processus de serveur de jeu avec les mêmes paramètres de lancement, ce qui signifie que chaque ordinateur peut héberger jusqu'à 10 sessions de jeu simultanément.
 - Sur chaque ordinateur, seules deux sessions de jeu peuvent être activées en même temps. Les sessions de jeu activées doivent être prêtes à accueillir les joueurs dans un délai de 300 secondes (5 minutes) sous peine d'être interrompues.
- Les joueurs peuvent se connecter aux serveurs de jeu en utilisant un port dans la plage suivante 33435 to 33535.
- Il génère un certificat TLS pour les communications cryptées entre le client du jeu et le serveur.
- La protection des sessions de jeu est activée pour toutes les sessions de jeu de la flotte.
- Les joueurs individuels sont limités à la création de trois nouvelles sessions de jeu sur une période de 15 minutes.
- Les métriques de cette flotte sont incluses dans le groupe de métriques AMERfleets, qui (dans cet exemple) regroupe les métriques d'un groupe de flottes en Amérique du Nord, centrale et du Sud.

```
aws gamelift create-fleet \  

```

```
--name ProdFleet123 \
--description "A fully configured prod fleet" \
--ec2-instance-type c5.large \
--region us-west-2 \
--locations "Location=sa-east-1" \
--fleet-type ON_DEMAND \
--build-id build-1111aaaa-22bb-33cc-44dd-5555eeee66ff \
--certificate-configuration "CertificateType=GENERATED" \
--runtime-configuration "GameSessionActivationTimeoutSeconds=300,
MaxConcurrentGameSessionActivations=2, ServerProcesses=[{LaunchPath=C:\game
\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe, Parameters=+sv_port
33435 +start_lobby, ConcurrentExecutions=10}]" \
--new-game-session-protection-policy "FullProtection" \
--resource-creation-limit-policy "NewGameSessionsPerCreator=3,
PolicyPeriodInMinutes=15" \
--ec2-inbound-permissions
"FromPort=33435,ToPort=33535,IpRange=0.0.0.0/0,Protocol=UDP" \
--metric-groups "AMERfleets"
```

Si la demande de création de flotte aboutit, Amazon GameLift Servers renvoie un ensemble d'attributs de flotte comprenant les paramètres de configuration que vous avez demandés et un nouvel identifiant de flotte. Amazon GameLift Servers lance ensuite le processus d'activation de la flotte et définit le statut de la flotte et les statuts de localisation sur Nouveau. Vous pouvez suivre le statut de la flotte et consulter d'autres d'informations de flotte à l'aide de ces commandes d'interface de ligne de commande :

- [describe-fleet-events](#)
- [describe-fleet-attributes](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-runtime-configuration](#)
- [describe-fleet-location-attributes](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

Vous pouvez changer la capacité de la flotte et d'autres paramètres de configuration, si nécessaire, à l'aide des commandes suivantes :

- [update-fleet-attributes](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Personnalisez votre Amazon GameLift Servers EC2 flottes gérées

Les rubriques de cette section décrivent certaines des personnalisations que vous pouvez apporter lors de la création d'une solution d'hébergement avec Amazon GameLift Servers EC2 flottes gérées. Ils fournissent des conseils et des bonnes pratiques pour créer et configurer une flotte pour héberger votre jeu.

Les décisions requises incluent :

- Où devez-vous déployer les ressources d'hébergement ? La latence est un facteur important dans le choix des emplacements de votre flotte, mais le coût varie également en fonction du lieu.
- Quel type d' EC2 instance convient le mieux à votre jeu ? Choisissez parmi les types d'instances disponibles sur tous les sites de votre flotte afin d'utiliser la meilleure combinaison d'architecture informatique, de mémoire, de stockage et de capacité réseau.
- De quelle taille de type d'instance avez-vous besoin ? Choisissez une taille de type d'instance en fonction des besoins en ressources (mémoire et processeur) du logiciel de votre serveur de jeu et d'autres facteurs.
- Votre flotte doit-elle utiliser des instances à la demande ou ponctuelles ? Déterminez si vous pouvez profiter d'une réduction de la tarification au comptant, compte tenu de la manière dont Amazon GameLift Servers protège contre le risque d'interruption des sessions de jeu.
- Comment souhaitez-vous que le logiciel de votre serveur de jeu s'exécute sur chaque instance de flotte ? La configuration d'exécution indique Amazon GameLift Servers quel logiciel serveur exécuter et comment.

Rubriques

- [Choisissez les ressources de calcul pour un parc géré](#)
- [Gérez comment Amazon GameLift Servers lance des serveurs de jeu](#)

Choisissez les ressources de calcul pour un parc géré

Pour déployer vos serveurs de jeu et héberger des sessions de jeu dans le cloud, Amazon GameLift Servers fournit des flottes gérées qui utilisent les [ressources Amazon Elastic Compute Cloud \(Amazon EC2\)](#) appelées instances. Utilisez les rubriques suivantes pour déterminer le type d'EC2 instances que vous souhaitez utiliser pour votre solution d'hébergement géré et comment les configurer pour exécuter le logiciel de votre serveur de jeu.

Note

Si vous envisagez d'utiliser des ressources d'hébergement dont vous êtes propriétaire, qu'il s'agisse de matériel sur site ou d'un autre hébergement basé sur le cloud, envisagez des options d'hébergement hybride avec Amazon GameLift Servers Anywhere. Consultez [Configuration d'un parc d'hébergement avec Amazon GameLift Servers](#).

Rubriques

- [Emplacement de la flotte](#)
- [Instances à la demande et instances ponctuelles](#)
- [Operating systems](#)
- [Types d'instances](#)
- [Quotas de service](#)

Emplacement de la flotte

Tenez compte des emplacements géographiques dans lesquels vous prévoyez de déployer vos serveurs de jeu. La disponibilité du type d'instance varie en fonction Région AWS de la zone locale.

Pour les flottes multisites, la disponibilité des instances et les quotas dépendent de la combinaison de la région d'origine de la flotte et des sites distants sélectionnés. Pour plus d'informations sur les emplacements des flottes, consultez [Amazon GameLift Servers points de service](#).

Envisagez [Balises ping UDP](#) de collecter des données de latence du réseau dans différentes zones géographiques afin d'anticiper la latence entre les appareils des joueurs et les emplacements potentiels de la flotte. Ces points de terminaison spéciaux acceptent les messages UDP au lieu des pings ICMP traditionnels, fournissant des mesures de latence précises pour vous aider à sélectionner les emplacements de flotte optimaux.

Pour les flottes Amazon GameLift Servers Anywhere, vous déterminez l'emplacement de votre matériel physique. Pour plus d'informations sur les emplacements personnalisés, consultez [Des emplacements pour Amazon GameLift Servers n'importe où](#).

Instances à la demande et instances ponctuelles

Les instances EC2 Amazon On-Demand et les instances Spot offrent le même matériel et les mêmes performances, mais leur disponibilité et leur coût diffèrent.

On-Demand instances

Vous pouvez acquérir une instance à la demande lorsque vous en avez besoin et la conserver aussi longtemps que vous le souhaitez. Les instances à la demande ont un coût fixe, ce qui signifie que vous payez en fonction de la durée pendant laquelle vous les utilisez, et il n'y a aucun engagement à long terme.

Instances Spot

Les instances Spot peuvent constituer une alternative rentable aux instances à la demande en utilisant la capacité AWS informatique inutilisée. Les prix des instances Spot fluctuent en fonction de l'offre et de la demande pour chaque type d'instance sur chaque site. AWS peut interrompre les instances Spot chaque fois qu'elle a besoin de récupérer sa capacité. Amazon GameLift Servers utilise les files d'attente et l'FleetIQ algorithme pour déterminer qui AWS va interrompre une instance Spot, il met l'instance en état de recyclage. Ensuite, lorsqu'aucune session de jeu n'est active sur l'instance, Amazon GameLift Servers essaie de la remplacer.

Pour plus d'informations sur l'utilisation des instances Spot, consultez [Concevoir une file d'attente pour les instances Spot](#).

Operating systems

Amazon GameLift Servers les instances prennent en charge les versions de serveurs de jeu qui s'exécutent sous Microsoft Windows ou Amazon Linux. Lorsque vous téléchargez une version de jeu sur Amazon GameLift Servers, spécifiez le système d'exploitation du jeu. Lorsque vous créez une EC2 flotte Amazon pour déployer le build du jeu, les instances Amazon GameLift Servers sont automatiquement configurées avec le système d'exploitation du build. Pour plus d'informations sur les systèmes d'exploitation de serveurs de jeu pris en charge, consultez [Obtenez des outils Amazon GameLift Servers de développement](#).

Lorsque vous utilisez une flotte Amazon GameLift Servers Anywhere, vous pouvez utiliser n'importe quel système d'exploitation compatible avec votre matériel. Amazon GameLift Servers Les flottes

Anywhere vous obligent à déployer votre version de jeu sur le matériel tout Amazon GameLift Servers en gérant vos ressources en un seul endroit.

Types d'instances

Le type d'instance d'une EC2 flotte Amazon détermine le type de matériel utilisé par les instances. Les différents types d'instances offrent différentes combinaisons de puissance de calcul, de mémoire, de stockage et de capacités réseau.

Lorsque vous choisissez parmi les types d'instances disponibles pour votre jeu, tenez compte des points suivants :

- Architecture de calcul de votre serveur de jeu : x64 ou Arm (AWS Graviton).

Note

Les instances Graviton Arm nécessitent un Amazon GameLift Servers serveur basé sur le système d'exploitation Linux. Le SDK Server 5.1.1 ou une version ultérieure est requis pour C++ et C#. Le SDK Server 5.0 ou une version ultérieure est requis pour Go. Ces instances ne prennent pas out-of-the-box en charge l'installation de Mono sur Amazon Linux 2023 (AL2023) ou Amazon Linux 2 (AL2).

- Les exigences en matière de calcul, de mémoire et de stockage de la version de votre serveur de jeu.
- Le nombre de processus serveur que vous prévoyez d'exécuter par instance.

En utilisant un type d'instance plus important, vous pourrez peut-être exécuter plusieurs processus de serveur sur chaque instance. Cela peut réduire le nombre d'instances nécessaires pour répondre à la demande des joueurs.

Pour plus d'informations, consultez:

- À propos des types d'instances, consultez [Amazon EC2 Instance Types](#).
- À propos de l'exécution de plusieurs processus par instance, voir [Gérez comment Amazon GameLift Servers lance des serveurs de jeu](#).

Quotas de service

Pour voir les quotas de service par défaut et les quotas actuels pour votre service Compte AWS, procédez comme suit : Amazon GameLift Servers

- Pour obtenir des informations générales sur les quotas de service pour Amazon GameLift Servers, voir les [Amazon GameLift Servers points de terminaison et les quotas](#) dans le Références générales AWS.
- Pour obtenir la liste des types d'instances disponibles par emplacement pour votre compte, ouvrez la page [Quotas de service](#) de la Amazon GameLift Servers console. Cette page affiche également l'utilisation actuelle de votre compte pour chaque type d'instance dans chaque emplacement.
- Pour obtenir la liste des quotas actuels de votre compte pour les types d'instances par région, exécutez la commande AWS Command Line Interface (AWS CLI) [describe-ec2-instance-limits](#). Cette commande renvoie le nombre d'instances actives que vous avez dans votre région par défaut (ou dans une autre région que vous spécifiez).

Alors que vous vous préparez à lancer votre jeu, remplissez un questionnaire de lancement dans la [Amazon GameLift Servers console](#). L'Amazon GameLift Servers équipe utilise le questionnaire de lancement pour déterminer les quotas et les limites appropriés pour votre jeu.

Gérez comment Amazon GameLift Servers lance des serveurs de jeu

Vous pouvez configurer la configuration d'exécution d'une EC2 flotte gérée pour exécuter plusieurs processus de serveur de jeu par instance. Cela permet d'utiliser vos ressources d'hébergement de manière plus efficace.

Comment une flotte gère plusieurs processus

Amazon GameLift Servers utilise la configuration d'exécution d'un parc pour déterminer le type et le nombre de processus à exécuter sur chaque instance. Une configuration d'exécution contient au moins une configuration de processus serveur qui représente un exécutable de serveur de jeu. Vous pouvez définir des configurations de processus de serveur supplémentaires pour exécuter d'autres types de processus liés à votre jeu. Chaque configuration de processus serveur contient les informations suivantes :

- Nom de fichier et chemin d'accès d'un exécutable dans votre build de jeu.
- (Facultatif) Paramètres à transmettre au processus serveur au lancement.
- Le nombre de processus à exécuter simultanément.

Lorsqu'une instance du parc est activée, elle lance l'ensemble des processus serveur définis dans la configuration d'exécution. Avec de multiples processus, Amazon GameLift Servers échelonne le lancement de chaque processus. Les processus du serveur ont une durée de vie limitée. À leur fin, Amazon GameLift Servers lance de nouveaux processus pour conserver le nombre et le type de processus serveur définis dans la configuration d'exécution.

Vous pouvez modifier la configuration d'exécution d'une flotte à tout moment en ajoutant, en modifiant ou en supprimant des configurations de processus serveur. Chaque instance vérifie régulièrement les mises à jour de la configuration d'exécution du parc afin de mettre en œuvre les modifications. Voici comment Amazon GameLift Servers adopte les modifications de configuration d'exécution :

1. L'instance envoie une demande à Amazon GameLift Servers pour la dernière version de la configuration d'exécution.
2. L'instance compare ses processus actifs à la dernière configuration d'exécution, puis effectue les opérations suivantes :
 - Si la configuration d'exécution mise à jour supprime un type de processus serveur, les processus serveur actifs de ce type continuent de s'exécuter jusqu'à leur fin. L'instance ne remplace pas ces processus de serveur.
 - Si la configuration d'exécution mise à jour réduit le nombre de processus simultanés pour un type de processus serveur, les processus serveur excédentaires de ce type continuent de s'exécuter jusqu'à leur fin. L'instance ne remplace pas ces processus serveur excédentaires.
 - Si la configuration d'exécution mise à jour ajoute un nouveau type de processus serveur ou augmente le nombre de processus simultanés pour un type existant, l'instance lance de nouveaux processus serveur, jusqu'à Amazon GameLift Servers maximum. Dans ce cas, l'instance lance de nouveaux processus serveur lorsque les processus existants prennent fin.

Optimisation d'une flotte pour de multiples processus

Pour utiliser plusieurs processus sur une flotte, procédez comme suit :

- [Créez un build](#) contenant les exécutables du serveur de jeu que vous souhaitez déployer dans une flotte, puis téléchargez le build sur Amazon GameLift Servers. Tous les serveurs de jeu d'une version doivent fonctionner sur la même plateforme et utiliser le SDK du serveur pour Amazon GameLift Servers.
- Créez une configuration d'exécution avec une ou plusieurs configurations de processus serveur et plusieurs processus simultanés.
- Intégrez les clients de jeu avec le AWS SDK version 2016-08-04 ou ultérieure.

Pour optimiser les performances de votre flotte, nous vous recommandons de procéder comme suit :

- Gérez les scénarios d'arrêt des processus du serveur afin que Amazon GameLift Servers peut recycler les processus de manière efficace. Par exemple :
 - Ajoutez une procédure d'arrêt au code de votre serveur de jeu qui appelle l'API du `serveurProcessEnding()`.
 - Implémentez la fonction de rappel `OnProcessTerminate()` dans le code de votre serveur de jeu pour gérer les demandes de résiliation provenant de Amazon GameLift Servers.
- Assurez-vous que Amazon GameLift Servers arrête et relance les processus du serveur défectueux. Signalez l'état de santé à Amazon GameLift Servers en implémentant la fonction de `OnHealthCheck()` rappel dans le code de votre serveur de jeu. Amazon GameLift Servers arrête automatiquement les processus du serveur signalés comme défectueux pendant trois rapports consécutifs. Si vous ne l'implémentez pas `OnHealthCheck()`, alors Amazon GameLift Servers suppose qu'un processus serveur est sain, sauf s'il ne répond pas à une communication.

Choisissez le nombre de processus par instance

Lorsque vous déterminez le nombre de processus simultanés à exécuter sur une instance, gardez à l'esprit les points suivants :

- Amazon GameLift Servers limite chaque instance à un [nombre maximum de processus simultanés](#). La somme de tous les processus simultanés pour les configurations de processus de serveur d'un parc ne peut pas dépasser ce quota.
- Pour maintenir des niveaux de performance acceptables, le type d' EC2 instance Amazon peut limiter le nombre de processus pouvant être exécutés simultanément. Testez différentes configurations pour votre jeu afin de trouver le nombre de processus correspondant à votre type d'instance préféré.
- Amazon GameLift Servers n'exécute pas plus de processus simultanés que le nombre total de processus configurés. Cela signifie que la transition de la configuration d'exécution précédente à la nouvelle configuration peut se faire progressivement.

Mettre à jour la configuration d'une Amazon GameLift Servers flotte

Utilisez la Amazon GameLift Servers console ou la AWS CLI pour mettre à jour les paramètres de votre flotte, modifier les sites distants ou supprimer une flotte. Pour les flottes gérées, vous ne pouvez

pas modifier la version du serveur de jeu ou le type d'instance d'une flotte. Au lieu de cela, vous devez remplacer la flotte.

 Outil de mise à jour rapide (pour le développement uniquement)

Dans le cas EC2 des flottes gérées, pour déployer une mise à jour de build sur un serveur de jeu, vous devez télécharger chaque nouvelle version Amazon GameLift Servers et créer une nouvelle flotte pour celle-ci.

L'outil de mise à jour rapide des versions vous permet de contourner ces étapes pendant le développement, ce qui vous permet de gagner du temps et d'accélérer les itérations de développement. Avec cet outil, vous pouvez rapidement mettre à jour les fichiers de build de votre jeu sur tous les ordinateurs d'une flotte existante. L'outil propose plusieurs options ; vous pouvez remplacer une version complète d'un jeu ou modifier 6 fichiers spécifiques, et vous pouvez gérer le redémarrage des processus du serveur de jeu après les mises à jour. Vous pouvez également l'utiliser pour mettre à jour les calculs individuels d'une flotte. Pour obtenir l'outil Fast Build Update et en savoir plus sur son utilisation, consultez le référentiel Amazon GameLift Servers Toolkit pour [l'outil Fast Build Update](#) Tool sur Github.

Vous pouvez mettre à jour les attributs de flotte modifiables, les paramètres de port et les configurations d'exécution à l'aide de la Amazon GameLift Servers console ou de la AWS CLI. Pour modifier les limites de mise à l'échelle, voir [Augmentez automatiquement la capacité de votre flotte avec Amazon GameLift Servers](#).

GameLift Serveurs Amazon console

1. Dans le volet de navigation de la [Amazon GameLift Servers console](#), sélectionnez Fleets.
2. Choisissez la flotte que vous souhaitez mettre à jour. Un parc doit être en ACTIVE état pour que vous puissiez le modifier.
3. Sur la page détaillée de la flotte, dans l'une des sections suivantes, choisissez Modifier.
 - Réglages du parc
 - Modifiez les attributs de la flotte tels que Name et Description.
 - Ajoutez ou supprimez des groupes de mesures qu'Amazon CloudWatch utilise pour suivre les Amazon GameLift Servers statistiques agrégées de plusieurs flottes.
 - Mettez à jour les paramètres de limite de création de ressources.
 - Activez ou désactivez la protection des sessions de jeu.

- Configuration d'exécution : vous pouvez modifier l'un des paramètres suivants de vos configurations d'exécution et ajouter ou supprimer des configurations d'exécution.
 - Modifiez le chemin de lancement de votre serveur de jeu.
 - Ajoutez, supprimez ou modifiez les paramètres de lancement facultatifs.
 - Modifiez le nombre de processus simultanés exécutés par vos serveurs de jeu.
 - Activation des sessions de jeu : modifiez la façon dont vous souhaitez que les processus du serveur s'exécutent et hébergent les sessions de jeu en mettant à jour le nombre maximum d'activations simultanées de sessions de jeu et le nouveau délai d'activation.
 - EC2 paramètres de port — Mettez à jour les adresses IP et les plages de ports qui autorisent l'accès entrant à la flotte.
4. Choisissez Confirmer pour enregistrer les modifications.

AWS CLI

Utilisez les commandes AWS CLI suivantes pour mettre à jour un parc :

- [update-fleet-attributes](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)

Mettre à jour les emplacements du parc

Vous pouvez ajouter ou supprimer les sites distants d'une flotte à l'aide de la Amazon GameLift Servers console ou de la AWS CLI. Vous ne pouvez pas modifier la région d'origine d'une flotte.

GameLift Serveurs Amazon console

1. Dans le volet de navigation de la [Amazon GameLift Servers console](#), sélectionnez Fleets.
2. Choisissez la flotte que vous souhaitez mettre à jour. Un parc doit être en ACTIVE état pour que vous puissiez le modifier.
3. Sur la page détaillée de la flotte, cliquez sur l'onglet Emplacements pour afficher les positions de la flotte.

4. Pour ajouter de nouveaux sites distants, choisissez Ajouter et sélectionnez les emplacements vers lesquels vous souhaitez déployer les instances. Cette liste n'inclut pas les instances pour lesquelles le type d'instance du parc n'est pas disponible.
5. Une fois les nouveaux points de vente sélectionnés, choisissez Ajouter. Amazon GameLift Servers ajoute les nouveaux emplacements à la liste, avec le statut défini sur NEW. Amazon GameLift Servers commence ensuite à approvisionner une instance dans chaque emplacement ajouté et à la préparer pour héberger des sessions de jeu.
6. Pour supprimer des sites distants existants de la flotte, utilisez les cases à cocher pour sélectionner un ou plusieurs sites répertoriés.
7. Une ou plusieurs flottes étant sélectionnées, choisissez Supprimer. Les emplacements supprimés restent dans la liste, le statut étant défini sur DELETING. Amazon GameLift Servers commence ensuite le processus de cessation de l'activité dans l'emplacement supprimé. Si des instances actives hébergent des sessions de jeu, Amazon GameLift Servers utilise le processus de fermeture du serveur de jeu pour mettre fin aux sessions de jeu, fermer les serveurs de jeu et arrêter les instances en toute simplicité.

AWS CLI

Utilisez les commandes AWS CLI suivantes pour mettre à jour l'emplacement des flottes :

- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Supprimer une flotte

Vous pouvez supprimer une flotte lorsque vous n'en avez plus besoin. La suppression d'une flotte supprime définitivement toutes les données associées aux sessions de jeu et aux sessions des joueurs, ainsi que les données métriques collectées. Comme alternative, vous pouvez conserver la flotte, désactiver la scalabilité automatique et mettre à l'échelle manuellement la flotte à 0 instance.

Note

Si la flotte dispose d'une connexion d'appairage VPC, demandez d'abord l'autorisation en appelant [CreateVpcPeeringAuthorization](#). Amazon GameLift Servers supprime la connexion d'appairage VPC lors de la suppression du parc.

Vous pouvez utiliser la Amazon GameLift Servers console ou l'outil AWS CLI pour supprimer un parc.

GameLift Serveurs Amazon console

1. Dans le volet de navigation de la [Amazon GameLift Servers console](#), sélectionnez Fleets.
2. Choisissez la flotte que vous souhaitez supprimer. Vous ne pouvez supprimer que des flottes dans ACTIVE notre ERROR statut.
3. Sélectionnez Delete (Supprimer).
4. Dans la boîte de dialogue Supprimer le parc de véhicules, confirmez la suppression en saisissant **delete**.
5. Sélectionnez Delete (Supprimer).

AWS CLI

Utilisez la commande AWS CLI suivante pour supprimer une flotte :

- [delete-fleet](#)

Déboguer les incidents de flotte Amazon GameLift Servers

Cette rubrique fournit des conseils sur la manière de résoudre les problèmes liés à vos EC2 flottes Amazon GameLift Servers gérées.

Problèmes de création de flotte

Lorsque vous créez une EC2 flotte gérée, le Amazon GameLift Servers service lance un flux de travail qui crée la flotte, déploie les EC2 instances avec la version de votre serveur de jeu installée et lance les processus du serveur de jeu sur chaque instance. Pour une description détaillée, voir [Comment fonctionne Amazon GameLift Servers la création de flottes](#). Une flotte ne peut pas héberger de sessions de jeu ni de joueurs tant qu'elle n'a pas atteint le statut Actif.

Vous pouvez résoudre les problèmes qui empêchent les flottes de devenir actives en identifiant la phase de création de flotte au cours de laquelle le problème s'est produit et en consultant les événements et les journaux de création de flotte. Si les journaux ne fournissent pas d'informations utiles, il est possible que le problème soit dû à une erreur de service interne. Dans ce cas, essayez de créer à nouveau la flotte. Si le problème persiste, essayez de télécharger à nouveau le build du

jeu pour corriger une éventuelle corruption du fichier). Vous pouvez également contacter le support Amazon GameLift Servers ou poser une question sur le forum.

Téléchargement et validation du build

Au cours de cette phase, Amazon GameLift Servers récupère le build de votre serveur de jeu téléchargé, extrait les fichiers et exécute les scripts d'installation. Si la création d'une flotte échoue au cours de ces phases, examinez les événements de la flotte et les journaux pour identifier le problème. Les causes possibles incluent :

- Amazon GameLift ServersImpossible d'obtenir le fichier de construction compressé (événementFLEET_BINARY_DOWNLOAD_FAILED). Vérifiez que l'emplacement de stockage du build est accessible, que vous créez une flotte Région AWS identique au build et que Amazon GameLift Servers vous disposez des autorisations appropriées pour y accéder.
- Amazon GameLift ServersImpossible d'extraire les fichiers de compilation (événementFLEET_CREATION_EXTRACTING_BUILD).
- Un script d'installation contenu dans les fichiers de compilation n'a pas pu s'exécuter correctement (événementFLEET_CREATION_FAILED_INSTALLER).

Développement des ressources de flotte

Les problèmes rencontrés au cours de cette phase concernent généralement l'allocation et le déploiement des ressources de la flotte. Les causes possibles incluent :

- Le type d'instance demandé n'est pas disponible.
- Le type de flotte demandé (Spot ou On-Demand) n'est pas disponible.

Activation des processus du serveur de jeu

Au cours de cette phase, Amazon GameLift Servers tente un certain nombre de tâches et teste des éléments clés, notamment la viabilité du serveur de jeu, les paramètres de configuration d'exécution et la capacité du serveur de jeu à se connecter au Amazon GameLift Servers service à l'aide du SDK du serveur.

Note

Au cours de cette phase, vous pouvez accéder à distance à une instance de flotte pour étudier les problèmes de manière plus approfondie. Consultez [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).

Les problèmes possibles incluent :

- Les processus du serveur ne démarrent pas. Cela suggère un problème lié aux paramètres de configuration d'exécution de la flotte (événements `FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND` ou `FLEET_VALIDATION_EXECUTABLE_RUNTIME_FAILURE`. Vérifiez que vous avez correctement défini le chemin de lancement et les paramètres de lancement facultatifs.
- Les processus du serveur commencent à s'exécuter, mais le parc ne s'active pas. Si les processus du serveur démarrent et s'exécutent correctement, mais que le parc ne passe pas au statut Actif, cela est probablement dû au fait que le processus du serveur ne parvient pas à communiquer avec le Amazon GameLift Servers service. Vérifiez que votre serveur de jeu effectue les bons appels au SDK du serveur (voir [Initialiser le processus du serveur](#)) :
 - Le processus du serveur ne parvient pas à s'initialiser (événement `SERVER_PROCESS_SDK_INITIALIZATION_TIMEOUT`). Le processus du serveur n'est pas correctement appelé `InitSdk()`.
 - Le processus du serveur ne parvient pas à avertir Amazon GameLift Servers lorsqu'il est prêt à héberger une session de jeu (événement `SERVER_PROCESS_PROCESS_READY_TIMEOUT`). Le processus du serveur s'est initialisé mais n'a pas été appelé `ProcessReady()` à temps.
- Une demande de connexion d'appairage VPC a échoué. Pour les flottes créées avec une connexion d'appairage de VPC (voir [Pour configurer l'appairage de VPC avec une nouvelle flotte](#)), l'appairage de VPC est effectué au cours de cette phase d'activation. Si un appairage de VPC échoue pour une raison quelconque, la nouvelle flotte n'obtiendra pas le statut Actif. Vous pouvez suivre le succès ou l'échec de la demande de peering en appelant [describe-vpc-peering-connections](#). Assurez-vous de vérifier qu'il existe une autorisation d'appairage VPC valide ([describe-vpc-peering-authorizations](#), car les autorisations ne sont valides que pendant 24 heures).

Problèmes liés aux processus serveur

Les processus serveur démarrent mais échouent rapidement ou indiquent une intégrité médiocre.

Hormis les problèmes liés à votre version de génération de jeu, cela peut se produire lorsque vous essayez d'exécuter trop de processus serveur simultanément sur l'instance. Le nombre optimal de processus simultanés dépend des exigences relatives au type d'instance et aux ressources de votre serveur de jeux. Essayez de réduire le nombre de processus simultanés, lequel est défini dans la configuration d'exécution de la flotte, pour voir si les performances s'améliorent. Vous pouvez modifier la configuration d'exécution d'une flotte à l'aide de la Amazon GameLift Servers

console (modifier les paramètres d'allocation de capacité de la flotte) ou en appelant la AWS CLI commande [update-runtime-configuration](#).

Problèmes de suppression de flotte

Une flotte ne peut pas être résiliée en raison d'un nombre d'instances maximum.

Le message d'erreur indique que la flotte en cours de suppression dispose encore d'instances actives, ce qui n'est pas autorisé. Vous devez d'abord mettre à l'échelle une flotte en la réduisant à zéro instance active. Cela s'effectue en définissant manuellement le nombre d'instances souhaitées de la flotte à « 0 », puis en attendant que la réduction prenne effet. Veillez à désactiver la scalabilité automatique, qui contrecarrera les paramètres manuels.

Les actions de VPC ne sont pas autorisées.

Ce problème ne concerne que les flottes pour lesquelles vous avez spécifiquement créé des connexions d'appairage VPC (voir. [Peering VPC pour Amazon GameLift Servers](#)). Ce scénario se produit car le processus de suppression d'une flotte inclut également la suppression du VPC de la flotte et de toutes les connexions d'appairage VPC. Vous devez d'abord obtenir une autorisation en appelant l'API du service for Amazon GameLift Servers [CreateVpcPeeringAuthorization\(\)](#) ou en utilisant la commande AWS CLI `create-vpc-peering-authorization`. Une fois que vous avez l'autorisation, vous pouvez supprimer la flotte.

Amazon GameLift Servers Realtime problèmes liés à la flotte

Sessions de jeu zombie : elles commencent à exécuter un jeu, mais ne se terminent jamais.

Vous pouvez observer ces problèmes dans les scénarios suivants :

- Les mises à jour des scripts ne sont pas collectées par les serveurs Realtime de la flotte.
- La flotte atteint rapidement sa capacité maximale et n'est pas réduite lorsque l'activité des joueurs (telle que les nouvelles demandes de session de jeu) diminue.

Ceci découle de façon quasi-certaine d'un échec de l'appel à `processEnding` dans votre script Realtime. La flotte est activée et les sessions de jeu démarrées, mais il n'y a pas de méthode pour les arrêter. Par conséquent, le serveur Realtime qui exécute la session de jeu n'est jamais libéré pour en démarrer une nouvelle, et de nouvelles sessions de jeu ne peuvent démarrer que lorsque de nouveaux serveurs Realtime sont activés. En outre, les mises à jour du script Realtime n'affectent pas les sessions de jeu déjà en cours d'exécution, mais seulement les nouvelles.

Pour éviter que cela se produise, les scripts doivent fournir un mécanisme pour déclencher un appel `processEnding`. Comme illustré dans [Amazon GameLift Servers Realtime exemple de script](#), une solution consiste à programmer un délai d'expiration de session inactive, tel que si aucun joueur n'est connecté pendant un certain temps, le script met un terme à la session de jeu actuelle.

Toutefois, si vous suivez ce scénario, quelques solutions de contournement vous permettent de débloquent vos serveurs Realtime. L'astuce consiste à déclencher le redémarrage des processus du Realtime serveur, ou des instances de parc sous-jacentes. Dans ce cas, ferme Amazon GameLift Servers automatiquement les sessions de jeu pour vous. Une fois que les serveurs Realtime sont libérés, ils peuvent démarrer de nouvelles sessions de jeu à l'aide de la dernière version du script Realtime.

Il existe deux méthodes pour y parvenir, en fonction de l'ampleur du problème :

- Mettre à l'échelle la totalité de la flotte en la réduisant. Cette méthode est la plus simple à mettre en œuvre mais a un effet étendu. Mettez à l'échelle la flotte en la réduisant à zéro instance, attendez la réduction complète de la flotte, puis remettez-la à l'échelle en l'augmentant. Cela permet d'effacer toutes les sessions de jeu existantes et vous permet de repartir à zéro avec le script Realtime mis à jour le plus récemment.
- Accéder à distance à l'instance et redémarrer le processus. Il s'agit d'une bonne option si vous n'avez que quelques processus à corriger. Si vous êtes déjà connecté à l'instance, par exemple aux journaux de processus ou de débogage, cette méthode peut être la plus rapide. Consultez [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).

Si vous choisissez de ne pas inclure de manière d'appeler `processEnding` dans votre script Realtime, il existe quelques situations épineuses qui peuvent se produire, même lorsque la flotte devient active et que les sessions de jeu sont démarrées. Tout d'abord, une session de jeu en cours d'exécution ne se termine pas. Par conséquent, le processus serveur qui exécute cette session de jeu n'est jamais libre de démarrer une nouvelle session de jeu. Deuxièmement, le serveur Realtime ne collecte aucune mise à jour de script.

Connectez-vous à distance à Amazon GameLift Servers instances de flotte

Vous pouvez vous connecter à n'importe quelle instance dans votre Amazon GameLift Servers EC2 flottes gérées. Les raisons les plus courantes pour accéder à distance à une instance sont les suivantes :

- Résolvez les problèmes liés à l'intégration de votre serveur de jeu.

- Ajustez votre configuration d'exécution et les autres paramètres spécifiques à la flotte.
- Obtenez l'activité du serveur de jeu en temps réel, par exemple le suivi des journaux.
- Exécutez des outils d'analyse comparative en utilisant le trafic réel des joueurs.
- Étudiez les problèmes spécifiques liés à une session de jeu ou à un processus serveur.

Lorsque vous vous connectez à une instance, prenez en compte les problèmes potentiels suivants :

- Vous pouvez vous connecter à n'importe quelle instance d'un parc actif. En général, vous ne pouvez pas vous connecter à des flottes non actives, telles que les flottes en cours d'activation ou présentant un état d'erreur. (La disponibilité de ces flottes peut être limitée pendant une courte période.) Pour obtenir de l'aide concernant les problèmes d'activation de la flotte, consultez [Déboguer les incidents de flotte Amazon GameLift Servers](#).
- La connexion à une instance active n'affecte pas l'activité d'hébergement de l'instance. L'instance continue de démarrer et d'arrêter les processus du serveur en fonction de la configuration d'exécution. Il active et exécute des sessions de jeu. L'instance peut s'arrêter en réponse à un événement de réduction ou à un autre événement.
- Toute modification apportée aux fichiers ou aux paramètres de l'instance peut avoir un impact sur les sessions de jeu actives de l'instance et sur les joueurs connectés.

Les instructions suivantes décrivent comment se connecter à distance à une instance à l'aide de l'interface de ligne de commande AWS (CLI). Vous pouvez également effectuer des appels programmatiques à l'aide du AWS SDK, comme indiqué dans la référence de l'API [de service pour Amazon GameLift Servers](#).

Collectez les données d'instance

Pour vous connecter à une Amazon GameLift Servers instance EC2 de flotte gérée, vous avez besoin des informations suivantes :

- L'ID de l'instance à laquelle vous souhaitez vous connecter. Vous pouvez utiliser l'ID d'instance ou l'ARN.
- le SDK du serveur pour Amazon GameLift Servers version utilisée sur l'instance. Le SDK du serveur est intégré à la version du jeu exécutée sur l'instance.

Les instructions suivantes décrivent comment effectuer ces tâches à l'aide de la AWS CLI. Vous devez connaître l'ID de flotte de l'instance à laquelle vous souhaitez vous connecter.

1. Obtenez le nom du calcul. Obtenez la liste de tous les ordinateurs actifs de la flotte. [Calcul de la liste d'appels](#) avec un identifiant de flotte ou un ARN. Pour une flotte à site unique, spécifiez uniquement l'identifiant de flotte. Pour une flotte à sites multiples, spécifiez l'identifiant de la flotte et un emplacement. Dans le cas EC2 des flottes gérées, `list-compute` renvoie une liste d'instances de flotte, dont la propriété `ComputeName` est l'ID de l'instance. Trouvez le calcul auquel vous souhaitez accéder.

Demande

```
aws gamelift list-compute \
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
  --location "sa-east-1"
```

Réponse

```
{
  "ComputeList": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "FleetArn": "arn:aws:gamelift:us-west-2::fleet/fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeName": "i-0abc12d3e45fa6b78",
      "IpAddress": "00.00.000.00",
      "DnsName":
        "b08444ki909kvqu6zpw3is24x5pyz4b6m05i3jbxvpk9craztu0lqrbbrbrnbkks.uwp57060n1k6dn1nw49b78hg1west-2.amazongamelift.com",
      "ComputeStatus": "Active",
      "Location": "sa-east-1",
      "CreationTime": "2023-07-09T22:51:45.931000-07:00",
      "OperatingSystem": "AMAZON_LINUX_2023",
      "Type": "c4.large"
    }
  ]
}
```

2. Trouvez la version du SDK du serveur. Pour obtenir ces informations, vous devez rechercher le build déployé dans la flotte. La version du SDK du serveur est une propriété de construction.
 - a. Appelez [describe-fleet-attributes](#) avec un ID de flotte ou un ARN pour obtenir l'ID de build et l'ARN de la flotte.

- b. Appelez [describe-build](#) avec l'ID de build ou l'ARN pour obtenir la version du SDK du serveur de la version.

Par exemple :

Demande

```
aws gamelift describe-fleet-attributes /  
  --fleet-ids "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

Réponse

```
{  
  "FleetAttributes": [  
    {  
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
      "ComputeType": "EC2",  
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",  
      . . .  
    }  
  ]  
}
```

Demande

```
aws gamelift describe-build /  
  --build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```

Réponse

```
"Build": {  
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",  
  "Name": "My_Game_Server_Build_One",  
  "OperatingSystem": "AMAZON_LINUX_2023",  
  
  "ServerSdkVersion": "5.1.1",  
  . . .  
}
```

Se connecter à une instance (SDK 5 du serveur)

Si l'instance à laquelle vous souhaitez vous connecter exécute une version de jeu avec la version 5.x du SDK du serveur, connectez-vous à l'instance à l'aide d'Amazon EC2 Systems Manager (SSM). Vous pouvez accéder à des instances à distance qui exécutent Windows ou Linux.

Avant de commencer :

Complétez les étapes de configuration SSM et installez le plug-in SSM sur votre machine locale. Pour plus d'informations, consultez [Configuration de SSM](#) et [installation du plug-in Session Manager pour la AWS CLI dans le](#) guide de l'utilisateur d'Amazon EC2 Systems Manager.

1. Demandez des informations d'identification d'accès pour l'instance. Appelez [get-compute-access](#) avec l'ID de flotte et le nom de calcul de l'instance à laquelle vous souhaitez vous connecter. Amazon GameLift Servers renvoie un ensemble d'informations d'identification temporaires pour accéder à l'instance. Par exemple :

Demande

```
aws gamelift get-compute-access \  
--compute-name i-11111111a222b333c \  
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
--region us-west-2
```

Réponse

```
{  
  "ComputeName": " i-11111111a222b333c ",  
  "Credentials": {  
    "AccessKeyId": " ASIAIOSFODNN7EXAMPLE ",  
    "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY ",  
    "SessionToken": " AQoDYXdzEJr...<remainder of session token>"  
  },  
  "FleetArn": " arn:aws:gamelift:us-west-2::fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa ",  
  "FleetId": " fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa "  
}
```


2. Exportez les informations d'accès (facultatif). Vous pouvez exporter les informations d'identification vers des variables d'environnement et les utiliser pour configurer la AWS CLI pour l'utilisateur par défaut. Pour plus de détails, consultez la section [Variables d'environnement pour configurer la AWS CLI](#) dans le guide de AWS Command Line Interface l'utilisateur.

```
export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
```

3. Connectez-vous à l'instance de flotte. Démarrez une session SSM avec l'instance à laquelle vous souhaitez vous connecter. Incluez la AWS région ou l'emplacement de l'instance. Pour plus d'informations, notamment sur la façon de configurer SSM et le plug-in SSM, consultez [Starting a session \(AWS CLI\)](#) dans le guide de l'utilisateur d'Amazon EC2 Systems Manager.

La demande de démarrage de session utilisera automatiquement les informations d'identification que vous avez acquises à l'étape 1.

```
aws ssm start-session \
--target i-11111111a222b333c \
--region us-west-2 \
```

Note

Si vous recevez une erreur de refus d'accès, il se peut qu'une variable d'AWS_PROFILE environnement soit définie sur un AWS profil, ce qui amène la AWS CLI à utiliser des informations d'identification incorrectes pour l'accès à distance. Pour résoudre ce problème, désactivez temporairement votre variable d'AWS_PROFILE environnement. Vous pouvez également créer un AWS profil personnalisé pour vos informations d'accès à distance et ajouter le paramètre de ligne de `--profile` commande à votre `start-session` demande.

Se connecter à une instance (SDK de serveur 4.x ou version antérieure)

Si l'instance à laquelle vous souhaitez vous connecter exécute une version de jeu avec la version 4 ou antérieure du SDK du serveur, suivez les instructions suivantes. Vous pouvez vous connecter à des instances qui exécutent Windows ou Linux. Connectez-vous à une instance Windows à l'aide

d'un client RDP (Remote Desktop Protocol). Connectez-vous à une instance Linux à l'aide d'un client SSH.

1. Demandez des informations d'identification d'accès pour l'instance. Lorsque vous disposez d'un ID d'instance, utilisez la commande [get-instance-access](#) pour demander des informations d'accès. En cas de succès, Amazon GameLift Servers renvoie le système d'exploitation, l'adresse IP et un ensemble d'informations d'identification (nom d'utilisateur et clé secrète) de l'instance. Le format des informations d'identification dépend du système d'exploitation de l'instance. Utilisez les instructions suivantes extraire les informations d'identification pour RDP ou SSH.
- Pour les instances Windows : pour se connecter à une instance Windows, RDP a besoin d'un nom d'utilisateur et d'un mot de passe. La commande `get-instance-access` renvoie ces valeurs sous la forme de chaînes simples, afin que vous puissiez les valeurs telles quelles. Exemple d'informations d'identification :

```
"Credentials": {
  "Secret": "aA1bBB2cCCd3EEE",
  "UserName": "gl-user-remote"
}
```

- Pour les instances Linux : pour se connecter à une instance Linux, SSH a besoin d'un nom d'utilisateur et d'une clé privée. Amazon GameLift Servers émet des clés privées RSA et les renvoie sous forme de chaîne unique, le caractère de nouvelle ligne (\n) indiquant les sauts de ligne. Pour rendre la clé privée utilisable, procédez comme suit : (1) convertissez la chaîne en .pem fichier et (2) définissez les autorisations pour le nouveau fichier. Exemple d'informations d'identification renvoyées :

```
"Credentials": {
  "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEAY7WZhaDsR1W3mR1QtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHofnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdE0QW
\nZ/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
\nG50TCFe0zfl8dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW
\nnoPzev/D8V+x4+bHthfSjR9Y7DvQFjfBVwHXigBdtZcU2/wei8D/HYwIDAQABaoIBAGZ1kaEvnrqu
\n/uler7vgIn5m71N5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
ufGxbL1\nmb5qwmGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA30zdXzMqexXVJ1TLZVEH0E7bh1Y9d801ozR
```

```

\noQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfq1
+lIp1\nYkriL0DbLXlvRAH+yHPrit2hH0jtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh
+GTfbjsXs0xQx/x\np9otyVwc7hsQ5TA5PZb
+mvkJ50BEKzet9XcKw0NBVELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHfJRry21R1trw2Vdpn+9g481URipzWV0Eihvm+xTtmaZ1Sp//1kq75XDwnU
\nWA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
\nngYBjb0+OZk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\nnoMo7yykABY70zd5wQewBQ4AdSlWSX4nGDtsiFxiI5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVwFFPrTkJNvJZzYt69qezx1sjgFKshy
\nWBhd4xHZtmCqpBP1AymEjr/T0lbxyARMXMnIOWIANNXMGB4KGSyl1mzSVAoQ+fqr+cJ3d0dyP11j
\njjb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvT+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjKpUfVTTrW0YFjXkfcrR/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
RSA PRIVATE KEY-----",
    "UserName": "gl-user-remote"
}

```

Lorsque vous utilisez la AWS CLI, vous pouvez générer automatiquement un `.pem` fichier en incluant les paramètres `--query` et `--output` dans votre `get-instance-access` requête.

Pour définir des autorisations sur le fichier `.pem`, exécutez la commande suivante :

```
$ chmod 400 MyPrivateKey.pem
```

2. Ouvrez un port pour la connexion à distance. Vous pouvez accéder aux instances dans Amazon GameLift Servers flottes passant par n'importe quel port autorisé dans la configuration de la flotte. Vous pouvez afficher les paramètres de port de la flotte à l'aide de la commande [describe-fleet-port-settings](#).

La bonne pratique consiste à ouvrir des ports pour l'accès à distance uniquement lorsque vous en avez besoin et à les fermer lorsque vous avez terminé. Vous ne pouvez pas mettre à jour les paramètres du port après avoir créé une flotte, mais avant qu'elle ne soit active. Si vous êtes bloqué, recréez la flotte avec les paramètres du port ouverts.

Utilisez la commande [update-fleet-port-settings](#) pour ajouter un paramètre de port pour la connexion à distance (22 pour SSH ou 3389 pour RDP, par exemple). Pour la valeur de plage d'adresses IP, spécifiez les adresses IP pour les périphériques que vous voulez utiliser pour la connexion (converties au format CIDR). Exemple :

```
$ AWS gamelift update-fleet-port-settings
    --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

```
--inbound-permission-authorizations  
"FromPort=22,ToPort=22,IpRange=54.186.139.221/32,Protocol=TCP"
```

L'exemple suivant ouvre le port 3389 sur un parc Windows

```
$ AWS gamelift update-fleet-port-settings  
--fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"  
--inbound-permission-authorizations  
"FromPort=3389,ToPort=3389,IpRange=54.186.139.221/32,Protocol=TCP"
```

3. Ouvrez un client de connexion à distance. Utilisez le Bureau à distance pour Windows ou SSH pour les instances Linux. Connectez-vous à l'instance à l'aide de l'adresse IP, du paramètre de port et des informations d'identification d'accès.

Exemple SSH :

```
ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
```

Afficher des fichiers sur des instances distantes

Lorsque vous êtes connecté à une instance à distance, vous disposez d'un accès utilisateur et administratif complet. Cela signifie que vous avez également la possibilité de produire des erreurs et des défaillances dans l'hébergement de jeux. Si l'instance héberge des parties avec des joueurs actifs, vous courez le risque de bloquer des sessions de jeu et de perdre des joueurs, ou de perturber les processus d'arrêt du jeu et de provoquer des erreurs dans les données et les journaux de jeu enregistrés.

Recherchez les ressources suivantes sur une instance d'hébergement :

- Fichiers de build de jeu. Ces fichiers sont la version du jeu que vous avez téléchargée sur Amazon GameLift Servers. Ils incluent un ou plusieurs exécutables, actifs et dépendances du serveur de jeu. Les fichiers de compilation du jeu se trouvent dans un répertoire racine appelé game :
 - Sous Windows : c:\game
 - Sous Linux : /local/game
- Fichiers journaux de jeu. Trouvez les fichiers journaux générés par votre serveur de jeu dans le répertoire game racine, quel que soit le chemin de répertoire que vous avez indiqué.

- Amazon GameLift Servers ressources d'hébergement. Le répertoire racine `Whitewater` contient les fichiers utilisés par Amazon GameLift Servers service de gestion de l'activité d'hébergement de jeux. Ne modifiez pas ces fichiers pour quelque raison que ce soit.
- Configuration d'exécution. N'accédez pas à la configuration d'exécution pour des instances individuelles. Pour modifier une propriété de configuration d'exécution, mettez à jour la configuration d'exécution du parc (voir le fonctionnement du AWS SDK [UpdateRuntimeConfiguration](#) ou le AWS CLI [update-runtime-configuration](#)).
- Données relatives à la flotte. Un fichier JSON contient des informations sur le parc auquel appartient l'instance, destinées aux processus du serveur exécutés sur l'instance. Le fichier JSON se trouve à l'emplacement suivant :
 - Sous Windows : `C:\GameMetadata\gamelift-metadata.json`
 - Sous Linux : `/local/gamemetadata/gamelift-metadata.json`
- Certificat TLS Si l'instance fait partie d'un parc sur lequel la génération de certificats TLS est activée, recherchez les fichiers de certificat, y compris le certificat, la chaîne de certificats, la clé privée et le certificat racine à l'emplacement suivant :
 - Sous Windows : `c:\\GameMetadata\Certificates`
 - Sous Linux : `/local/gamemetadata/certificates/`

Peering VPC pour Amazon GameLift Servers

Cette rubrique fournit des conseils sur la façon de configurer une connexion d'appairage VPC entre votre Amazon GameLift Servers-serveurs de jeu hébergés et vos autres serveurs non hébergés-Amazon GameLift Servers ressources. Utilisez les connexions de peering Amazon Virtual Private Cloud (VPC) pour permettre à vos serveurs de jeu de communiquer directement et en privé avec vos autres AWS ressources, telles qu'un service Web ou un référentiel. Vous pouvez établir un peering VPC avec toutes les ressources exécutées AWS et gérées par un AWS compte auquel vous avez accès.

Note

L'appairage de VPC est une fonction avancée. Pour en savoir plus sur les options préférées permettant à vos serveurs de jeu de communiquer directement et en privé avec vos autres AWS ressources, consultez [Communiquez avec les autres AWS ressources de vos flottes](#).

Si vous connaissez déjà Amazon VPCs et le peering VPC, sachez que la configuration du peering avec Amazon GameLift Servers les serveurs de jeu sont quelque peu différents. Vous n'avez pas accès au VPC qui contient vos serveurs de jeu. Il est contrôlé par le Amazon GameLift Servers service : vous ne pouvez donc pas demander directement de peering VPC pour ce service. Au lieu de cela, vous devez d'abord pré-autoriser le VPC auprès de votre partenaire non professionnel. Amazon GameLift Servers ressources pour accepter une demande de peering émanant du Amazon GameLift Servers service. Ensuite, vous déclenchez Amazon GameLift Servers pour demander le peering VPC que vous venez d'autoriser. Amazon GameLift Servers gère les tâches de création de la connexion d'appairage, de configuration des tables de routage et de configuration de la connexion.

Pour configurer l'appairage de VPC pour une flotte existante

1. Obtenez les identifiants de AWS compte et les informations d'identification.

Vous avez besoin d'un identifiant et d'informations de connexion pour les AWS comptes suivants. Vous pouvez trouver un AWS compte en vous IDs connectant [AWS Management Console](#) et en consultant les paramètres de votre compte. Pour obtenir les informations d'identification, accédez à la console IAM.

- AWS compte que vous utilisez pour gérer votre Amazon GameLift Servers serveurs de jeux.
- AWS compte que vous utilisez pour gérer votre compte non-A Amazon GameLift Servers ressources.

Si vous utilisez le même compte pour Amazon GameLift Servers et non-A Amazon GameLift Servers ressources, vous avez besoin d'un identifiant et d'informations d'identification pour ce compte uniquement.

2. Obtenez les identifiants de chaque VPC.

Obtenez les informations suivantes pour que les deux VPCs soient comparés :

- VPC pour votre Amazon GameLift Servers serveurs de jeu — C'est votre Amazon GameLift Servers numéro de flotte. Vos serveurs de jeu sont déployés dans Amazon GameLift Servers sur un parc d' EC2 instances. Une flotte est automatiquement placée dans son propre VPC, qui est géré par le Amazon GameLift Servers service. Comme vous n'avez pas d'accès direct au VPC, celui-ci est identifié par l'identifiant de flotte.
- VPC pour votre entreprise non gouvernementale Amazon GameLift Servers AWS ressources — Vous pouvez établir un peering VPC avec toutes les ressources exécutées AWS et gérées par un AWS compte auquel vous avez accès. Si vous n'avez pas encore créé de VPC pour

ces ressources, consultez [Getting started with Amazon VPC](#). Une fois que vous avez créé un VPC, vous pouvez trouver son ID en vous connectant au VPC pour [AWS Management Console](#) Amazon et en consultant votre VPCs

 Note

Lors de la configuration d'un peering, les deux VPCs doivent exister dans la même région. Le VPC pour votre Amazon GameLift Servers les serveurs de jeu de la flotte se trouvent dans la même région que la flotte.

3. Autorisez un appairage de VPC.

Au cours de cette étape, vous pré-autorisez une future demande de Amazon GameLift Servers pour associer le VPC à vos serveurs de jeu à votre VPC pour les non-initiés, Amazon GameLift Servers ressources. Cette étape met à jour le groupe de sécurité pour votre VPC.

Pour autoriser le peering VPC, appelez l'API de service [CreateVpcPeeringAuthorization\(\)](#) ou utilisez la commande CLI AWS `create-vpc-peering-authorization` Passez cet appel en utilisant le compte qui gère votre organisation non gouvernementale Amazon GameLift Servers ressources. Identifiez les informations suivantes :

- ID VPC homologue : il s'agit du VPC dont vous n'êtes pas le propriétaire. Amazon GameLift Servers ressources.
- Amazon GameLift Servers AWS ID de compte — Il s'agit du compte que vous utilisez pour gérer votre Amazon GameLift Servers flotte.

Une fois que vous avez autorisé un peering VPC, l'autorisation reste valide pendant 24 heures, sauf si elle est révoquée. Vous pouvez gérer vos autorisations d'appairage de VPC à l'aide des opérations suivantes :

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).

4. Demandez une connexion d'appairage.

Avec une autorisation valide, vous pouvez demander que Amazon GameLift Servers établir une connexion d'appairage.

Pour demander un peering VPC, appelez l'API de service [CreateVpcPeeringConnection\(\)](#) ou utilisez la commande CLI AWS `create-vpc-peering-connection`. Passez cet appel en utilisant le compte qui gère votre Amazon GameLift Servers serveurs de jeux. Utilisez les informations suivantes pour identifier les deux VPCs que vous souhaitez comparer :

- ID VPC homologue et ID de AWS compte : il s'agit du VPC pour votre entreprise non établie. Amazon GameLift Servers les ressources et le compte que vous utilisez pour les gérer. L'ID du VPC doit correspondre à celui d'une autorisation d'appairage valide.
- Numéro de flotte : il identifie le VPC de votre Amazon GameLift Servers serveurs de jeux.

5. Suivez l'état de la connexion de l'appairage.

La demande d'une connexion d'appairage de VPC est une opération asynchrone. Pour suivre l'état d'une demande d'appairage et gérer les cas de réussite ou d'échec, utilisez l'une des options suivantes :

- Interrogez de façon continue avec `DescribeVpcPeeringConnections()`. Cette opération récupère l'enregistrement de connexion de l'appairage de VPC, y compris l'état de la demande. Si une connexion d'appairage est créée avec succès, l'enregistrement de la connexion contient également le bloc CIDR d'adresses IP privées attribué au VPC.
- Gérez les événements de flotte associés aux connexions de peering VPC avec [DescribeFleetEvents\(\)](#), y compris les événements de réussite et d'échec.

Une fois la connexion d'appairage établie, vous pouvez la gérer à l'aide des opérations suivantes :

- [DescribeVpcPeeringConnections\(\)](#) (AWS CLI `describe-vpc-peering-connections`).
- [DeleteVpcPeeringConnection\(\)](#) (AWS CLI `delete-vpc-peering-connection`).

Pour configurer l'appairage de VPC avec une nouvelle flotte

Vous pouvez créer un nouveau Amazon GameLift Servers flotte et demandez une connexion d'appairage VPC en même temps.

1. Obtenez les identifiants de AWS compte et les informations d'identification.

Vous avez besoin d'un identifiant et d'informations de connexion pour les deux AWS comptes suivants. Vous pouvez trouver un AWS compte en vous IDs connectant [AWS Management](#)

[Console](#) et en consultant les paramètres de votre compte. Pour obtenir les informations d'identification, accédez à la console IAM.

- AWS compte que vous utilisez pour gérer votre Amazon GameLift Servers serveurs de jeux.
- AWS compte que vous utilisez pour gérer votre compte non-A Amazon GameLift Servers ressources.

Si vous utilisez le même compte pour Amazon GameLift Servers et non-A Amazon GameLift Servers ressources, vous avez besoin d'un identifiant et d'informations d'identification pour ce compte uniquement.

2. Obtenez l'ID VPC de votre entreprise non gouvernementale Amazon GameLift Servers AWS ressources.

Si vous n'avez pas encore créé de VPC pour ces ressources, faites-le maintenant (voir [Getting started with Amazon VPC](#)). Veillez à créer le nouveau VPC dans la région où vous prévoyez de créer votre nouvelle flotte. Si vous n'êtes pas... Amazon GameLift Servers les ressources sont gérées sous un AWS compte ou un groupe d'utilisateurs/utilisateurs différent de celui que vous utilisez avec Amazon GameLift Servers, vous devrez utiliser ces informations d'identification de compte lors de la demande d'autorisation à l'étape suivante.

Une fois que vous avez créé un VPC, vous pouvez trouver son ID dans la console Amazon VPC en consultant votre. VPCs

3. Autoriser un peering VPC avec des non-initiateurs Amazon GameLift Servers ressources.

Lorsque Amazon GameLift Servers crée la nouvelle flotte et un VPC correspondant, il envoie également une demande d'appariement avec le VPC pour votre non-propriétaire Amazon GameLift Servers ressources. Vous devez pré-autoriser cette demande. Cette étape met à jour le groupe de sécurité pour votre VPC.

En utilisant les informations d'identification du compte qui gèrent votre compte non... Amazon GameLift Servers ressources, appelez l'API du service [CreateVpcPeeringAuthorization\(\)](#) ou utilisez la commande AWS CLI `create-vpc-peering-authorization`. Identifiez les informations suivantes :

- ID VPC homologue : ID du VPC avec votre non-partenaire Amazon GameLift Servers ressources.

- Amazon GameLift Servers AWS ID de compte — ID du compte que vous utilisez pour gérer votre Amazon GameLift Servers flotte.

Une fois que vous avez autorisé un peering VPC, l'autorisation reste valide pendant 24 heures, sauf si elle est révoquée. Vous pouvez gérer vos autorisations d'appairage de VPC à l'aide des opérations suivantes :

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
 - [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).
4. Suivez les instructions pour [créer une nouvelle flotte à l'aide de la AWS CLI](#). Incluez les paramètres requis supplémentaires suivants :
- `peer-vpc-aws-account-id` — ID du compte que vous utilisez pour gérer le VPC avec votre non-utilisateur Amazon GameLift Servers ressources.
 - `peer-vpc-id` — Identifiant du VPC auprès de votre non-membre Amazon GameLift Servers .

Un appel réussi à [create-fleet](#) avec les paramètres d'appairage de VPC génère à la fois une nouvelle flotte et une nouvelle demande d'appairage de VPC. Le statut de la flotte est défini sur `New` et le processus d'activation de la flotte est initié. Le statut de la demande d'appairage est défini sur `initiating-request`. Vous pouvez suivre le succès ou l'échec de la demande de peering en appelant [describe-vpc-peering-connections](#).

Lorsque vous demandez une connexion d'appairage de VPC avec une nouvelle flotte, les deux actions aboutissent ou échouent. Si une flotte échoue pendant le processus de création, la connexion d'appairage de VPC n'est pas établie. De même, si une connexion d'appairage de VPC échoue pour une raison quelconque, la nouvelle flotte ne peut pas passer du statut `Activating` au statut `Active`.

 Note

La nouvelle connexion d'appairage de VPC n'est pas terminée tant que la flotte n'est pas prête à être active. Cela signifie que la connexion n'est pas disponible et ne peut pas être utilisée pendant le processus d'installation du serveur de jeu.

L'exemple suivant crée à la fois une nouvelle flotte et une connexion d'appairage entre un VPC préétabli et le VPC de la nouvelle flotte. Le VPC préétabli est identifié de manière unique par la combinaison de vos partenaires non établis Amazon GameLift Servers AWS ID de compte et ID VPC.

```
$ AWS gamelift create-fleet
  --name "My_Fleet_1"
  --description "The sample test fleet"
  --ec2-instance-type "c5.large"
  --fleet-type "ON_DEMAND"
  --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
  --runtime-configuration "GameSessionActivationTimeoutSeconds=300,
                           MaxConcurrentGameSessionActivations=2,
                           ServerProcesses=[{LaunchPath=C:\game\Bin64.dedicated
\MultiplayerSampleProjectLauncher_Server.exe,
                                           Parameters=+sv_port 33435 +start_lobby,
                                           ConcurrentExecutions=10}]]"
  --new-game-session-protection-policy "FullProtection"
  --resource-creation-limit-policy "NewGameSessionsPerCreator=3,
                                   PolicyPeriodInMinutes=15"
  --ec2-inbound-permissions
  "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"

  "FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP"
  --metric-groups "EMEAfleets"
  --peer-vpc-aws-account-id "111122223333"
  --peer-vpc-id "vpc-a11a11a"
```

Version copiable :

```
AWS gamelift create-fleet --name "My_Fleet_1" --description "The
sample test fleet" --fleet-type "ON_DEMAND" --metric-groups
"EMEAfleets" --build-id "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff"
--ec2-instance-type "c5.large" --runtime-configuration
"GameSessionActivationTimeoutSeconds=300,MaxConcurrentGameSessionActivations=2,ServerProcesses
\game\Bin64.dedicated\MultiplayerSampleProjectLauncher_Server.exe,Parameters=
+sv_port 33435 +start_lobby,ConcurrentExecutions=10}]]" --new-game-session-
protection-policy "FullProtection" --resource-creation-limit-policy
"NewGameSessionsPerCreator=3,PolicyPeriodInMinutes=15" --ec2-inbound-
permissions "FromPort=33435,ToPort=33435,IpRange=0.0.0.0/0,Protocol=UDP"
"FromPort=33235,ToPort=33235,IpRange=0.0.0.0/0,Protocol=UDP" --peer-vpc-aws-account-id
"111122223333" --peer-vpc-id "vpc-a11a11a"
```

Résolution des problèmes d'appairage de VPC

Si vous ne parvenez pas à établir une connexion d'appairage VPC pour votre Amazon GameLift Servers serveurs de jeux, considérez ces causes fondamentales courantes :

- Une autorisation pour la connexion demandée est introuvable.
 - Vérifiez le statut d'une autorisation VPC pour les non-utilisateurs Amazon GameLift Servers VPC. Il se peut qu'il n'existe pas ou qu'il ait expiré.
 - Vérifiez les régions des deux VPCs que vous essayez de regarder. S'ils ne sont pas dans la même région, ils ne peuvent pas être appairés.
- Les blocs CIDR (voir Configurations de [connexion d'appairage VPC non valides](#)) de vos VPCs deux blocs se chevauchent. Les blocs IPv4 CIDR affectés à peered VPCs ne peuvent pas se chevaucher. Le bloc CIDR du VPC pour votre Amazon GameLift Servers la flotte est automatiquement attribuée et ne peut pas être modifiée. Vous devrez donc modifier le bloc CIDR du VPC pour votre non-proprétaire Amazon GameLift Servers ressources. Pour résoudre ce problème :
 - Recherchez ce bloc CIDR pour votre Amazon GameLift Servers flotte en appelant `DescribeVpcPeeringConnections()`.
 - Accédez à la console Amazon VPC, trouvez le VPC pour votre entreprise non... Amazon GameLift Servers ressources, et modifiez le bloc CIDR afin qu'elles ne se chevauchent pas.
- La nouvelle flotte ne s'est pas activée (lors de la demande d'appairage VPC avec une nouvelle flotte). Si la nouvelle flotte n'a pas réussi à passer au statut Actif, il n'y a pas de VPC à appairer, de sorte que la connexion d'appairage ne peut pas réussir.

Amazon GameLift Servers flottes de conteneurs gérées

Amazon GameLift Servers les flottes de conteneurs gérées fournissent des ressources basées sur le cloud pour héberger votre logiciel de serveur de jeu conteneurisé. Avec un parc de conteneurs géré, vous bénéficiez de la flexibilité, de la sécurité et de la fiabilité des AWS Cloud ressources, optimisées pour l'hébergement de jeux multijoueurs. Amazon GameLift Servers fournit des outils de gestion d'hôte robustes.

 Accélérez l'intégration grâce à ces outils pour les conteneurs gérés :

- Le [kit de démarrage pour conteneurs](#) rationalise l'intégration et la configuration de la flotte. Il ajoute des fonctionnalités essentielles de gestion des sessions de jeu à votre serveur de

jeu et utilise des modèles préconfigurés pour créer une flotte de conteneurs et un pipeline de déploiement automatisé pour votre serveur de jeu. Après le déploiement, utilisez la Amazon GameLift Servers console et les outils d'API pour surveiller les performances de la flotte, gérer les sessions de jeu et analyser les indicateurs.

- Pour les développeurs Unreal Engine ou Unity, utilisez les [Amazon GameLift Servers plugins et le serveur du](#) moteur de jeu SDKs pour intégrer votre serveur de jeu et créer une flotte de conteneurs à partir de l'environnement de développement de votre moteur de jeu. Les flux de travail guidés du plugin vous aident à créer une solution simple et rapide avec un hébergement basé sur le cloud à l'aide de conteneurs gérés. Vous pouvez vous appuyer sur cette base pour créer une solution d'hébergement personnalisée pour votre jeu.

Un parc de conteneurs géré est un ensemble d'instances Amazon Elastic Compute Cloud (Amazon EC2) exécutant Linux, qui Amazon GameLift Servers possède et exploite en fonction de votre configuration. Ces instances sont physiquement situées dans des zones prises en charge Régions AWS ou des zones locales. Lorsque vous créez un parc de conteneurs, vous choisissez un type d'EC2 instance qui répond aux exigences de votre serveur de jeu en termes de puissance de calcul, de mémoire, de stockage et de capacités réseau.

Pour un parc de conteneurs géré, vous stockez des images de conteneurs basées sur Linux dans un référentiel Amazon Elastic Container Registry (Amazon ECR) et vous créez une définition de groupe de conteneurs pour décrire votre architecture de conteneurs. Lorsque vous créez une flotte, Amazon GameLift Servers approvisionnez une instance de flotte avec la dernière version de Linux Amazon Machine Image (AMI) et utilisez la définition du groupe de conteneurs pour déployer vos images de conteneurs. Toutes les instances d'un parc de conteneurs utiliseront la même version d'AMI, même si vous mettez à jour la définition d'un groupe de conteneurs ou modifiez une image de conteneur.

Note

À titre de bonne pratique, nous vous recommandons de remplacer vos flottes tous les 30 jours afin de garantir un environnement d' up-to-date exécution sécurisé pour vos serveurs de jeux hébergés. Cela nécessite la création d'une nouvelle flotte et la migration du trafic des joueurs vers celle-ci. Pour plus d'informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

Après le déploiement de l'instance conteneurisée, les conteneurs commencent à lancer les processus du serveur de jeu. Chaque processus du serveur de jeu établit une connexion au Amazon GameLift Servers service, indique qu'il est prêt à héberger une session de jeu et commence à communiquer l'état de santé. Amazon GameLift Servers peut ensuite demander au processus du serveur de démarrer une session de jeu.

Outre le déploiement de la flotte, Amazon GameLift Servers gère les tâches de gestion des hôtes suivantes afin que vous n'ayez pas à :

- Suit l'état de tous les conteneurs de la flotte et remplace les conteneurs périmés ou insalubres.
- Gère l'authentification pour la communication entre les processus du serveur et le Amazon GameLift Servers service.
- Propose des outils d'auto-scaling qui ajustent la capacité de la flotte de manière dynamique pour répondre à la demande des joueurs.
- Indique les indicateurs de performance pour les EC2 instances, les conteneurs et les processus de serveur de la flotte.

Consultez les rubriques suivantes pour savoir comment configurer et gérer des flottes de conteneurs gérées :

- [Feuille de route de développement pour l'hébergement avec des conteneurs Amazon GameLift Servers gérés](#)
- [Création d'une flotte de conteneurs Amazon GameLift Servers gérée](#)
- [Personnalisez une flotte de Amazon GameLift Servers conteneurs](#)
- [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#)
- [Mettre à jour et Amazon GameLift Servers gérer un parc de conteneurs](#)

Création d'une flotte de conteneurs Amazon GameLift Servers gérée

Créez un parc de conteneurs Amazon GameLift Servers géré pour déployer et héberger votre serveur de jeu conteneurisé dans le AWS cloud. Lorsque vous créez une flotte de conteneurs, spécifiez des définitions de groupes de conteneurs qui spécifient une ou plusieurs images de conteneurs (au moins une qui inclut la version de votre serveur de jeu) et des paramètres de configuration.

Lorsque vous créez une nouvelle ressource de flotte de conteneurs gérée, vous lancez immédiatement la première phase de création de flotte. La création d'une flotte gérée passe par plusieurs phases : Amazon GameLift Servers provisionnement d'une EC2 instance, installation d'un environnement d'exécution, déploiement de vos groupes de conteneurs sur l'instance et lancement des processus du serveur de jeu. En fonction de l'environnement d'exécution requis par la création de votre serveur de jeu, Amazon GameLift Servers déploie la dernière version d'Amazon Machine Image (AMI) au moment de la création de la flotte (et toutes les futures instances de la flotte utiliseront la même version). Vous pouvez surveiller l'état d'une flotte dans la console ou à l'aide du AWS Command Line Interface (AWS CLI). Lorsqu'une flotte est prête à accueillir des sessions de jeu, son statut passe à **ACTIVE**. Pour obtenir de l'aide face à des problèmes de création de flotte, veuillez consulter [Déboguer les incidents de flotte Amazon GameLift Servers](#).

Vous pouvez choisir de créer une flotte de conteneurs vide, puis d'ajouter ou de mettre à jour les définitions des groupes de conteneurs de la flotte ultérieurement. Si vous créez une flotte sans définition de groupe de conteneurs, la flotte n'atteindra pas le statut actif.

Note

À titre de bonne pratique, nous vous recommandons de remplacer vos flottes tous les 30 jours afin de garantir un environnement d' up-to-date exécution sécurisé pour vos serveurs de jeux hébergés. Cela nécessite la création d'une nouvelle flotte et la migration du trafic des joueurs vers celle-ci. Pour plus d'informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

Utilisez la [Amazon GameLift Servers console](#) ou le AWS Command Line Interface (AWS CLI) pour créer une flotte de conteneurs.

Console

Dans la [Amazon GameLift Servers console](#), sélectionnez l' Région AWS endroit où vous souhaitez créer la flotte. Les définitions des groupes de conteneurs doivent se trouver dans la même région que celle où vous souhaitez créer la flotte.

Ouvrez la barre de navigation gauche de la console et choisissez Conteneurs gérés : flottes. Sur la page Flottes, choisissez Create container fleet.

Étape 1 : définir les détails du parc de conteneurs géré

1. Dans la section Détails de la flotte de conteneurs, entrez une description de la flotte.
2. Spécifiez un rôle IAM pour la flotte. Ce rôle dispose des Amazon GameLift Servers autorisations nécessaires pour gérer le parc de conteneurs en votre nom. Pour obtenir de l'aide pour créer le rôle de service requis, consultez [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).
3. Choisissez une option de configuration du journal. L' CloudWatch option est sélectionnée par défaut. Fournissez les informations requises en fonction de l'option que vous avez sélectionnée.
4. Ajoutez des groupes de conteneurs à la flotte. Il s'agit d'une étape facultative. Vous pouvez choisir de créer une flotte sans groupe de conteneurs avec un plan pour les ajouter ultérieurement. Une flotte dépourvue de groupes de conteneurs ne déploiera aucune instance de flotte et ne pourra pas encore héberger de jeux, mais la ressource de flotte est créée.
 - Sélectionnez une définition de groupe de conteneurs de serveurs de jeu. Spécifiez éventuellement la version de la définition que vous souhaitez déployer. Si vous ne spécifiez pas le numéro de version, utilise Amazon GameLift Servers automatiquement la dernière version.
 - Ajoutez éventuellement une définition et une version de groupe de conteneurs par instance. Si vous ne spécifiez pas le numéro de version, utilise Amazon GameLift Servers automatiquement la dernière version.
5. Dans les détails supplémentaires, vous pouvez définir certaines personnalisations facultatives. Aucun de ces paramètres n'est requis pour créer le parc de conteneurs.

Étape 2 : définir les détails de l'instance

1. Dans Déploiement d'instances, sélectionnez un ou plusieurs sites distants sur lesquels déployer les instances. La région d'origine est automatiquement sélectionnée (il s'agit de la région dans laquelle vous créez la flotte). Si vous sélectionnez des sites supplémentaires, des instances de flotte sont également déployées sur ces sites.

Important

Pour utiliser des régions qui ne sont pas activées par défaut, activez-les dans votre Compte AWS.

- Les flottes dont les régions ne sont pas activées et que vous avez créées avant le 28 février 2022 ne sont pas affectées.
- Pour créer de nouvelles flottes multi-sites ou pour mettre à jour les flottes multi-sites existantes, activez d'abord les régions que vous choisissez d'utiliser.

Pour plus d'informations sur les régions qui ne sont pas activées par défaut et sur la manière de les activer, consultez la section [Gestion Régions AWS](#) dans le Références générales AWS.

2. Sélectionnez une configuration d'instance pour le parc. La console calcule automatiquement le minimum de vCPU et de mémoire requis (en fonction des limites totales que vous définissez pour chaque groupe de conteneurs). Il filtre la liste complète des types d'instances disponibles en fonction des besoins en ressources et des emplacements que vous avez saisis. Vous pouvez ajouter des filtres supplémentaires selon vos besoins.

Pour plus d'informations sur le choix d'un type d'instance, consultez [Configuration d'une flotte de conteneurs](#). La taille du type d'instance que vous choisissez aura un impact sur la manière dont les groupes de conteneurs du serveur de jeu sont regroupés dans chaque instance de flotte. En fonction de votre choix, pensez à revoir vos paramètres pour les groupes de conteneurs de serveurs de jeu souhaités par instance.

Étape 4 : vérifier et créer

- Passez en revue les paramètres de configuration de votre flotte.

Vous pouvez mettre à jour les métadonnées et la configuration de la flotte à tout moment, quel que soit l'état de la flotte. Pour de plus amples informations, veuillez consulter [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#). Vous pouvez mettre à jour la capacité de la flotte une fois que celle-ci a atteint le statut ACTIF. Pour de plus amples informations, veuillez consulter [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#). Vous pouvez également ajouter ou supprimer des sites distants.

Lorsque vous avez terminé de réviser, choisissez Créer.

Si votre demande est acceptée, la console affiche la page détaillée de la nouvelle ressource de flotte. Au départ NEW, le statut est le suivant : au Amazon GameLift Servers moment où commence le processus de création de flotte. Vous pouvez suivre le statut de la nouvelle

flotte sur la page Flottes. Une flotte est prête à accueillir des sessions de jeu lorsqu'elle atteint son statut `ACTIVE`.

AWS CLI

Pour créer une flotte de conteneurs avec le AWS CLI, ouvrez une fenêtre de ligne de commande et utilisez la `create-container-fleet` commande. Pour plus d'informations sur cette commande, consultez [create-container-fleet](#) la référence des AWS CLI commandes.

L'exemple de `create-container-fleet` demande ci-dessous crée une nouvelle flotte de conteneurs présentant les caractéristiques suivantes :

- `ContainerGroupsConfiguration` Spécifie une définition de groupe de conteneurs de serveurs de jeux uniquement : `MyAdventureGameContainerGroup`. Le nombre de groupes de conteneurs de serveurs de jeu qui seront déployés sur chaque instance de flotte est calculé par Amazon GameLift Servers.
- Le parc utilise par défaut des instances à la demande `c5.large`.
- Par défaut, le parc ouvre un ensemble de ports de connexion et de ports d'autorisations entrantes tels que calculés par Amazon GameLift Servers. Il déploie des groupes de conteneurs aux emplacements suivants :

```
aws gamelift create-container-fleet \  
  --fleet-role-arn arn:aws:iam::MyAccount:role/MyContainersRole \  
  --game-server-container-group-definition-name "rn:aws:gamelift:us-  
west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2" \  
  --game-server-container-group-definition-version 1
```

Si la demande de création de flotte aboutit, Amazon GameLift Servers renvoie un ensemble d'attributs de flotte comprenant les paramètres de configuration que vous avez demandés et un nouvel identifiant de flotte de conteneurs. Amazon GameLift Servers définit ensuite le statut de la flotte et les statuts de localisation sur Nouveau et lance le processus d'activation de la flotte. Vous pouvez suivre le statut de la flotte et consulter d'autres d'informations de flotte à l'aide de ces commandes d'interface de ligne de commande :

- [describe-fleet-events](#)
- [describe-container-fleet](#)
- [describe-fleet-capacity](#)

- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

Vous pouvez changer la capacité de la flotte et d'autres paramètres de configuration, si nécessaire, à l'aide des commandes suivantes :

- [update-container-fleet](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Mettre à jour et Amazon GameLift Servers gérer un parc de conteneurs

Vous pouvez mettre à jour la plupart des propriétés d'un parc de conteneurs géré, y compris les définitions des groupes de conteneurs. Selon les paramètres mis à jour, une mise à jour de flotte peut déclencher un nouveau déploiement de flotte. Lors d'un déploiement de flotte, toutes les instances de la flotte sont supprimées et remplacées par des instances dotées de la nouvelle configuration. Les paramètres qui nécessitent un déploiement sont les suivants :

- Définitions des groupes de conteneurs, y compris les mises à jour des images de conteneurs
- Plages de ports de connexion et autorisations entrantes
- Configuration du journal

Vous pouvez suivre l'état des déploiements de flottes dans la [Amazon GameLift Serversconsole](#) ou dans le AWS Command Line Interface (AWS CLI) pour créer une flotte de conteneurs.

Note

Vous ne pouvez pas mettre à jour l'environnement d'exécution d'une flotte. Lors de la création de la flotte, l'Amazon Machine Image (AMI) de la flotte est configurée sur la dernière

version disponible de l'AMI Linux. Toutes les images de conteneur déployées sur cette flotte doivent être compatibles avec cette version. Pour modifier l'AMI de la flotte ou passer à une version plus récente, vous devez créer une nouvelle flotte. À titre de bonne pratique, nous vous recommandons de remplacer vos flottes tous les 30 jours afin de garantir un environnement d' up-to-dateexécution sécurisé pour vos serveurs de jeux hébergés. Pour plus d'informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

Console

Dans la [Amazon GameLift Serversconsole](#), sélectionnez l' Région AWS endroit où vous souhaitez créer la flotte. Les définitions des groupes de conteneurs doivent se trouver dans la même région que celle où vous souhaitez créer la flotte.

Ouvrez la barre de navigation gauche de la console et choisissez Conteneurs gérés : flottes. Sur la page des flottes de conteneurs gérées, sélectionnez une flotte dans la liste et choisissez Modifier.

1. Mettez à jour les paramètres du parc de conteneurs selon les besoins. Lorsque vous avez terminé, choisissez Créer.
2. Si vos mises à jour nécessitent un déploiement de flotte, il vous est demandé de spécifier les options de déploiement comme suit :
 - Protection des sessions de jeu. Vous pouvez choisir de protéger les instances de flotte qui ont des sessions de jeu actives (déploiement sécurisé). Avec ce paramètre, les instances de flotte ne sont remplacées qu'après la fin des sessions de jeu. Vous pouvez également choisir de remplacer les instances de flotte indépendamment de l'activité de la session de jeu (déploiement non sécurisé). Les déploiements non sécurisés sont utiles pendant les phases de développement et de test afin de réduire le temps de déploiement.
 - Pourcentage de santé minimum. Vous pouvez gérer la rapidité avec laquelle les instances du parc sont remplacées. Utilisez ce paramètre pour maintenir un nombre minimal de tâches saines pendant le déploiement. Une valeur faible donne la priorité à la vitesse de déploiement, tandis qu'une valeur élevée garantit que la disponibilité du serveur de jeu reste élevée tout au long du déploiement.
 - Stratégie en cas d'échec du déploiement. Décidez des mesures à prendre en cas d'échec d'un déploiement. Un échec de déploiement signifie que certains conteneurs mis à jour ont échoué aux vérifications d'état et sont considérés comme défectueux. Vous pouvez configurer des déploiements pour rétablir automatiquement l'état précédemment déployé

de toutes les instances de flotte. Vous pouvez également choisir de conserver certaines des instances de flotte altérées pour les utiliser dans le cadre du débogage.

Si votre demande est acceptée, la console affiche l'onglet Déploiements pour le parc de conteneurs géré. Utilisez cet onglet pour suivre l'état de chaque déploiement. Si vous lancez un nouveau déploiement pour la flotte, cette action annule automatiquement tout déploiement en cours pour la flotte.

AWS CLI

Pour créer une flotte de conteneurs avec le AWS CLI, ouvrez une fenêtre de ligne de commande et utilisez la `update-container-fleet` commande. Pour plus d'informations sur cette commande, reportez-vous [update-container-fleet](#) à la référence des AWS CLI commandes.

L'exemple suivant met à jour un parc de conteneurs existant avec les caractéristiques suivantes :

- Il met à jour la définition du groupe de conteneurs du serveur de jeu pour utiliser la version 2.
- Elle spécifie les options de déploiement sécurisé.

```
{
  "DeploymentConfiguration": {
    "ImpairmentStrategy": "ROLLBACK",
    "MinimumHealthyPercentage": 75,
    "ProtectionStrategy": "WITH_PROTECTION"
  },
  "FleetId": "containerfleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "GameServerContainerGroupDefinitionName": "arn:aws:gamelift:us-
west-2:111122223333:containergroupdefinition/MyAdventureGameContainerGroup:2"
}
```

Personnalisez une flotte de Amazon GameLift Servers conteneurs

Les rubriques de cette section décrivent certaines des fonctionnalités facultatives des conteneurs Amazon GameLift Servers gérés. Vous pouvez choisir d'utiliser l'une ou l'ensemble de ces fonctionnalités.

Rubriques

- [Définissez des limites de ressources](#)

- [Désignez les contenants essentiels](#)
- [Configuration des connexions réseau](#)
- [Configurez des contrôles de santé pour les conteneurs](#)
- [Définir les dépendances des conteneurs](#)
- [Configuration d'une flotte de conteneurs](#)

Définissez des limites de ressources

Pour chaque groupe de conteneurs, vous pouvez déterminer la quantité de mémoire et de puissance de calcul dont le groupe de conteneurs a besoin pour exécuter son logiciel. Amazon GameLift Servers s'appuie sur ces informations pour gérer les ressources au sein du groupe de conteneurs. Il utilise également ces informations pour calculer le nombre de groupes de conteneurs de serveurs de jeu qu'une image de flotte peut contenir. Vous pouvez également définir des limites pour des conteneurs individuels.

Vous pouvez définir une limite maximale de mémoire et de puissance de calcul pour un groupe de conteneurs. Par défaut, ces ressources sont partagées par tous les conteneurs du groupe. Vous pouvez personnaliser davantage la gestion des ressources en définissant des limites pour les conteneurs individuels.

Définissez des limites facultatives pour des conteneurs individuels

La définition de limites de ressources spécifiques aux conteneurs vous permet d'exercer un meilleur contrôle sur la manière dont les conteneurs individuels peuvent utiliser les ressources du groupe. Si vous ne définissez pas de limites spécifiques aux conteneurs, tous les conteneurs du groupe partagent les ressources du groupe. Le partage offre une plus grande flexibilité pour utiliser les ressources là où elles sont nécessaires. Cela augmente également le risque de concurrence entre les processus et d'entraîner une défaillance des conteneurs.

Définissez l'une des `ContainerDefinition` propriétés suivantes pour n'importe quel conteneur.

- `MemoryHardLimitMebibytes`— Définissez une limite de mémoire maximale pour le conteneur. Si le conteneur dépasse cette limite, cela entraîne un redémarrage.
- `Vcpulimit` — Réservez un minimum de ressources vCPU à l'usage exclusif du conteneur. Le conteneur dispose toujours de la quantité réservée. Il peut dépasser ce minimum à tout moment, si des ressources supplémentaires sont disponibles. (1024 unités de processeur équivalent à 1 vCPU.)

Définir des limites de ressources totales pour un groupe de conteneurs

Si vous définissez des limites pour des conteneurs individuels, vous devrez peut-être modifier la quantité de mémoire et de ressources vCPU dont le groupe de conteneurs a besoin. L'objectif est d'allouer suffisamment de ressources pour optimiser les performances du serveur de jeu. Amazon GameLift Servers utilise ces limites pour calculer comment regrouper les groupes de conteneurs de serveurs de jeu sur une instance de flotte. Vous les utiliserez également lorsque vous choisirez un type d'instance pour un parc de conteneurs.

Calculez la mémoire totale et le vCPU nécessaires pour un groupe de conteneurs. Éléments à prendre en compte :

- Quels sont tous les processus qui s'exécutent dans tous les conteneurs du groupe de conteneurs ? Additionnez les ressources nécessaires à ces processus. Prenez note de toute limite spécifique au contenant.
- Combien de processus de serveur de jeu simultanés prévoyez-vous d'exécuter dans chaque groupe de conteneurs ? Vous le déterminez dans l'image du conteneur de votre serveur de jeu.

Sur la base de votre estimation des exigences relatives aux groupes de conteneurs, définissez les `ContainerGroupDefinition` propriétés suivantes :

- `TotalMemoryLimitMebibytes`— Définissez une limite de mémoire maximale pour le groupe de conteneurs. Tous les conteneurs du groupe partagent la mémoire allouée. Si vous définissez des limites de conteneur individuelles, la limite de mémoire totale doit être égale ou supérieure à la limite de mémoire spécifique au conteneur la plus élevée.
- `TotalVcpuLimit`— Définissez une limite maximale de vCPU pour le groupe de conteneurs. Tous les conteneurs du groupe partagent les ressources CPU allouées. Si vous définissez des limites de processeur individuelles, la limite de processeur totale doit être égale ou supérieure à la somme de toutes les limites de processeur spécifiques au conteneur. Il est recommandé de définir cette valeur pour doubler la somme des limites du processeur du conteneur.

Exemple de scénario

Supposons que nous définissions un groupe de conteneurs de serveurs de jeu avec les trois conteneurs suivants :

- Le conteneur A est le conteneur de notre serveur de jeu. Nous estimons les besoins en ressources d'un serveur de jeu à 512 MiB et 1024 CPU. Nous prévoyons que le conteneur exécute 1 processus serveur. Comme ce conteneur exécute nos logiciels les plus critiques, nous n'avons défini aucune limite de mémoire ni de limite de réserve de vCPU.

- Le conteneur B est un conteneur de support dont les besoins en ressources sont estimés à 1 024 MiB et 1 536 CPU. Nous avons défini une limite de mémoire de 2048 MiB et une limite de réserve CPU de 1024 CPU.
- Le conteneur C est un autre conteneur de support. Nous avons défini une limite de mémoire dure de 512 Mo et une limite de réserve du processeur de 512 processeurs.

À l'aide de ces informations, nous avons défini les limites totales suivantes pour le groupe de conteneurs :

- Limite de mémoire totale : 7680 MiB. Cette valeur dépasse la limite de mémoire maximale (1024 MiB).
- Limite totale du processeur : 13312 processeurs. Cette valeur dépasse la somme de la limite du processeur (1024+512 processeurs).

Désignez les contenants essentiels

Pour un groupe de conteneurs par instance, désignez chaque conteneur comme essentiel ou non essentiel. Les groupes de conteneurs par instance doivent avoir au moins un conteneur de support essentiel. Le conteneur essentiel effectue le travail essentiel du groupe de conteneurs. On s'attend à ce que le contenant essentiel fonctionne toujours. En cas d'échec, l'ensemble du groupe de conteneurs redémarre.

Définissez la `ContainerDefinition` propriété `Essential` sur `true` ou `false` pour chaque conteneur.

Configuration des connexions réseau

Vous pouvez personnaliser l'accès au réseau pour permettre au trafic externe de se connecter à n'importe quel conteneur d'une flotte de conteneurs. Par exemple, vous devez établir des connexions réseau avec le conteneur qui exécute les processus de votre serveur de jeu, afin que les clients du jeu puissent rejoindre votre jeu et y jouer. Les clients de jeu se connectent aux serveurs de jeux à l'aide de ports et d'adresses IP.

Dans une flotte de conteneurs, la connexion entre un client et un serveur n'est pas directe. En interne, un processus dans un conteneur écoute sur un port à conteneurs. En externe, le trafic entrant se connecte à une instance de flotte via un port de connexion. Amazon GameLift Servers gère les mappages entre les ports de conteneur internes et les ports de connexion externes, afin que le trafic entrant soit acheminé vers le processus approprié sur l'instance.

Amazon GameLift Servers fournit un niveau de contrôle supplémentaire pour vos connexions réseau. Chaque flotte de conteneurs dispose d'un paramètre d'autorisations entrantes, qui vous permet de contrôler l'accès à chaque port de connexion externe. Par exemple, vous pouvez supprimer les autorisations pour tous les ports de connexion afin de bloquer tout accès aux conteneurs de la flotte.

Vous pouvez mettre à jour les autorisations entrantes, les ports de connexion et les ports de conteneurs d'une flotte.

Warning

Si vous fournissez un `InstanceConnectionPortRange` ou une valeur personnalisée `InstanceInboundPermissions`, vous ne pouvez plus gérer l'une ou l'autre des valeurs de votre flotte. Vous devez définir les deux champs pour éviter tout comportement indéfini.

Définissez les plages de ports pour conteneurs

Configurez les plages de ports de conteneur dans le cadre de chaque définition de conteneur. Il s'agit d'un paramètre obligatoire pour la définition d'un groupe de conteneurs. Vous devez configurer suffisamment de ports pour prendre en charge tous les processus exécutés simultanément qui nécessitent un accès externe. Certains conteneurs n'auront pas besoin de ports.

Le conteneur de votre serveur de jeu, qui gère vos serveurs de jeu, a besoin d'un port pour chaque processus de serveur de jeu exécuté simultanément. Le processus du serveur de jeu écoute le port assigné et le signale à Amazon GameLift Servers.

Définissez les plages de ports de connexion

Configurez votre flotte de conteneurs avec un ensemble de ports de connexion. Les ports de connexion fournissent un accès externe aux instances de flotte qui exécutent vos conteneurs. Amazon GameLift Servers assigne les ports de connexion et les mappe aux ports de conteneurs selon les besoins.

Par défaut, Amazon GameLift Servers calcule le nombre de ports requis pour tous les groupes de conteneurs et définit une plage de ports adaptée à ces besoins. Nous vous recommandons vivement d'utiliser des valeurs Amazon GameLift Servers calculées, qui sont mises à jour lorsque vous déployez des mises à jour d'une définition de groupe de conteneurs. Si vous devez personnaliser les plages de ports de connexion, suivez les instructions suivantes.

Lorsque vous créez un parc de conteneurs, définissez une plage de ports de connexion (voir [ContainerFleet: InstanceConnectionPortRange](#)). Assurez-vous que la gamme comporte suffisamment de ports pour être mappée à tous les ports à conteneurs définis pour tous les conteneurs des deux groupes de conteneurs de la flotte. Pour calculer le nombre minimal de ports de connexion nécessaires, utilisez la formule suivante :

```
[Total number of container ports defined for containers in the game
server container group] * [Number of game server container groups per
instance] + [Total number of container ports defined for containers in
the per-instance container group]
```

Il est recommandé de doubler le nombre minimum de ports de connexion.

 Note

Le nombre de ports de connexion peut potentiellement limiter le nombre de groupes de conteneurs de serveurs de jeu par instance. Si une flotte ne dispose que de suffisamment de ports de connexion pour un seul groupe de conteneurs de serveurs de jeu par instance, elle ne Amazon GameLift Servers déploiera qu'un seul groupe de conteneurs de serveurs de jeu, même si les instances disposent d'une puissance de calcul suffisante pour plusieurs groupes de conteneurs de serveurs de jeux.

Définir les autorisations de trafic entrant

Les autorisations entrantes contrôlent l'accès externe à une flotte de conteneurs en spécifiant les ports de connexion à ouvrir pour le trafic entrant. Vous pouvez utiliser ce paramètre pour activer ou désactiver l'accès au réseau d'une flotte selon vos besoins.

Par défaut, Amazon GameLift Servers calcule le nombre de ports requis pour tous les groupes de conteneurs et définit une plage de ports adaptée à ces besoins. Nous vous recommandons vivement d'utiliser des valeurs Amazon GameLift Servers calculées, qui sont mises à jour lorsque vous déployez des mises à jour d'une définition de groupe de conteneurs. Si vous devez personnaliser les plages de ports de connexion, suivez les instructions suivantes.

Lorsque vous créez une flotte de conteneurs, définissez un ensemble d'autorisations entrantes (voir [ContainerFleet: InstanceInboundPermissions](#)). Les ports d'autorisation entrants doivent correspondre aux plages de ports de connexion de la flotte.

 Note

Étant donné que les ports de conteneurs sont sélectionnés de manière aléatoire parmi les `InstanceConnectionPortRange`, afin de garantir que les connexions de session peuvent être établies, tous les ports `InstanceConnectionPortRange` entrants doivent être couverts par les ports de `InstanceInboundPermissions`

Exemple de scénario

Cet exemple montre comment définir les trois propriétés de connexion réseau.

- Le groupe de conteneurs de serveurs de jeu de notre flotte comprend 1 conteneur, qui exécute 1 processus de serveur de jeu.

Dans la définition du groupe de conteneurs du serveur de jeu, nous avons défini le `PortConfiguration` paramètre de ce conteneur comme suit :

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP"} ]
}
```

- Notre flotte dispose également d'un groupe de conteneurs par instance comprenant 1 conteneur. Il a 1 processus qui nécessite un accès au réseau. Dans la définition du conteneur par instance, nous avons défini le `PortConfiguration` paramètre de ce conteneur comme suit :

```
"PortConfiguration": {
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ] }
```

- Notre flotte est configurée avec 20 groupes de conteneurs de serveurs de jeu par instance de flotte. À partir de ces informations, nous pouvons utiliser la formule pour calculer le nombre de ports de connexion dont nous avons besoin :
 - Minimum : 21 ports [1 port conteneur de serveur de jeu* 20 groupes de conteneurs de serveurs de jeux par instance + 1 port de conteneur par instance]
 - Meilleure pratique : 42 ports [ports minimum* 2]

Lors de la création de la flotte de conteneurs, nous avons défini le `InstanceConnectionPortRange` paramètre comme suit :

```
"InstanceConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- Nous voulons autoriser l'accès à tous les ports de connexion disponibles. Lors de la création de la flotte de conteneurs, nous avons défini le `InstanceInboundPermissions` paramètre comme suit :

```
"InstanceInboundPermissions": [  
  {"FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":  
    "TCP"} ]
```

Configurez des contrôles de santé pour les conteneurs

Un conteneur redémarre automatiquement en cas de panne du terminal et cesse de fonctionner. Si un conteneur est considéré comme essentiel, il invite l'ensemble du groupe de conteneurs à redémarrer.

Tous les conteneurs de serveurs de jeu sont automatiquement considérés comme essentiels. Les conteneurs de support peuvent être considérés comme essentiels, mais ils doivent être dotés d'un mécanisme permettant de signaler leur état de santé. Vous pouvez également définir des contrôles de santé pour les conteneurs de support non essentiels.

Vous pouvez définir des critères personnalisés supplémentaires pour mesurer l'état du conteneur et utiliser un bilan de santé pour tester ces critères. Pour configurer un contrôle de l'état d'un conteneur, vous pouvez le définir dans une image de conteneur Docker ou dans votre définition de conteneur. Si vous définissez un contrôle de santé dans la définition du conteneur, celui-ci remplace tous les paramètres de l'image du conteneur.

Définissez les `SupportContainerDefinition` propriétés suivantes pour le contrôle de l'état d'un conteneur :

- **Command**— Fournissez une commande qui vérifie certains aspects de l'état du conteneur. Vous décidez des critères à utiliser pour mesurer l'état de santé. La commande doit générer une valeur de sortie de 1 (malsain) ou 0 (sain).
- **StartPeriod**— Spécifiez un délai initial avant que les échecs du bilan de santé ne commencent à être comptabilisés. Ce délai donne au conteneur le temps de démarrer ses processus.
- **Interval**— Décidez à quelle fréquence exécuter la commande de contrôle de santé. À quelle vitesse souhaitez-vous détecter et résoudre une panne de conteneur ?

- **Timeout**— Décidez combien de temps vous devez attendre en cas de succès ou d'échec avant de réessayer la commande de contrôle de santé. Combien de temps la commande de contrôle de santé doit-elle prendre pour s'exécuter ?
- **Retries**— Combien de fois la commande de contrôle de santé doit-elle être réessayée avant d'enregistrer un échec ?

Définir les dépendances des conteneurs

Au sein de chaque groupe de conteneurs, vous pouvez définir des dépendances entre les conteneurs en fonction de l'état du conteneur. Une dépendance a un impact sur le moment où le conteneur dépendant peut démarrer ou s'arrêter en fonction de l'état d'un autre conteneur.

L'un des principaux cas d'utilisation des dépendances consiste à créer des séquences de démarrage et d'arrêt pour le groupe de conteneurs.

Par exemple, vous souhaitez peut-être que le conteneur A démarre en premier et se termine correctement avant le démarrage des conteneurs B et C. Pour ce faire, créez d'abord une dépendance entre le conteneur B et le conteneur A, à condition que le conteneur A soit terminé avec succès. Créez ensuite une dépendance pour le conteneur C sur le conteneur A avec la même condition. Les séquences de démarrage se produisent dans l'ordre inverse de l'arrêt.

Configuration d'une flotte de conteneurs

Lorsque vous créez une flotte de conteneurs, tenez compte des points de décision suivants. La plupart de ces points dépendent de l'architecture et de la configuration de votre conteneur.

Décidez où vous souhaitez déployer votre flotte

En général, vous souhaitez déployer vos flottes géographiquement à proximité de vos joueurs afin de minimiser le temps de latence. Vous pouvez déployer votre flotte de conteneurs vers tous ceux Région AWS qui le prennent Amazon GameLift Servers en charge. Si vous souhaitez déployer le même serveur de jeu dans d'autres zones géographiques, vous pouvez ajouter des sites distants à la flotte, notamment Régions AWS des Zones Locales. Dans le cas d'un parc multisite, vous pouvez ajuster la capacité indépendamment de chaque site du parc. Pour plus d'informations sur les emplacements de flotte pris en charge, consultez [Amazon GameLift Serverspoints de service](#).

Envisagez [Balises ping UDP](#) de collecter des données de latence du réseau dans différentes zones géographiques afin d'anticiper la latence entre les appareils des joueurs et les emplacements potentiels de la flotte. Ces points de terminaison spéciaux acceptent les messages

UDP au lieu des pings ICMP traditionnels, fournissant des mesures de latence précises pour vous aider à sélectionner les emplacements de flotte optimaux.

Choisissez un type et une taille d'instance pour votre flotte

Amazon GameLift Servers prend en charge un large éventail de types d'EC2 instances Amazon, qui sont tous disponibles pour une utilisation avec un parc de conteneurs. La disponibilité et le prix du type d'instance varient en fonction de l'emplacement. Vous pouvez consulter la liste des types d'instances pris en charge, filtrée par emplacement, dans la Amazon GameLift Servers console (sous Ressources, Instance et quotas de service).

Lorsque vous choisissez un type d'instance, considérez d'abord la famille d'instances. Les familles d'instances offrent différentes combinaisons de capacités de processeur, de mémoire, de stockage et de mise en réseau. Obtenez plus d'informations sur les [familles d'EC2 instances](#). Au sein de chaque famille, vous avez le choix entre différentes tailles d'instance. Tenez compte des problèmes suivants lors de la sélection d'une taille d'instance :

- Quelle est la taille d'instance minimale capable de prendre en charge votre charge de travail ? Utilisez ces informations pour éliminer les types d'instances trop petits.
- Quelles tailles de type d'instance conviennent le mieux à votre architecture de conteneur ? Idéalement, vous devez choisir une taille qui puisse accueillir plusieurs copies du groupe de conteneurs de votre serveur de jeu avec un minimum d'espace perdu.
- Quelle granularité de mise à l'échelle convient le mieux à votre jeu ? L'augmentation de la capacité du parc implique l'ajout ou la suppression d'instances, et chaque instance représente la capacité d'héberger un nombre spécifique de sessions de jeu. Déterminez la capacité que vous souhaitez ajouter ou supprimer pour chaque instance. Si la demande des joueurs varie de plusieurs milliers d'une minute à l'autre, il peut être judicieux d'utiliser de très grandes instances pouvant héberger des centaines ou des milliers de sessions de jeu. En revanche, vous préférerez peut-être un contrôle de mise à l'échelle plus précis avec des types d'instances plus petits.
- Des économies sont-elles possibles en fonction de la taille ? Il se peut que le coût de certains types d'instances varie en fonction de la situation géographique en raison de la disponibilité.

Définissez d'autres paramètres de flotte facultatifs

Vous pouvez utiliser les fonctionnalités optionnelles suivantes lors de la configuration d'une flotte de conteneurs :

- Configurez vos serveurs de jeu pour accéder à d'autres AWS ressources. Consultez [Communiquez avec les autres AWS ressources de vos flottes](#).

- Empêchez les sessions de jeu avec des joueurs actifs de se terminer prématurément lors d'un événement à réduction progressive.
- Limitez le nombre de sessions de jeu qu'une personne peut créer dans la flotte dans un laps de temps limité.

Création d'une définition de groupe de conteneurs pour une flotte de Amazon GameLift Servers conteneurs

Une définition de groupe de conteneurs décrit comment déployer vos applications de serveur de jeu conteneurisé sur une flotte de conteneurs. Il s'agit d'un plan Amazon GameLift Servers qui indique les images de conteneurs à déployer dans la flotte et comment les exécuter. Lorsque vous créez une flotte de conteneurs, vous spécifiez les définitions de groupes de conteneurs à déployer dans la flotte. Pour plus d'informations sur les groupes de conteneurs, consultez [Composants du parc de conteneurs](#).

Avant de commencer

Conseils sur la marche à suivre avant de commencer à créer une définition de groupe de conteneurs :

- Finalisez les images de vos conteneurs et transférez-les vers un référentiel Amazon Elastic Container Registry (Amazon ECR) dans le référentiel Région AWS où vous prévoyez de créer le groupe de conteneurs. Amazon GameLift Servers capture un instantané de chaque image au moment où vous créez la définition du groupe de conteneurs, et utilise cet instantané lors du déploiement sur un parc de conteneurs. Consultez [Créez une image de conteneur pour Amazon GameLift Servers](#).
- Créez vos définitions de conteneurs sous forme de fichiers JSON. Une définition de groupe de conteneurs inclut une ou plusieurs définitions de conteneurs. Vous pouvez utiliser les fichiers JSON si vous créez une définition de groupe de conteneurs à l'aide du AWS CLI for Amazon GameLift Servers.
- Vérifiez que votre AWS utilisateur dispose des autorisations IAM pour accéder au référentiel Amazon ECR. Consultez [Exemples d'autorisations IAM pour Amazon GameLift Servers](#).

Création d'une définition de groupe de conteneurs de serveurs de jeu

Un groupe de conteneurs de serveurs de jeux exécute le logiciel de votre serveur de jeu. Un groupe de conteneurs de serveurs de jeu possède un conteneur de serveur de jeu qui exécute le fichier exécutable du serveur de jeu. Il peut également comporter un ou plusieurs conteneurs de support pour exécuter des logiciels supplémentaires destinés à prendre en charge votre serveur de jeu. (Ces conteneurs sont parfois appelés conteneurs « sidecar ».)

Cette rubrique explique comment créer une définition simple de groupe de conteneurs de serveurs de jeu à l'aide de la Amazon GameLift Servers console ou des outils de la AWS CLI. Pour des informations plus détaillées sur les fonctionnalités optionnelles, consultez [Personnalisez une flotte de Amazon GameLift Servers conteneurs](#).

Note

Vous pouvez modifier la plupart des définitions de groupes de conteneurs et des paramètres de définition de conteneur après les avoir créés. Si vous modifiez la définition d'un conteneur, Amazon GameLift Servers capture un nouvel instantané des images du conteneur mises à jour.

Pour créer une définition de groupe de conteneurs de serveurs de jeu simple, procédez comme suit :

Les instructions suivantes décrivent comment créer une définition de groupe de conteneurs avec les paramètres minimaux requis et en utilisant les valeurs Amazon GameLift Servers par défaut.

Console

Dans la [Amazon GameLift Servers console](#), sélectionnez l' Région AWS endroit où vous souhaitez créer le groupe de conteneurs.

Ouvrez la barre de navigation gauche de la console et choisissez Conteneurs gérés : définitions de groupes. Sur la page de définition des groupes de conteneurs, choisissez Créer une définition de groupe.

Étape 1 : définir les détails de la définition du groupe de conteneurs

1. Entrez un nom de définition de groupe de conteneurs. Le nom doit être propre à la région Compte AWS et.
2. Sélectionnez le type de groupe de conteneurs du serveur de jeu.

3. Pour Limite de mémoire totale, entrez les ressources de mémoire maximales à mettre à disposition pour tous les conteneurs du groupe de conteneurs. Pour obtenir de l'aide sur le calcul de cette valeur, consultez [Définissez des limites de ressources](#).
4. Pour la limite totale de vCPU, entrez la puissance de calcul maximale à mettre à disposition pour tous les conteneurs du groupe de conteneurs. Pour obtenir de l'aide sur le calcul de cette valeur, consultez [Définissez des limites de ressources](#).

Étape 2 : ajouter des définitions de conteneurs

Au minimum, un groupe de conteneurs de serveurs de jeux possède un conteneur de serveurs de jeux. Dans la console, la première définition de conteneur que vous créez est le conteneur du serveur de jeu. Cette étape décrit comment définir les paramètres minimaux requis pour la définition d'un conteneur de serveur de jeu.

1. Entrez le nom de la définition du conteneur. Chaque conteneur défini pour le groupe doit avoir une valeur de nom unique.
2. Lien vers une image de conteneur avec la version de votre serveur de jeu. Entrez l'URI de l'image Amazon ECR pour une image de conteneur dans un référentiel public ou privé. Vous pouvez utiliser l'un des formats suivants :
 - URI de l'image uniquement : [Compte AWS].dkr.ecr.[Région AWS].amazonaws.com/[repository ID]
 - URI de l'image et résumé : [Compte AWS].dkr.ecr.[Région AWS].amazonaws.com/[repository ID]@[digest]
 - URI de l'image et balise : [Compte AWS].dkr.ecr.[Région AWS].amazonaws.com/[repository ID]:[tag]
3. Spécifiez la version Amazon GameLift Servers du SDK du serveur utilisée par le build du serveur de jeu. Pour une flotte de conteneurs, cette valeur doit être égale ou supérieure à 5,2.0.
4. Dans Internal container port range, définissez le protocole et définissez une plage de ports. La taille de la plage doit être supérieure au nombre de processus de serveur de jeu simultanés qui s'exécuteront dans ce conteneur. Si le conteneur du serveur de jeu n'exécute qu'un seul processus serveur par conteneur, cette plage de ports n'a besoin que de quelques ports. Pour en savoir plus, consultez [Configuration des connexions réseau](#).
5. Ajoutez d'autres conteneurs selon les besoins pour exécuter des logiciels de support supplémentaires. Les conteneurs supplémentaires sont automatiquement désignés

conteneurs de support. Un groupe de conteneurs de serveurs de jeu ne peut comporter qu'un seul conteneur de serveur de jeu et jusqu'à huit conteneurs de support. Fournissez les paramètres minimaux requis suivants :

- Nom de la définition du conteneur
- URI de l'image ECR.
- Ports de conteneur internes (incluez-les uniquement si le conteneur possède des processus nécessitant un accès au réseau.)

Étape 3 : Configuration des dépendances

- Si votre définition de groupe de conteneurs comporte plusieurs conteneurs, vous pouvez éventuellement définir des dépendances entre les conteneurs. Pour de plus amples informations, veuillez consulter [Définir les dépendances des conteneurs](#).

Étape 3 : Examen et création

1. Passez en revue tous les paramètres de définition de votre groupe de conteneurs. Utilisez Modifier pour apporter des modifications à n'importe quelle section, y compris à chacune de vos définitions de conteneur pour le groupe.
2. Lorsque vous avez terminé de réviser, choisissez Créer.

Si votre demande aboutit, la console affiche la page détaillée de la nouvelle ressource de définition de groupe de conteneurs. Au départ `COPYING`, le statut est le suivant : Amazon GameLift Servers commence à prendre des instantanés de toutes les images du conteneur pour le groupe. Lorsque cette phase est terminée, le statut de définition du groupe de conteneurs passe à `READY`. Une définition de groupe de conteneurs doit avoir le `READY` statut requis pour que vous puissiez créer une flotte de conteneurs à l'aide de cette définition.

AWS CLI

Lorsque vous utilisez la AWS CLI pour créer une définition de groupe de conteneurs, conservez vos configurations de définition de conteneur dans un JSON fichier séparé. Vous pouvez référencer le fichier dans votre commande CLI. Voir [Création d'un JSON fichier de définition de conteneur](#) pour des exemples de schéma.

Création d'une définition de groupe de conteneurs

Pour créer une nouvelle définition de groupe de conteneurs, utilisez la commande `create-container-group-definition` CLI. Pour plus d'informations sur cette commande, consultez la référence [create-container-group-definition](#) des commandes de la AWS CLI.

Cet exemple illustre une demande de définition de groupe de conteneurs de serveurs de jeux. Cela suppose que vous avez créé un fichier JSON contenant les définitions de conteneur pour ce groupe.

```
aws gamelift create-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --game-server-container-definition file://MyAdventureGameContainers.json
```

Création d'un **JSON** fichier de définition de conteneur

Lorsque vous créez une définition de groupe de conteneurs, vous définissez également les conteneurs du groupe. Une définition de conteneur spécifie le référentiel Amazon ECR dans lequel l'image du conteneur est stockée, ainsi que les configurations facultatives pour les ports réseau, les limites d'utilisation du processeur et de la mémoire, ainsi que d'autres paramètres. Nous vous recommandons de créer un JSON fichier unique contenant les configurations de tous les conteneurs d'un groupe de conteneurs. La gestion d'un fichier est utile pour le stockage, le partage et le suivi des versions de ces configurations critiques. Si vous utilisez la AWS CLI pour créer vos définitions de groupes de conteneurs, vous pouvez référencer le fichier dans la commande.

Pour créer une définition de conteneur

1. Créez et ouvrez un nouveau .JSON fichier. Par exemple :

```
[~/work/glc]$ vim SimpleServer.json
```

2. Créez une définition de conteneur distincte pour chacun des conteneurs du groupe. Copiez l'exemple de contenu suivant et modifiez-le selon les besoins de vos conteneurs. Pour plus de détails sur la syntaxe d'une définition de conteneur, consultez [ContainerDefinitionInput](#) la référence de l'Amazon GameLift Servers API.

3. Enregistrez le fichier localement afin de pouvoir y faire référence dans une commande AWS CLI.

Exemple : définition du conteneur d'un serveur de jeu

Exemple

Cet exemple décrit le conteneur essentiel pour le groupe de conteneurs de votre serveur de jeu. Le conteneur de répliques essentiel inclut votre application de serveur de jeu, l'Amazon GameLift Serversagent, et peut inclure d'autres logiciels de support pour l'hébergement de votre jeu. La définition doit inclure un nom, une URI d'image et une configuration de port. Cet exemple définit également certaines limites de ressources spécifiques au conteneur.

```
{
  "ContainerName": "MyAdventureGameServer",
  "ImageUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/gl-
containers:myadventuregame-server",
  "PortConfiguration": {
    "ContainerPortRanges": [
      {
        "FromPort": 2000,
        "Protocol": "TCP",
        "ToPort": 2010
      }
    ]
  },
  "ServerSdkVersion": "5.2.0"
}
```

Mettre à jour la définition d'un groupe de conteneurs pour un Amazon GameLift Servers flotte de conteneurs

Vous pouvez mettre à jour la plupart des propriétés d'une définition de groupe de conteneurs, y compris les définitions de conteneurs individuelles. Les définitions de groupes de conteneurs ont un numéro de version. Lorsque vous mettez à jour la définition d'un groupe de conteneurs, Amazon GameLift Servers enregistre la mise à jour et incrémente le numéro de version de la définition. Lorsque vous configurez un parc de conteneurs, vous pouvez spécifier la version d'une définition de groupe de conteneurs à déployer.

Après avoir mis à jour la définition d'un groupe de conteneurs, vous pouvez déployer la nouvelle version sur un parc de conteneurs nouveau ou existant.

Mettre à jour la définition d'un groupe de conteneurs de serveurs de jeu

Cette rubrique explique comment mettre à jour la définition d'un groupe de conteneurs de serveurs de jeu à l'aide du Amazon GameLift Servers outils de console ou de AWS CLI. Pour des informations plus détaillées sur les fonctionnalités optionnelles, consultez [Personnalisez une flotte de Amazon GameLift Servers conteneurs](#).

Pour mettre à jour la définition d'un groupe de conteneurs :

Console

Dans [Amazon GameLift Servers console](#), sélectionnez l' Région AWS endroit où vous souhaitez créer le groupe de conteneurs.

Ouvrez la barre de navigation gauche de la console et choisissez Conteneurs gérés : définitions de groupes. Sur la page de définition des groupes de conteneurs, choisissez une définition et une version de groupe de conteneurs à mettre à jour.

Après avoir enregistré vos mises à jour, vous pouvez utiliser la nouvelle version pour créer de nouvelles flottes de conteneurs ou déployer les mises à jour sur une flotte de conteneurs existante.

Étape 1 : définir les détails de la définition du groupe de conteneurs

- Vous pouvez mettre à jour les paramètres de mémoire totale et de limite de vCPU.

Étape 2 : ajouter des définitions de conteneurs

Vous pouvez effectuer les mises à jour de définition de conteneur suivantes :

- Mettez à jour les définitions de conteneurs existantes.
 - Ajoutez de nouvelles définitions de conteneurs de support.
 - Supprimez les définitions des conteneurs de support.
1. Vous pouvez mettre à jour l'URI de l'image ECR. Assurez-vous de mettre à jour le paramètre de version du SDK du serveur pour qu'il corresponde à la nouvelle image.
 2. Vous pouvez mettre à jour la plage de ports de conteneurs internes selon vos besoins. Les modifications que vous apportez à ces paramètres peuvent avoir un impact sur les paramètres du port de connexion d'une flotte de conteneurs lorsque ces modifications sont

déployées sur une flotte. Pour en savoir plus, consultez [Configuration des connexions réseau](#).

Étape 3 : Configuration des dépendances

- Vous pouvez modifier les dépendances selon vos besoins. Pour de plus amples informations, veuillez consulter [Définir les dépendances des conteneurs](#).

Étape 3 : Examen et création

- Passez en revue les mises à jour des définitions de vos groupes de conteneurs. Utilisez Modifier pour apporter des modifications supplémentaires dans n'importe quelle section. Lorsque vous avez terminé, choisissez Create pour générer une nouvelle version de la définition du groupe de conteneurs.

Si votre demande aboutit, la console affiche la page détaillée de la nouvelle ressource de définition de groupe de conteneurs. Au départ, le statut est COPYING le suivant : Amazon GameLift Servers commence à prendre des instantanés de toutes les images du conteneur pour le groupe. Lorsque cette phase est terminée, le statut de définition du groupe de conteneurs passe àREADY. Une définition de groupe de conteneurs doit avoir le READY statut requis pour que vous puissiez créer une flotte de conteneurs à l'aide de cette définition.

AWS CLI

Lorsque vous utilisez la AWS CLI pour créer ou mettre à jour une définition de groupe de conteneurs, conservez vos configurations de définition de conteneur dans un JSON fichier séparé. Vous pouvez référencer le fichier dans votre commande CLI. Voir [Création d'un JSON fichier de définition de conteneur](#) pour des exemples de schéma.

Lorsque vous mettez à jour une définition, il vous suffit de spécifier les valeurs que vous souhaitez mettre à jour. Amazon GameLift Servers conserve toutes les valeurs que vous n'incluez pas dans votre demande de mise à jour. Si vous modifiez la définition d'un conteneur. Toutefois, lorsque vous modifiez une définition de conteneur, fournissez un ensemble complet.

Pour mettre à jour la définition d'un groupe de conteneurs

Pour mettre à jour une nouvelle définition de groupe de conteneurs, utilisez la commande `update-container-group-definition` CLI. Pour plus d'informations sur cette

commande, consultez le manuel [update-container-group-definition](#) de référence des commandes de la AWS CLI.

Exemple : groupe de conteneurs de serveurs de jeux

Vous pouvez spécifier une version de définition de groupe de conteneurs lors de la récupération, de la mise à jour ou de la suppression d'une définition de groupe de conteneurs, ou lors de la création ou de la mise à jour d'un parc de conteneurs. Chaque définition de groupe de conteneurs possède une propriété de version. En outre, la valeur ARN de la définition et indique le numéro de version.

Cet exemple illustre une demande de modification de la définition d'un groupe de conteneurs de serveurs de jeux. Cela suppose que vous avez créé un fichier JSON contenant les définitions de conteneur pour ce groupe. Cet exemple utilise la valeur ARN pour le nom de la définition et indique que la mise à jour concerne la version 1.

```
aws gamelift update-container-group-definition \
  --name arn:aws:gamelift:us-west-2:111122223333:containergroupdefinition/
  MyAdventureGameContainerGroup:1 \
  --operating-system AMAZON_LINUX_2023 \
  --container-group-type GAME_SERVER \
  --total-memory-limit-mebibytes 4096 \
  --total-vcpu-limit 1 \
  --container-definitions file://SimpleServer.json
```

Cloner une définition de groupe de conteneurs

Vous pouvez utiliser le plugin Amazon GameLift Servers console pour cloner une définition de groupe de conteneurs existante.

Pour cloner un groupe de conteneurs

1. Dans [Amazon GameLift Servers console](#), allez dans le volet de navigation de gauche et choisissez Groupes de conteneurs.
2. Sur la page de liste des groupes de conteneurs, sélectionnez le groupe de conteneurs existant que vous souhaitez cloner. Une fois que vous avez sélectionné un groupe de conteneurs, le bouton Cloner est actif.
3. Choisissez Clone (Cloner). Cette action ouvre l'assistant de création de groupes de conteneurs avec des paramètres préremplis.

4. Entrez un nouveau nom pour le groupe de conteneurs cloné. Les groupes de conteneurs d'une même région doivent avoir des noms uniques.
5. Parcourez les pages du groupe de conteneurs et de définition du conteneur, passez en revue et créez le nouveau groupe de conteneurs.

Supprimer la définition d'un groupe de conteneurs pour Amazon GameLift Servers flotte de conteneurs

Plusieurs options s'offrent à vous pour supprimer une définition de groupe de conteneurs. Lorsque vous supprimez une définition de groupe de conteneurs, cette action supprime également toutes les définitions de conteneurs du groupe de conteneurs.

Les définitions de groupes de conteneurs peuvent avoir plusieurs versions. Les versions des groupes de conteneurs portent le même nom, mais un numéro de version différent. La définition du groupe de conteneurs ARNs spécifie à la fois le nom et la version.

Vous pouvez spécifier une version de définition de groupe de conteneurs lors de la récupération, de la mise à jour ou de la suppression d'une définition de groupe de conteneurs, ou lors de la création ou de la mise à jour d'un parc de conteneurs. Chaque définition de groupe de conteneurs possède une propriété de version. En outre, la valeur ARN de la définition indique le numéro de version.

Il existe plusieurs méthodes pour supprimer des définitions de groupes de conteneurs :

- Vous pouvez supprimer toutes les versions d'une définition spécifique.
- Vous pouvez supprimer une version particulière d'une définition spécifique.
- Vous pouvez conserver un certain nombre de versions les plus récentes et supprimer les anciennes versions d'une définition spécifique. Par exemple, vous pouvez supprimer toutes les versions antérieures à la version 5.

Vous pouvez supprimer une version de définition de groupe de conteneurs uniquement si elle n'est pas utilisée dans un parc de conteneurs.

Pour supprimer une définition de groupe de conteneurs

Console

Dans [Amazon GameLift Servers console](#), sélectionnez l' Région AWS endroit où vous souhaitez créer le groupe de conteneurs.

Ouvrez la barre de navigation gauche de la console et choisissez Conteneurs gérés : définitions de groupes. Sur la page des définitions des groupes de conteneurs, sélectionnez la définition que vous souhaitez modifier et choisissez Supprimer.

Vous êtes invité à sélectionner le type de suppression que vous souhaitez effectuer et à définir des paramètres supplémentaires en fonction du type de suppression.

AWS CLI

- Pour supprimer une définition de groupe de conteneurs, utilisez la commande `delete-container-group-definition` CLI et fournissez des valeurs pour le type de suppression que vous souhaitez effectuer. Pour plus d'informations sur cette commande, consultez le manuel [delete-container-group-definition](#) de référence des commandes de la AWS CLI.

Cet exemple illustre une demande de suppression de toutes les versions de la définition de groupe de conteneurs de serveurs de jeu antérieures à la version 5.

Exemple

```
aws gamelift delete-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --version-count-to-retain 5 \
```

Amazon GameLift Servers Des flottes partout

Utiliser Anywhere des flottes dont vous souhaitez profiter Amazon GameLift Servers fonctionnalités avec vos propres ressources d'hébergement. Les flottes Anywhere sont couramment utilisées comme environnements de test pour le développement itératif ou aux côtés de flottes gérées dans une solution d'hébergement hybride.

Un parc Anywhere consiste en un ensemble de ressources informatiques (virtuelles ou physiques) que vous fournissez et gérez. Les ordinateurs peuvent se trouver dans n'importe quel emplacement géographique connecté, qu'il s'agisse d'un ordinateur portable local ou d'un avant-poste distant. Lorsque vous configurez une flotte Anywhere, vous ajoutez des ordinateurs à la flotte en les enregistrant via Amazon GameLift Servers. Chaque ordinateur est enregistré avec son adresse IP (ou son nom DNS) afin que Amazon GameLift Servers peut établir une connexion avec elle.

Vous déployez le logiciel de serveur de jeu sur un parc Anywhere en l'installant sur chaque ordinateur et en lançant les processus du serveur de jeu. Chaque processus de serveur de jeu lancé établit une

connexion avec Amazon GameLift Servers service et indique qu'il est prêt à héberger une session de jeu. Vous pouvez utiliser vos outils de gestion de configuration et de déploiement existants pour gérer les tâches de déploiement initial et de gestion de l'hôte. Quelques tâches supplémentaires sont requises pour l'utilisation avec Amazon GameLift Servers, y compris :

- Enregistrez et désenregistrez des ordinateurs pour les ajouter ou les supprimer de la flotte.
- Conservez les jetons up-to-date d'authentification sur tous les ordinateurs. Les processus du serveur sur le calcul les utilisent lors de la connexion au Amazon GameLift Servers service.

Note

Déployez éventuellement votre flotte Anywhere avec Amazon GameLift Servers Agent pour automatiser ces tâches de gestion clés. Consultez [Collaborez avec l'Amazon GameLift Serversagent](#).

Consultez les rubriques suivantes pour savoir comment configurer et gérer des flottes Anywhere :

- [Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers Anywhere](#)
- [Feuille de route de développement pour l'hébergement hybride avec Amazon GameLift Servers](#)
- [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#)
- [Créez un Amazon GameLift Servers Flotte Anywhere](#)
- [Comment fonctionne Amazon GameLift Servers la création de flottes](#)
- [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#)

Workflow de création de flotte en tout lieu

Pour les flottes Anywhere, Amazon GameLift Servers configure uniquement la ressource de flotte. Vous configurez et enregistrez des ordinateurs auprès de la flotte, vous installez le logiciel du serveur de jeu et lancez les processus du serveur de jeu pour héberger les sessions de jeu.

1. Amazon GameLift Servers crée les ressources de la flotte dans la région d'origine de la flotte.
L'état de la flotte et le statut de localisation personnalisé sont définis sur Nouveau.
2. Amazon GameLift Servers commence à consigner les événements dans le journal des événements de la flotte.

3. Après la création de la ressource de flotte. Amazon GameLift Servers définit le statut de la flotte sur Actif. À ce stade, vous pouvez enregistrer de nouveaux ordinateurs dans la flotte.

Créez un Amazon GameLift Servers Flotte Anywhere

Cette rubrique décrit comment créer un Amazon GameLift Servers Une flotte n'importe où. Avec une flotte Anywhere, vous pouvez utiliser le noyau Amazon GameLift Servers fonctionnalités de gestion des sessions de jeu tout en hébergeant des sessions de jeu avec vos propres ressources informatiques. Créez une flotte Anywhere pour votre matériel sur site ou d'autres ressources basées sur le cloud.

Partout où les flottes sont couramment utilisées Amazon GameLift Servers flottes gérées dans une solution d'hébergement hybride. Ils fournissent également des environnements de test utiles lors du développement d'un jeu à héberger avec Amazon GameLift Servers. Consultez ces rubriques pour en savoir plus sur le moment et la manière de procéder à l'incorporation Amazon GameLift Servers Anywhere intègre une solution d'hébergement de jeux :

- [Hébergement hybride](#)
- [Configuration d'un parc d'hébergement avec Amazon GameLift Servers](#)
- [Configurez le développement itératif avec Amazon GameLift Servers N'importe où](#)

Les flottes Anywhere étant autogérées, leur mise en place nécessite un travail supplémentaire. Pour préparer une flotte Anywhere à accueillir des sessions de jeu et des joueurs, vous devez effectuer les tâches suivantes :

Rubriques

- [Avant de commencer](#)
- [Créez un emplacement personnalisé](#)
- [Créez une flotte Anywhere](#)
- [Ajouter un ordinateur à la flotte](#)
- [Démarrez un serveur de jeu](#)

Avant de commencer

Avant de créer une flotte Anywhere, effectuez les tâches suivantes. Pour des conseils plus détaillés, consultez le [Feuille de route de développement pour l'hébergement avec Amazon GameLift Servers](#)

[Anywhere](#) ou [Feuille de route de développement pour l'hébergement hybride avec Amazon GameLift Servers](#).

- Intégrez le code de votre serveur de jeu au Amazon GameLift Servers SDK du serveur version 5.x (ou supérieure). Vous n'avez pas besoin d'effectuer toutes les tâches d'intégration au jeu, uniquement celles requises pour la création d'un serveur de jeu. Une pratique courante consiste à configurer votre machine locale en tant que flotte Anywhere et à utiliser une interface de ligne de commande pour tester l'intégration de votre serveur de jeu (voir [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#)). Vous pouvez intégrer des composants supplémentaires (tels qu'un Amazon GameLift Servers client de jeu activé) au fur et à mesure que vous les développez.
- Package du logiciel de votre serveur de jeu pour l'installer sur les ordinateurs de votre flotte Anywhere. Le package doit inclure la version de votre serveur de jeu intégré et tous les logiciels de support nécessaires au fonctionnement de votre serveur de jeu.
- Décidez si vous souhaitez utiliser le Amazon GameLift Servers Agent auprès de votre flotte Anywhere. L'agent est un outil de gestion des processus sur ordinateur qui automatise certaines des tâches clés liées à la gestion des processus du serveur et des calculs à utiliser avec Amazon GameLift Servers. Pour plus d'informations, consultez [Collaborez avec l'Amazon GameLift Serversagent](#).

Créez un emplacement personnalisé

Créez un emplacement personnalisé pour représenter l'emplacement physique de vos ressources informatiques. Lorsque vous créez une flotte Anywhere, au moins un emplacement personnalisé doit déjà être défini. Vous pouvez créer des emplacements personnalisés supplémentaires et les ajouter à un parc existant à tout moment.

Pour créer un emplacement personnalisé

Utilisez soit le Amazon GameLift Servers console ou le AWS Command Line Interface (AWS CLI) pour créer un emplacement personnalisé.

Console

Dans [Amazon GameLift Servers console](#), utilisez le volet de navigation pour ouvrir la page Emplacements. Choisissez Créer un emplacement pour ouvrir la boîte de dialogue Créer.

1. Dans la boîte de dialogue, entrez le nom de l'emplacement. Il est recommandé d'utiliser un nom qui décrit un emplacement significatif pour un ensemble de ressources de calcul. Il peut s'agir d'emplacements géographiques, d'un nom de centre de données ou d'un autre identifiant de localisation. Amazon GameLift Servers ajoute le nom de votre emplacement personnalisé avec custom-.
2. (Facultatif) Ajoutez des balises à votre emplacement personnalisé. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez. Attribuez des balises aux AWS ressources que vous souhaitez classer de manière utile, par exemple par objectif, propriétaire ou environnement. Choisissez Ajouter un nouveau tag pour chaque tag que vous souhaitez ajouter.
3. Sélectionnez Create (Créer).

AWS CLI

Créez un emplacement personnalisé à l'aide de la [create-location](#) commande. Entrez une location-name valeur, qui doit commencer par custom-. Il est recommandé d'utiliser un nom qui décrit un emplacement significatif pour un ensemble de ressources de calcul. Il peut s'agir d'emplacements géographiques, d'un nom de centre de données ou d'un autre identifiant de localisation.

```
aws gamelift create-location \
  --location-name custom-location-1
```

Sortie

```
{
  "Location": {
    "LocationName": "custom-location-1",
    "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-
location-1"
  }
}
```

Créez une flotte Anywhere

Créez une flotte Anywhere pour un ensemble de ressources informatiques dont vous êtes propriétaire. Une nouvelle flotte Anywhere commence à être vide ; vous ajoutez des ordinateurs à la flotte en les enregistrant.

Lors de sa création, une nouvelle flotte Anywhere passe rapidement de statut NEW à ACTIVE. Vous pouvez ajouter des calculs à la flotte une fois celle-ci atteinte. ACTIVE

Pour créer une flotte Anywhere

Utilisez soit le Amazon GameLift Servers console ou le AWS Command Line Interface (AWS CLI) pour créer une flotte Anywhere.

Console

Dans [Amazon GameLift Servers console](#), utilisez le volet de navigation pour ouvrir la page Fleets. Choisissez Créer une flotte pour démarrer le flux de travail de création de flotte.

Étape 1 Choisissez le type de calcul

Sélectionnez l'option N'importe où, puis Next.

Étape 2 Définir les détails de la flotte

Au cours de cette étape, spécifiez certains paramètres clés pour l'ensemble de la flotte.

1. Remplissez la section Détails de la flotte :
 - a. Entrez un nom de flotte. Nous recommandons d'utiliser un modèle de dénomination des flottes qui facilite l'identification des types de flottes lors de la consultation des listes de flottes.
 - b. Fournissez une brève description de la flotte.
2. Définissez ces informations supplémentaires facultatives selon vos besoins. Vous pourrez mettre à jour ces paramètres de flotte ultérieurement.
 - a. Lorsque vous créez une flotte à des fins de production ou de pré-production, utilisez ce paramètre pour spécifier une valeur de coût horaire pour les calculs de la flotte. Amazon GameLift Servers peut utiliser ces informations pendant le processus de placement des sessions de jeu pour sélectionner les ressources d'hébergement en fonction du coût.

- b. Si vous souhaitez combiner les données métriques de ce parc et d'autres, spécifiez un nom de groupe de métriques. Utilisez le même nom de groupe de mesures pour toutes les flottes que vous souhaitez combiner. Consultez les métriques du groupe de métriques pour voir les données agrégées.
3. Ajoutez des balises facultatives à votre position personnalisée. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez. Attribuez des balises aux AWS ressources que vous souhaitez classer de manière utile, par exemple par objectif, propriétaire ou environnement. Choisissez Ajouter un nouveau tag pour chaque tag que vous souhaitez ajouter.
4. Choisissez Next pour continuer le flux de travail.

Étape 3 Sélectionnez des emplacements personnalisés

Au cours de cette étape, identifiez l'emplacement physique des ordinateurs que vous prévoyez d'ajouter à cette flotte. Vous pouvez définir un ou plusieurs emplacements dès maintenant, et vous pouvez ajouter ou supprimer des emplacements ultérieurement selon vos besoins.

1. Dans Emplacements personnalisés, sélectionnez un ou plusieurs emplacements pour les calculs de la flotte. La liste inclut tous les emplacements personnalisés qui ont été définis dans votre sélection actuelle Région AWS. Pour définir un nouvel emplacement personnalisé que vous souhaitez ajouter à la flotte, choisissez Créer un emplacement.
2. Choisissez Next pour continuer le flux de travail.

Étape 4 Réviser et créer

Vérifiez vos paramètres avant de créer la flotte.

Lorsque vous êtes prêt à déployer la nouvelle flotte, choisissez Create. Amazon GameLift Servers lance immédiatement le processus d'activation de la flotte, en attribuant un identifiant unique et en plaçant la flotte dans NEW son statut. Vous pouvez suivre la progression de la flotte sur la page Flottes.

AWS CLI

Utilisez la [create-fleet](#) commande pour créer un parc de types de calcul ANYWHERE. Entrez un nom et au moins un emplacement personnalisé. Amazon GameLift Servers crée la ressource de

flotte Anywhere dans votre ressource par défaut actuelle Région AWS (ou vous pouvez ajouter une balise `--region` pour en spécifier une autre Région AWS).

L'exemple de demande suivant crée une nouvelle flotte avec les paramètres minimaux requis. Remplacez *FleetName* et *custom-location* par vos propres informations.

```
aws gamelift create-fleet \  
--name FleetName \  
--compute-type ANYWHERE \  
--locations "Location=custom-location"
```

Exemple de réponse

```
{  
  "FleetAttributes": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "Name": "HardwareAnywhere",  
    "CreationTime": "2023-02-23T17:57:42.293000+00:00",  
    "Status": "ACTIVE",  
    "MetricGroups": [  
      "default"  
    ],  
    "CertificateConfiguration": {  
      "CertificateType": "DISABLED"  
    },  
    "ComputeType": "ANYWHERE"  
  }  
}
```

Lors de sa création, une nouvelle flotte Anywhere passe rapidement au statut de flotte `ACTIVE`. Vous pouvez ajouter des calculs à la flotte une fois celle-ci atteinte. `ACTIVE`

Notez que la réponse n'inclut pas les emplacements de la flotte. Vous pouvez obtenir les détails complets de la flotte en appelant [describe-fleet-attributes](#) et [describe-fleet-location-attributes](#).

Ajouter un ordinateur à la flotte

Pour ajouter une ressource de calcul à une flotte et la préparer à héberger des sessions de jeu, effectuez les tâches suivantes :

- Enregistrez le calcul auprès de la flotte. L'inscription indique Amazon GameLift Servers quelles ressources d'hébergement physiques font partie de la flotte.
- Demandez un jeton d'authentification pour le calcul. Chaque serveur de jeu qui s'exécute sur l'ordinateur a besoin de ce jeton pour se connecter au Amazon GameLift Servers service. Les jetons d'authentification sont temporaires et doivent être régulièrement actualisés.

Note

Si vous déployez le logiciel de votre serveur de jeu avec Amazon GameLift Servers Agent, vous pouvez ignorer cette étape. L'agent enregistre automatiquement chaque calcul et conserve un jeton d'authentification valide pour le calcul. Consultez [Collaborez avec l'Amazon GameLift Servers agent](#).

Vous pouvez enregistrer un calcul et demander un jeton d'authentification à l'aide de la AWS CLI ou en effectuant des appels programmatiques au AWS SDK pour Amazon GameLift Servers. Ces actions ne sont pas disponibles via le Amazon GameLift Servers console.

Il est recommandé d'automatiser ces deux tâches en ajoutant un script de démarrage à chaque calcul. Le script de démarrage appelle automatiquement les `get-compute-auth-token` commandes `register-compute` et. Vous pouvez également automatiser les tâches pour actualiser régulièrement le jeton d'authentification pendant toute la durée de vie du calcul et annuler l'enregistrement du calcul à l'arrêt.

Chacune des actions de démarrage renvoie des valeurs spécifiques au calcul que vous devez stocker sur le calcul. Lorsqu'un processus de serveur de jeu démarre sur le calcul, il doit transmettre ces valeurs en tant que paramètres du serveur lors de l'initialisation d'une connexion avec le Amazon GameLift Servers service (voir [ServerParameters](#) la référence du SDK du serveur). Nous vous recommandons de définir ces valeurs spécifiques au calcul (ou leur emplacement de stockage) en tant que variables d'environnement. Si vous utilisez le Amazon GameLift Servers Agent, cette tâche est gérée pour vous. Les valeurs spécifiques au calcul sont les suivantes :

- `register-compute` renvoie une valeur pour `GameLiftServiceSdkEndpoint`. Définissez cette valeur sur le paramètre `websocketUrl` du serveur.
- `compute-auth-token` renvoie le jeton d'authentification. Définissez cette valeur sur le paramètre `authToken` du serveur.

AWS CLI

Les instructions suivantes décrivent comment soumettre manuellement chaque demande à l'aide de la AWS CLI.

Pour enregistrer un ordinateur

Appelez [register-compute](#) pour enregistrer un ordinateur. Identifiez l'ID de la flotte à laquelle ajouter le calcul. Fournissez les informations de calcul suivantes : un nom, une adresse IP et un emplacement significatifs. L'emplacement du calcul doit être un emplacement personnalisé déjà associé à la flotte. Si vous souhaitez utiliser un autre emplacement personnalisé, utilisez Amazon GameLift Servers console pour mettre à jour le parc ou appeler la commande AWS CLI [create-fleet-locations](#) pour ajouter un emplacement personnalisé au parc.

Dans l'exemple suivant, remplacez les valeurs d'espace réservé pour votre calcul et votre flotte. La `fleet-id` valeur est renvoyée lorsque vous créez une flotte Anywhere. Vous pouvez obtenir les détails complets de la flotte en appelant [describe-fleet-attributes](#) et [describe-fleet-location-attributes](#).

```
aws gamelift register-compute \  
  --compute-name HardwareAnywhere \  
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --ip-address 10.1.2.3 \  
  --location custom-location-1
```

Exemple de sortie

```
{  
  "Compute": {  
    "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "FleetArn": "arn:aws:gamelift:us-west-2:111122223333:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",  
    "ComputeName": "HardwareAnywhere",
```

```

    "ComputeArn": "arn:aws:gamelift:us-west-2:111122223333:compute/
HardwareAnywhere",
    "IpAddress": "10.1.2.3",
    "ComputeStatus": "Active",
    "Location": "custom-location-1",
    "CreationTime": "2023-02-23T18:09:26.727000+00:00",
    "GameLiftServiceSdkEndpoint": "wss://us-west-2.api.amazongamelift.com"
  }
}

```

Pour demander un jeton d'authentification

Appelez [get-compute-auth-token](#) pour demander un jeton d'authentification valide. Enregistrez un calcul. Identifiez l'ID du parc et le nom du calcul.

Dans l'exemple suivant, remplacez les valeurs d'espace réservé pour votre calcul et votre flotte. La `fleet-id` valeur est renvoyée lorsque vous créez une flotte Anywhere. Vous pouvez obtenir les détails complets de la flotte en appelant [describe-fleet-attributes](#). Pour obtenir des informations de calcul, appelez [list-compute](#) avec l'identifiant de la flotte pour voir tous les calculs enregistrés dans la flotte.

```

aws gamelift get-compute-auth-token \
  --fleet-id arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --compute-name HardwareAnywhere

```

Exemple de sortie

```

{
  "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "FleetArn": "arn:aws:gamelift:us-east-1:111122223333:fleet/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
  "ComputeName": "HardwareAnywhere",
  "ComputeArn": "arn:aws:gamelift:us-east-1:111122223333:compute/
HardwareAnywhere",
  "AuthToken": "0c728041-3e84-4aaa-b927-a0fb202684c0",
  "ExpirationTimestamp": "2023-02-23T18:47:54+00:00"
}

```

Démarrez un serveur de jeu

Après avoir créé une flotte Anywhere et ajouté un ou plusieurs ordinateurs à la flotte, vous êtes prêt à commencer à gérer vos serveurs de jeu.

Étape 1 Installez le logiciel de votre serveur de jeu

Créez votre serveur de jeu et installez tous les logiciels qui en dépendent sur chaque ordinateur de votre flotte Anywhere. La version du serveur de jeu doit être intégrée à Amazon GameLift Servers SDK de serveur version 5.x (ou supérieure) avec les fonctionnalités minimales requises pour communiquer avec Amazon GameLift Servers service.

Étape 2 Préparez vos ordinateurs pour exécuter un serveur de jeu

Assurez-vous que chaque ordinateur est enregistré et possède un jeton d'authentification valide. Si vous utilisez des scripts pour gérer ces tâches, assurez-vous qu'ils s'exécutent sur chaque ordinateur avant de démarrer tout processus du serveur de jeu.

Si vous avez déployé le Amazon GameLift Servers Agent avec le logiciel de votre serveur de jeu, assurez-vous que le fichier exécutable de l'agent est lancé.

Étape 3 Lancer un processus de serveur de jeu

Exécutez une instance de l'exécutable de votre serveur de jeu sur un ordinateur. Si la version de votre serveur de jeu est correctement intégrée, le processus du serveur de jeu appelle l'action du SDK du serveur `InitSDK()` avec un ensemble de paramètres de serveur valides. Lorsque le processus serveur est prêt à héberger une session de jeu, il appelle `ProcessReady()`.

Note

Si vous avez déployé le logiciel de votre serveur de jeu avec Amazon GameLift Servers Agent, vous pouvez ignorer cette étape. L'agent lance automatiquement les processus du serveur de jeu en fonction des instructions d'exécution que vous fournissez.

Vous pouvez suivre la progression en consultant les métriques des processus du serveur pour l'activation et les processus actifs du serveur. Consultez [Métriques Amazon GameLift Servers pour les flottes](#). Si le processus de votre serveur de jeu ne parvient pas à s'initialiser, vérifiez qu'il récupère les bonnes valeurs de paramètres du serveur pour le calcul sur lequel il s'exécute.

Abstraire un Amazon GameLift Servers désignation de la flotte avec un alias

Un Amazon GameLift Servers un alias est utilisé pour abstraire une destination d'hébergement. Les destinations d'hébergement racontent Amazon GameLift Servers où rechercher les ressources disponibles pour organiser une nouvelle session de jeu pour les joueurs. Les alias sont utiles dans les scénarios suivants :

- Si votre jeu n'utilise pas de files d'attente impliquant plusieurs flottes pour le placement des sessions de jeu, il demande de nouvelles sessions de jeu en spécifiant un Amazon GameLift Servers numéro de flotte. Au cours de la durée de vie d'un jeu, vous remplacerez la flotte plusieurs fois, pour mettre à jour la version d'un serveur, mettre à jour le matériel d'hébergement et le système d'exploitation, ou résoudre des problèmes de performances. Utilisez un alias pour extraire l'identifiant de la flotte afin de pouvoir transférer facilement le trafic des joueurs d'une flotte existante vers une nouvelle flotte.
- Si vous souhaitez faire autre chose que créer une nouvelle session de jeu lorsqu'un client de jeu en fait la demande. Par exemple, vous souhaitez peut-être rediriger les joueurs qui utilisent un out-of-date client vers un site Web de mise à niveau.

Un alias doit spécifier une stratégie de routage. Il en existe deux types. Une stratégie de routage simple achemine le trafic des joueurs vers un identifiant de flotte spécifique, que vous pouvez mettre à jour pour rediriger le trafic. Une stratégie de routage du terminal renvoie un message au client au lieu de créer une nouvelle session de jeu. Vous pouvez modifier la stratégie de routage d'un alias à tout moment.

Si vous utilisez une file d'attente pour le placement des sessions de jeu, vous n'avez pas besoin d'alias pour rediriger le trafic lors du remplacement d'une flotte. Avec une file d'attente, vous pouvez simplement ajouter la nouvelle flotte et supprimer l'ancienne flotte. Cette action n'est pas visible pour les joueurs, car les nouvelles demandes de session de jeu sont automatiquement satisfaites à l'aide de la nouvelle flotte. Cela n'a aucun impact sur les sessions de jeu existantes. Vous pouvez identifier les destinations des files d'attente à l'aide d'un identifiant de flotte ou d'un alias.

Rubriques

- [Créez un Amazon GameLift Servers alias](#)
- [Modifier un alias](#)

Créez un Amazon GameLift Servers alias

Cette rubrique décrit comment créer un Amazon GameLift Servers alias à utiliser pour le placement des sessions de jeu.

Pour créer un alias

Utilisez l'un ou l'autre Amazon GameLift Servers console ou le AWS Command Line Interface (AWS CLI) pour créer un alias.

Console

Dans [Amazon GameLift Servers console](#), utilisez le volet de navigation pour ouvrir la page Alias.

1. Choisissez Créer un alias.
2. Entrez un nom d'alias. Nous vous recommandons d'inclure des caractéristiques significatives dans le nom d'alias pour faciliter l'affichage d'une liste d'alias.
3. Entrez une description de l'alias selon vos besoins.
4. Choisissez une stratégie de routage pour l'alias.
 - a. Si vous choisissez une stratégie de routage simple, sélectionnez un identifiant de flotte dans la liste à associer à cet alias. La liste inclut toutes les flottes actuellement sélectionnées Région AWS. Vous devez créer un alias dans la même région que la flotte.
 - b. Si vous choisissez une stratégie de routage de terminal, entrez la valeur de chaîne que vous souhaitez Amazon GameLift Servers pour revenir à un client de jeu en réponse à une demande de session de jeu. Une demande avec un alias de terminal génère une exception avec le message intégré.
5. (Facultatif) Ajoutez des balises à la ressource d'alias. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez. Attribuez des balises aux AWS ressources que vous souhaitez classer de manière utile, par exemple par objectif, propriétaire ou environnement. Choisissez Ajouter un nouveau tag pour chaque tag que vous souhaitez ajouter.
6. Lorsque vous êtes prêt à déployer la nouvelle flotte, choisissez Create.

AWS CLI

Utilisez la [create-alias](#) commande pour créer un alias. Amazon GameLift Servers crée la ressource alias dans votre ressource par défaut actuelle Région AWS (ou vous pouvez ajouter une balise `--region` pour en spécifier une autre Région AWS).

Incluez au minimum un nom d'alias et une stratégie de routage. Pour une stratégie de routage simple, spécifiez l'ID d'une flotte dans la même région que l'alias. Pour une stratégie de routage de terminal, fournissez une chaîne de message.

Modifier un alias

Vous pouvez modifier un alias à l'aide du Amazon GameLift Servers console ou à l'aide de la commande AWS CLI [update-alias](#).

Cette rubrique décrit comment modifier un Amazon GameLift Servers alias à utiliser pour le placement des sessions de jeu. Vous pouvez apporter les modifications suivantes :

Pour modifier un alias

Utilisez l'un ou l'autre Amazon GameLift Servers console ou le AWS Command Line Interface (AWS CLI) pour modifier un alias.

Console

Dans [Amazon GameLift Servers console](#), utilisez le volet de navigation pour ouvrir la page Alias.

1. Sélectionnez l'alias que vous souhaitez modifier, puis cliquez sur Modifier. Si vous ne voyez pas l'alias que vous souhaitez modifier, vérifiez qu'il est actuellement sélectionné Région AWS.
2. Sur la page Modifier l'alias, vous pouvez apporter les modifications suivantes :
 - Modifiez le nom de l'alias.
 - Modifiez la description de l'alias.
 - Changez la stratégie de routage de simple à terminal, ou de terminal à simple.
 - Pour un alias doté d'une stratégie de routage simple, modifiez l'ID de flotte auquel l'alias est associé.
 - Pour un alias doté d'une stratégie de routage de terminal, modifiez le texte du message.

3. Sélectionnez Enregistrer les modifications. Lors de la mise à jour de l'ID de flotte pour un alias à l'aide d'une stratégie de routage simple, la transition peut prendre jusqu'à 2 minutes. Pendant ce temps, de nouveaux emplacements de session de jeu peuvent avoir lieu sur l'ancienne flotte.

AWS CLI

Utilisez la [update-alias](#) commande pour apporter des modifications à une ressource d'alias. Vous pouvez mettre à jour une ressource d'alias dans votre ressource par défaut actuelle Région AWS, ou vous pouvez ajouter une `--region` balise pour en spécifier une autre Région AWS.

Vous pouvez modifier les propriétés suivantes :

- Nom d'alias.
- Description de l'alias.
- Type de stratégie de routage. Assurez-vous de fournir un identifiant de flotte ou une chaîne de message pour la nouvelle stratégie de routage.
- Numéro de flotte pour une stratégie de routage simple existante. L'ID de flotte doit se trouver dans la même région que l'alias.
- Chaîne de message pour une stratégie de routage de terminal existante.

Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente

Une file d'attente de sessions de jeu est le principal mécanisme Amazon GameLift Servers utilisé pour rechercher les serveurs de jeu disponibles et les choisir pour héberger de nouvelles sessions de jeu. Les files d'attente constituent un moyen bien plus efficace de traiter un grand nombre de demandes de sessions de jeu et de les placer dans plusieurs flottes de ressources d'hébergement. Si votre solution d'hébergement utilise plusieurs flottes et que vous traitez de gros volumes de demandes, vous aurez probablement besoin d'une file d'attente.

Lorsque votre jeu souhaite démarrer une nouvelle session de jeu pour les joueurs, il envoie une demande de placement au Amazon GameLift Servers service, qui l'achemine vers la file d'attente. La configuration de la file d'attente détermine quand et comment les demandes sont traitées. Lors du traitement d'une demande de placement, Amazon GameLift Servers recherche dans un ensemble de flottes un serveur de jeu qui hébergera la session de jeu. Le placement réussit lorsqu'il Amazon GameLift Servers trouve un serveur de jeu disponible et l'invite à démarrer une session de jeu.

Rubriques

- [Caractéristiques des files d'attente](#)
- [Création d'une file d'attente de session de jeu](#)
- [Personnaliser une file d'attente de session de jeu](#)
- [Configurer la notification des événements pour le placement des sessions de jeu](#)
- [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#)

Caractéristiques des files d'attente

Une file d'attente de session de Amazon GameLift Servers jeu est une ressource AWS cloud. Vous pouvez créer une file d'attente dans Région AWS n'importe quel Amazon GameLift Servers support (voir [Amazon GameLift Servers points de service](#)). Les demandes de placement de session de jeu sont envoyées à cet endroit et y sont traitées.

L'automatisation du placement des sessions de jeu à l'aide de files d'attente offre des avantages significatifs tant pour les développeurs de jeux que pour les joueurs. Il s'agit des licences suivantes :

- Les files d'attente offrent le « meilleur placement possible ». Lors du traitement des demandes de placement de session de jeu, une file d'attente utilise l'algorithme Amazon GameLift Servers FleetIQ pour hiérarchiser les emplacements en fonction d'un ensemble de préférences définies, notamment le coût, l'emplacement et le temps de latence des joueurs.
- Les files d'attente prennent en charge les flottes Spot afin de réduire les coûts d'hébergement de jeux. Vous pouvez configurer vos files d'attente avec des flottes AWS Spot, qui offrent souvent des coûts d'hébergement nettement inférieurs, ainsi qu'avec des flottes à la demande. Le faible coût étant l'un des principaux critères de placement, les files d'attente peuvent toujours tirer parti des différences de coûts.
- Les files d'attente peuvent placer de nouveaux jeux plus rapidement en cas de forte demande. En configurant une file d'attente composée de plusieurs flottes, vous offrez des options plus flexibles pour le placement des sessions de jeu. Mais des flottes supplémentaires fournissent également une capacité de sauvegarde lorsque la demande augmente. Pour toute demande de placement, si Amazon GameLift Servers vous ne parvenez pas à placer une session de jeu dans l'emplacement préféré, elle passe automatiquement à l'évaluation des autres emplacements.
- Les files d'attente peuvent rendre la disponibilité des serveurs de jeu plus résiliente. Des pannes peuvent survenir. Dans le cas d'une file d'attente composée de plusieurs flottes, un ralentissement ou une panne ne doivent pas nécessairement affecter l'accès des joueurs à votre jeu. En configurant votre file d'attente avec des flottes dotées de capacités dans différentes Régions AWS zones de disponibilité, vous pouvez faire en sorte que les joueurs puissent toujours trouver une session de jeu à rejoindre.
- Obtenez des statistiques sur le placement des sessions de jeu et les performances des files d'attente. Amazon GameLift Servers met des statistiques sur les files d'attente, notamment des statistiques sur les réussites et les échecs de placement, le nombre de demandes dans la file d'attente et le temps moyen passé par les demandes dans la file d'attente. Vous pouvez également consulter ces métriques à l'aide de la console Amazon GameLift Servers ou dans CloudWatch.

Pour commencer par créer une file d'attente de démarrage de base, consultez [Création d'une file d'attente de session de jeu](#).

Création d'une file d'attente de session de jeu

Les files d'attente sont utilisées pour placer de nouvelles sessions de jeu sur plusieurs flottes et sites. Votre jeu démarre de nouvelles sessions de jeu en soumettant des demandes de placement à une file d'attente. Une file d'attente est configurée avec des instructions sur le traitement des

demandes. Pour en savoir plus sur le lancement de demandes de placement dans une session de jeu, consultez [Créez des sessions de jeu](#).

Pour créer une file d'attente de session de jeu

Ces instructions montrent comment créer une file d'attente de travail simple avec des paramètres de configuration minimaux et des paramètres par défaut. Il existe plusieurs options pour personnaliser la configuration d'une file d'attente. Ces options vous aident à effectuer les meilleurs placements possibles en fonction des besoins de votre jeu. Pour en savoir plus sur la personnalisation des files d'attente pour votre jeu, consultez [Personnaliser une file d'attente de session de jeu](#). Vous pouvez mettre à jour la plupart des paramètres de configuration des files d'attente à tout moment.

Vous pouvez créer une file d'attente de session de jeu à l'aide de la Amazon GameLift Servers console ou de la AWS CLI.

Console

Dans la [Amazon GameLift Servers console](#), sélectionnez AWS la région dans laquelle vous souhaitez travailler. Ouvrez la barre de navigation gauche de la console et choisissez Queues.

1. Sur la page Files d'attente, choisissez Créer une file d'attente pour démarrer le flux de travail.
2. Dans Paramètres de file d'attente, entrez les paramètres suivants :
 - a. Entrez le nom de la file d'attente. Ce nom doit être unique à Région AWS celui dans lequel vous créez la file d'attente.
 - b. Conservez le paramètre de délai d'expiration par défaut, qui est de 600 secondes (ou 10 minutes). Cette valeur contrôle la durée des Amazon GameLift Servers tentatives de création d'une nouvelle session de jeu avant de s'arrêter. Amazon GameLift Servers recherche les ressources disponibles jusqu'à expiration de la demande. Vous pouvez mettre à jour le paramètre de délai d'expiration d'une file d'attente à tout moment.
 - c. Ignorez la section Politiques de latence des joueurs. Une file d'attente utilise des politiques de latence uniquement lorsqu'elle reçoit des demandes de placement qui incluent des données de latence des joueurs. Vous pouvez ajouter des politiques de latence à une file d'attente à tout moment. Pour en savoir plus sur la création de politiques de latence, consultez [Créez une politique de latence pour les joueurs](#).
3. Ignorez la section Emplacements de placement des sessions de jeu pour utiliser le paramètre par défaut Tous les emplacements. Ce paramètre vous permet de créer une liste d'emplacements autorisés dans lesquels la file d'attente peut effectuer des placements

(également appelée configuration de filtre). Pour en savoir plus sur la hiérarchisation par emplacement et les configurations de filtre, consultez [Prioriser les placements par emplacement](#).

4. Sous Ordre de destination, ajoutez une ou plusieurs flottes à la file d'attente. Vous pouvez identifier les flottes en utilisant fleet IDs ou ARNs en utilisant un alias de flotte. Lorsque vous ajoutez plusieurs flottes, n'oubliez pas qu'elles doivent toutes exécuter des versions de jeu similaires et être compatibles avec tout client de jeu utilisant cette file d'attente. En outre, toutes les flottes d'une file d'attente doivent avoir la même configuration de certificats.
 - a. Sélectionnez la région dans laquelle la flotte ou l'alias a été créé. Pour une flotte à sites multiples, il s'agit de la « région d'origine ».
 - b. Pour le type de destination, sélectionnez une flotte ou un alias.
 - c. Vos sélections de région et de type alimentent une liste déroulante des flottes ou alias existants. Sélectionnez-en une à désigner comme destination de file d'attente.
 - d. Pour spécifier une autre flotte ou un autre alias pour la file d'attente, choisissez Ajouter une destination et répétez les étapes précédentes.
 - e. Après avoir ajouté une liste de destinations, utilisez drag-and-drop cette fonctionnalité pour réorganiser les destinations. Amazon GameLift Servers utilise cet ordre pour hiérarchiser les emplacements par destination.
5. Ignorez la section Priorité de placement des sessions de jeu pour conserver l'ordre de priorité par défaut. Ce paramètre vous permet de Amazon GameLift Servers choisir où rechercher les ressources d'hébergement disponibles pour les nouveaux emplacements de session de jeu. Pour plus d'informations sur la hiérarchisation des emplacements, consultez [Prioriser le placement des sessions de jeu](#). Vous pouvez mettre à jour les priorités de placement d'une file d'attente à tout moment.
6. Sous Ordre de localisation, conservez les valeurs par défaut. Ce paramètre est utilisé lors de la hiérarchisation en fonction de l'emplacement de la flotte. Il fournit un ordre de localisation à utiliser. Lorsque vous utilisez les paramètres de priorité par défaut, l'emplacement est utilisé comme facteur décisif lorsque la destination préférée est une flotte composée de plusieurs sites.
7. Ignorez la section facultative des paramètres de notification d'événements. Les notifications d'événements sont requises pour les files d'attente qui traitent un grand nombre de demandes de placement. Pour les files d'attente qui traitent de faibles volumes, par exemple à des fins de développement ou de test, vous pouvez suivre l'état des demandes de placement en interrogeant auprès [DescribeGameSessionPlacement](#). Pour en savoir plus,

consultez [Configurer la notification des événements pour le placement des sessions de jeu](#). Vous pouvez mettre à jour les paramètres de notification d'événements d'une file d'attente à tout moment.

8. Choisissez Créer pour générer une nouvelle file d'attente avec un minimum de personnalisation.

AWS CLI

Exemple Créer une file d'attente

L'exemple suivant crée une file d'attente de session de jeu avec les configurations suivantes :

- Un délai d'attente de cinq minutes.
- Deux destinations de la flotte.
- Filtrez pour autoriser uniquement les placements dans les emplacements suivants :us-east-1,us-east-2.us-west-2, etca-central-1.
- Ordre prioritaire basé sur le coût, puis sur les emplacements dans un ordre spécifié.

```
aws gamelift create-game-session-queue \  
  --name "sample-test-queue" \  
  --timeout-in-seconds 300 \  
  --destinations DestinationArn="arn:aws:gamelift:us-east-1:111122223333:fleet/  
fleet-772266ba-8c82-4a6e-b620-a74a62a93ff8" DestinationArn="arn:aws:gamelift:us-  
east-1:111122223333:fleet/fleet-33f28fb6-aa8b-4867-85b4-ceb217bf5994" \  
  --filter-configuration "AllowedLocations=us-east-1, ca-central-1, us-east-2, us-  
west-2" \  
  --priority-configuration PriorityOrder="COST","LOCATION",LocationOrder="us-  
east-1","us-east-2","ca-central-1","us-west-2" \  
  --notification-target "arn:aws:sns:us-east-1:111122223333:gamelift-test.fifo"
```

Note

Vous pouvez obtenir les valeurs ARN de la flotte et de l'alias en appelant l'un [describe-fleet-attributes](#) ou l'autre en appelant [describe-alias](#) avec l'ID de flotte ou d'alias.

Si la `create-game-session-queue` demande aboutit, Amazon GameLift Servers renvoie un [GameSessionQueue](#) objet avec la nouvelle configuration de file d'attente. Vous pouvez désormais envoyer des demandes à la file d'attente en utilisant [StartGameSessionPlacement](#).

Exemple Créez une file d'attente avec des politiques de latence pour les joueurs

L'exemple suivant crée une file d'attente de session de jeu avec les configurations suivantes :

- Un délai d'attente de dix minutes
- Trois destinations de la flotte
- Un ensemble de politiques de latence des joueurs

```
aws gamelift create-game-session-queue \  
  --name "matchmaker-queue" \  
  --timeout-in-seconds 600 \  
  --destinations DestinationArn=arn:aws:gamelift:us-east-1::alias/alias-a1234567-  
b8c9-0d1e-2fa3-b45c6d7e8910 \  
                  DestinationArn=arn:aws:gamelift:us-west-2::alias/alias-b0234567-  
c8d9-0e1f-2ab3-c45d6e7f8901 \  
                  DestinationArn=arn:aws:gamelift:us-west-2::fleet/fleet-f1234567-  
b8c9-0d1e-2fa3-b45c6d7e8912 \  
  --player-latency-policies  
  "MaximumIndividualPlayerLatencyMilliseconds=50,PolicyDurationSeconds=120" \  
  
  "MaximumIndividualPlayerLatencyMilliseconds=100,PolicyDurationSeconds=120" \  
  "MaximumIndividualPlayerLatencyMilliseconds=150" \  
  \
```

Si la `create-game-session-queue` demande aboutit, Amazon GameLift Servers renvoie un [GameSessionQueue](#) objet avec la nouvelle configuration de file d'attente.

Personnaliser une file d'attente de session de jeu

Cette rubrique explique comment personnaliser les files d'attente de vos sessions de jeu afin de prendre les meilleures décisions possibles concernant le placement des sessions de jeu. Pour plus d'informations sur les files d'attente des sessions de jeu et leur fonctionnement, consultez [Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente](#).

Ces Amazon GameLift Servers fonctionnalités nécessitent des files d'attente :

- [Mise en relation avec FlexMatch](#)
- [Concevoir une file d'attente pour les instances Spot](#)

Rubriques

- [Bonnes pratiques en matière de files d'attente pour les sessions de Amazon GameLift Servers jeu](#)
- [Définition de l'étendue d'une file d'attente](#)
- [Créez une file d'attente multi-sites](#)
- [Prioriser le placement des sessions de jeu](#)
- [Évaluer les métriques relatives aux files](#)
- [Concevoir une file d'attente pour les instances Spot](#)

Bonnes pratiques en matière de files d'attente pour les sessions de Amazon GameLift Servers jeu

Une file d'attente de session de jeu contient une liste de flottes dans lesquelles Amazon GameLift Servers vous pouvez placer de nouvelles sessions de jeu. Chaque flotte peut avoir des ressources d'hébergement déployées dans plusieurs emplacements géographiques. Lors du choix d'un emplacement, la file d'attente sélectionne une flotte et un emplacement de flotte en fonction d'un ensemble de priorités que vous définissez pour la flotte.

Tenez compte des directives et des meilleures pratiques suivantes :

- Ajoutez des flottes dans des lieux qui couvrent vos joueurs. Vous pouvez ajouter des flottes et des alias dans n'importe quel emplacement disponible. La localisation est importante si vous effectuez des placements en fonction de la latence signalée par les joueurs.
- Utilisez des alias pour toutes les flottes. Attribuez un alias à chaque flotte d'une file d'attente et utilisez les noms d'alias lorsque vous définissez des destinations dans votre file d'attente.
- Utilisez une version ou un script de jeu identique ou similaire pour toutes les flottes. La file d'attente peut permettre aux joueurs de participer à des sessions de jeu sur n'importe quelle flotte de la file d'attente. Les joueurs doivent pouvoir participer à n'importe quelle session de jeu sur n'importe quelle flotte.
- Créez des flottes sur au moins deux sites. En hébergeant des serveurs de jeu sur au moins un autre site, vous réduisez l'impact des pannes régionales sur vos joueurs. Vous pouvez réduire la

taille de vos flottes de sauvegarde et utiliser le dimensionnement automatique pour augmenter la capacité si l'utilisation augmente.

- Priorisez le placement de votre session de jeu. Une file d'attente hiérarchise les choix de placement en fonction de plusieurs éléments, notamment l'ordre de la liste des destinations.
- Créez votre file d'attente au même endroit que votre service client. En plaçant votre file d'attente à proximité de votre service client, vous pouvez minimiser le temps de latence des communications.
- Utilisez des flottes comportant plusieurs sites. Utilisez la configuration du filtre de file d'attente pour empêcher la file d'attente de placer les sessions de jeu à des emplacements spécifiques. Vous pouvez utiliser au moins deux flottes comportant plusieurs sites d'attache différents pour atténuer l'impact des emplacements de jeu en cas de panne régionale.
- Utilisez le même paramètre de certificat TLS pour toutes les flottes. Les clients de jeu qui se connectent aux sessions de jeu de vos flottes doivent disposer de protocoles de communication compatibles.

Définition de l'étendue d'une file d'attente

La population de joueurs de votre jeu peut comprendre des groupes de joueurs qui ne devraient pas jouer ensemble. Par exemple, si vous publiez votre jeu en deux langues, chaque langue doit disposer de ses propres serveurs de jeu.

Pour configurer le placement des sessions de jeu pour votre population de joueurs, créez une file d'attente distincte pour chaque segment de joueurs. Délimitez chaque file d'attente pour placer les joueurs sur les bons serveurs de jeu. Voici quelques méthodes courantes pour délimiter les files d'attente :

- Par localisation géographique. Lorsque vous déployez vos serveurs de jeu dans plusieurs zones géographiques, vous pouvez créer des files d'attente pour les joueurs de chaque site afin de réduire la latence des joueurs.
- Par des variantes de build ou de script. Si vous avez plusieurs variantes de votre serveur de jeu, il se peut que vous souteniez des groupes de joueurs qui ne peuvent pas jouer lors des mêmes sessions de jeu. Par exemple, les versions ou les scripts des serveurs de jeu peuvent prendre en charge différents langages ou types d'appareils.
- Par type d'événement. Vous pouvez créer une file d'attente spéciale pour gérer les parties destinées aux participants à des tournois ou à d'autres événements spéciaux.

Concevez plusieurs files d'attente

En fonction de votre jeu et de vos joueurs, vous souhaitez peut-être créer plusieurs files d'attente de sessions de jeu. Lorsque le service client de votre jeu demande une nouvelle session de jeu, il indique la file d'attente de session de jeu à utiliser. Pour vous aider à déterminer s'il convient d'utiliser plusieurs files d'attente, considérez les points suivants :

- Variantes de votre serveur de jeu. Vous pouvez créer une file d'attente distincte pour chaque variante de votre serveur de jeu. Toutes les flottes d'une file d'attente doivent déployer des serveurs de jeu compatibles. En effet, les joueurs qui utilisent la file d'attente pour rejoindre des parties doivent pouvoir jouer sur l'un des serveurs de jeu de la file d'attente.
- Différents groupes de joueurs. Vous pouvez personnaliser le placement Amazon GameLift Servers des sessions de jeu en fonction du groupe de joueurs. Par exemple, vous pouvez avoir besoin de files d'attente personnalisées pour certains modes de jeu nécessitant un type d'instance ou une configuration d'exécution spécifiques. Vous pouvez également avoir besoin d'une file d'attente spéciale pour gérer les emplacements pour un tournoi ou un autre événement.
- Statistiques de la file d'attente des sessions de jeu. Vous pouvez configurer des files d'attente en fonction de la manière dont vous souhaitez collecter les statistiques de placement des sessions de jeu. Pour de plus amples informations, veuillez consulter [Métriques Amazon GameLift Servers pour les files d'attente](#).

Créez une file d'attente multi-sites

Nous recommandons une conception multi-sites pour toutes les files d'attente. Cette conception peut améliorer la vitesse de placement et la résilience de l'hébergement. Une conception multi-sites est nécessaire pour utiliser les données de latence des joueurs afin de permettre aux joueurs de participer à des sessions de jeu avec une latence minimale. Si vous créez des files d'attente multisites utilisant des flottes d'instances Spot, suivez les instructions figurant dans. [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#)

L'un des moyens de créer une file d'attente multi-sites consiste à ajouter une [flotte multi-sites](#) à une file d'attente. Ainsi, la file d'attente peut placer des sessions de jeu dans n'importe quel emplacement de la flotte. Vous pouvez également ajouter d'autres flottes avec des configurations ou des emplacements d'origine différents à des fins de redondance. Si vous utilisez un parc d'instances Spot multi-sites, suivez les meilleures pratiques et incluez un parc d'instances à la demande avec les mêmes emplacements.

L'exemple suivant décrit le processus de conception d'une file d'attente multisite de base. Dans cet exemple, nous utilisons deux flottes : une flotte d'instances Spot et une flotte d'instances à la demande. Chaque flotte possède les emplacements Régions AWS de placement suivants : `us-east-1`, `us-east-2`, `ca-central-1`, et `us-west-2`.

Pour créer une file d'attente multisite de base avec des flottes multi-sites

1. Choisissez un emplacement pour créer la file d'attente. Vous pouvez minimiser la latence des demandes en plaçant la file d'attente à proximité de l'endroit où vous avez déployé le service client. Dans cet exemple, nous créons la file d'attente dans `us-east-1`.
2. Créez une nouvelle file d'attente et ajoutez vos flottes multi-sites comme destinations de file d'attente. L'ordre de destination détermine le Amazon GameLift Servers lieu des sessions de jeu. Dans cet exemple, nous listons d'abord le parc d'instances Spot et ensuite le parc d'instances à la demande.
3. Définissez l'ordre de priorité de placement des sessions de jeu de la file d'attente. Cet ordre détermine l'endroit où la file d'attente recherche en premier un serveur de jeu disponible. Dans cet exemple, nous utilisons l'ordre de priorité par défaut.
4. Définissez l'ordre de localisation. Si vous ne définissez pas l'ordre des emplacements, Amazon GameLift Servers utilise les emplacements par ordre alphabétique.

Prioriser le placement des sessions de jeu

Amazon GameLift Servers utilise un algorithme pour déterminer comment prioriser les destinations d'une file d'attente et déterminer où placer une nouvelle session de jeu. L'algorithme est basé sur un ensemble ordonné de critères. Vous pouvez utiliser l'ordre de priorité par défaut ou le personnaliser. Vous pouvez modifier l'ordre de priorité d'une file d'attente à tout moment.

Ordre de priorité par défaut

1. Latence : si la demande de placement de session de jeu inclut des données de latence spécifiques à l'emplacement des joueurs, Amazon GameLift Servers calcule la latence moyenne des joueurs dans chaque emplacement et tente de placer une session de jeu dans un emplacement de flotte présentant la moyenne la plus faible.

2. Coût — Si une demande n'inclut pas de données de latence, ou si plusieurs flottes ont une latence égale, le coût d'hébergement de chaque flotte est Amazon GameLift Servers évalué. Le coût d'hébergement d'une flotte varie en fonction du type de flotte (Spot ou On-Demand), du type d'instance et de l'emplacement.
3. Destination — Si plusieurs flottes ont le même temps de latence et les mêmes coûts, Amazon GameLift Servers hiérarchise les flottes en fonction de l'ordre de destination indiqué dans la configuration de la file d'attente.
4. Emplacement — Pour les files d'attente comportant des flottes à sites multiples, si tous les autres critères sont égaux, Amazon GameLift Servers hiérarchise les emplacements du parc en fonction de l'ordre alphabétique.

Personnalisez la façon dont une file d'attente hiérarchise les emplacements des sessions de jeu

Vous pouvez choisir de personnaliser la façon dont une file d'attente hiérarchise les critères de placement. La file d'attente applique la priorisation personnalisée à toutes les demandes de placement de session de jeu qu'elle reçoit.

Note

Si vous créez une configuration de priorité personnalisée et que vous n'incluez pas les quatre critères, les critères manquants Amazon GameLift Servers sont automatiquement ajoutés dans l'ordre par défaut.

Pour personnaliser la configuration des priorités d'une file d'attente

Utilisez la [Amazon GameLift Serversconsole](#) ou le AWS Command Line Interface (AWS CLI) pour créer une configuration de priorité personnalisée.

Console

Dans la [Amazon GameLift Serversconsole](#), vous pouvez personnaliser les priorités d'une file d'attente lors de la création d'une nouvelle file d'attente ou de la mise à jour d'une file d'attente existante. Sélectionnez AWS la région dans laquelle vous souhaitez travailler.

Ouvrez la barre de navigation gauche de la console et choisissez Queues. Sur la page Files d'attente, sélectionnez une file d'attente existante et choisissez Modifier.

1. Accédez à la section **Priorité de placement des sessions de jeu**. Faites glisser et déposez chaque critère de priorité pour créer l'ordre souhaité.
2. Accédez à la section **Ordre de localisation**. Ajoutez les emplacements que vous souhaitez prioriser. Cette liste est utile lorsque la file d'attente comporte des flottes comportant plusieurs emplacements. Vous devez au minimum spécifier un emplacement. Les emplacements que vous spécifiez ici sont priorisés en premier, suivis de tous les autres emplacements figurant dans les destinations de la file d'attente.
3. Sélectionnez **Enregistrer les modifications**.

AWS CLI

Utilisez la [update-game-session-queue](#) commande avec l'`--priority-configuration` option permettant de personnaliser l'ordre de priorité d'une file d'attente. Amazon GameLift Servers met à jour une file d'attente dans votre AWS région par défaut actuelle, ou vous pouvez ajouter une `--region` balise pour spécifier une autre AWS région.

L'exemple de demande suivant ajoute ou met à jour la configuration de priorité pour une file d'attente spécifiée.

```
aws gamelift update-game-session-queue \  
  --name "example-queue-with-priority" \  
  --priority-configuration \  
    PriorityOrder="COST", 'LOCATION', "DESTINATION", LocationOrder="us-east-1", "us- \  
east-2", "ca-central-1", "us-west-2" \  

```

Prioriser les placements en fonction du temps de latence

Si vous souhaitez offrir à vos joueurs la meilleure expérience de jeu possible et garantir une latence minimale, suivez les étapes suivantes lors de la configuration de votre système de placement de sessions de jeu :

- Définissez votre file d'attente pour prioriser la latence lorsque vous choisissez où placer les sessions de jeu. La latence figure en haut de la liste des priorités par défaut. Vous pouvez également personnaliser la configuration des priorités de votre file d'attente et choisir où placer la latence par ordre de priorité.
- Configurez des politiques de latence des joueurs pour votre file d'attente. Les politiques de latence vous permettent de définir des limites strictes quant à la latence à autoriser lors du placement

d'une session de jeu. Si Amazon GameLift Servers vous ne parvenez pas à placer une session de jeu sans dépasser les limites, la demande de placement expirera et échouera. Vous pouvez configurer une seule politique de latence ou créer une série de politiques qui assouplissent progressivement la limite de latence au fil du temps. Grâce à une série de règles, vous pouvez définir des limites de latence initiales très faibles, tout en répondant aux besoins des joueurs présentant des latences plus élevées après un court délai. Pour plus de détails sur la création de politiques de latence, consultez [Créez une politique de latence pour les joueurs](#).

- Lorsque vous faites des demandes de placement dans une session de jeu (voir [StartGameSessionPlacement](#)), incluez les données de latence pour chaque joueur. Les données de latence des joueurs incluent une valeur pour chaque emplacement possible où une session de jeu peut être placée. Par exemple, pour une file d'attente qui place les sessions de jeu dans Régions AWS us-east-2 et ca-central-1, les données de latence peuvent ressembler à ceci :

```
"PlayerLatencies": [  
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "us-east-2" },  
  { "LatencyInMilliseconds": 100, "PlayerId": "player1", "RegionIdentifier": "ca-central-1" },  
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "us-east-2" },  
  { "LatencyInMilliseconds": 150, "PlayerId": "player2", "RegionIdentifier": "ca-central-1" }  
]
```

Pour obtenir des mesures de latence précises, utilisez Amazon GameLift Servers les balises ping UDP. Ces points de terminaison vous permettent de mesurer la latence réelle du réseau UDP entre les appareils des joueurs et chacun des sites d'hébergement potentiels, ce qui permet de prendre des décisions de placement plus précises que l'utilisation de pings ICMP. Pour plus d'informations sur l'utilisation de balises ping UDP pour mesurer la latence, reportez-vous à [Balises ping UDP](#)

Prioriser les placements par emplacement

Vous pouvez configurer une file d'attente pour effectuer des placements dans les sessions de jeu en fonction d'une liste hiérarchisée d'emplacements géographiques. L'emplacement est l'un des critères qui déterminent comment une file d'attente choisit où placer une nouvelle session de jeu. Par défaut, la localisation est priorisée en quatrième position, après la latence, le coût et la destination.

En ce qui concerne le placement des sessions de jeu, la destination et le lieu ont des significations quelque peu différentes :

- La destination fait référence à une flotte spécifique et inclut toutes les ressources d'hébergement de la flotte, où qu'elles soient déployées. Lorsque vous établissez des priorités par destination, vous Amazon GameLift Servers pouvez effectuer un placement avec n'importe quel emplacement de la flotte. Les flottes gérées sur plusieurs sites et les flottes Anywhere peuvent disposer de ressources d'hébergement déployées sur un ou plusieurs sites.
- L'emplacement fait référence à une position géographique spécifique où les ressources d'hébergement d'une flotte sont déployées. Une flotte peut avoir plusieurs emplacements Régions AWS, notamment des Zones Locales ou des emplacements personnalisés (pour une flotte Anywhere). Une flotte gérée sur un seul site possède un seul emplacement et c'est toujours un Région AWS. Une flotte gérée sur plusieurs sites possède une région d'origine et peut avoir des sites distants. Une flotte Anywhere possède un ou plusieurs emplacements personnalisés.

Lorsque vous hiérarchisez les emplacements par emplacement, Amazon GameLift Servers recherche toutes les destinations de file d'attente qui incluent l'emplacement prioritaire et y recherche une ressource d'hébergement disponible. S'il existe plusieurs destinations avec l'emplacement prioritaire, Amazon GameLift Servers passe au critère de priorité suivant (coût, latence, destination).

Vous pouvez influencer la hiérarchisation des emplacements d'une file d'attente de plusieurs manières.

- Configurez la façon dont la file d'attente gère toutes les demandes de placement de session de jeu :
 - Ajoutez une configuration de priorité à la file d'attente. La configuration des priorités d'une file d'attente inclut une liste ordonnée d'emplacements. Vous pouvez définir un ou plusieurs emplacements à prioriser. Cette liste n'exclut aucun emplacement, elle indique simplement Amazon GameLift Servers où rechercher en premier lieu une ressource d'hébergement disponible. Une liste d'emplacements ordonnée est souvent utilisée lorsque vous souhaitez canaliser la majeure partie du trafic vers un ou plusieurs emplacements géographiques spécifiques et utiliser des emplacements supplémentaires comme capacité de sauvegarde. Ajoutez une configuration de priorité en appelant [UpdateGameSessionQueue](#).
 - Ajoutez une configuration de filtre à la file d'attente. Une configuration de filtre est une liste d'autorisations pour la file d'attente. Il indique Amazon GameLift Servers d'ignorer les emplacements qui ne figurent pas sur la liste lorsque vous recherchez une ressource d'hébergement disponible. Il existe deux utilisations courantes de la configuration d'un filtre. Tout

d'abord, pour les flottes comportant plusieurs sites, vous pouvez utiliser un filtre pour exclure certains emplacements de la flotte. Ensuite, vous souhaitez peut-être interdire temporairement les emplacements sur un lieu donné ; par exemple, un site peut rencontrer des problèmes transitoires. Comme vous pouvez mettre à jour la configuration des filtres d'une file d'attente à tout moment, vous pouvez facilement ajouter et supprimer des emplacements selon vos besoins. Ajoutez une configuration de filtre en appelant [UpdateGameSessionQueue](#).

- Utilisez des instructions spéciales pour les demandes de placement individuelles :
 - Incluez une liste de priorité prioritaire dans une demande de placement de session de jeu. Vous pouvez fournir une autre liste de sites prioritaires avec n'importe quelle [StartGameSessionPlacement](#) demande. Cette liste remplace efficacement la hiérarchisation configurée de la file d'attente pour les emplacements pour cette seule demande. Cela n'a aucun impact sur les autres demandes. Cette fonctionnalité de dérogation comporte quelques exigences :
 - Utilisez une liste de remplacement uniquement avec une file d'attente dont la configuration de priorité est en place LOCATION comme première priorité.
 - N'incluez pas les données de latence des joueurs dans la même demande de placement. L'inclusion des données de latence crée des conflits lors de la priorisation des emplacements qui ne Amazon GameLift Servers peuvent pas être résolus.
 - Décidez de la manière dont vous Amazon GameLift Servers souhaitez procéder s'il ne trouve aucune ressource disponible dans la liste de priorité prioritaire. Choisissez entre revenir aux autres emplacements de la file d'attente ou limiter les emplacements à la liste de remplacement. Par défaut, Amazon GameLift Servers revient en arrière pour tenter de le placer sur les autres emplacements de la file d'attente.
 - Mettez à jour la configuration du filtre de la file d'attente selon les besoins, par exemple en ajoutant des emplacements sur la liste de remplacement. La liste de remplacement n'invalide pas la liste de filtres.

Créez une politique de latence pour les joueurs

Si vos demandes de placement incluent des données de latence des joueurs, Amazon GameLift Servers trouve les sessions de jeu aux endroits où la latence moyenne est la plus faible pour tous les joueurs. Le fait de placer les sessions de jeu en fonction de la latence moyenne des joueurs Amazon GameLift Servers empêche de placer la plupart des joueurs dans des jeux à latence élevée.

Cependant, cela place Amazon GameLift Servers toujours les joueurs avec une latence extrême. Pour répondre aux besoins de ces joueurs, créez des politiques de latence des joueurs.

Une politique de latence des joueurs Amazon GameLift Servers empêche de placer une session de jeu demandée à un endroit où les joueurs concernés par la demande subiraient une latence supérieure à la valeur maximale. Les politiques de latence des joueurs peuvent également Amazon GameLift Servers empêcher de faire correspondre les demandes de session de jeu aux joueurs à latence plus élevée.

Tip

Pour gérer les règles spécifiques à la latence, telles que l'exigence d'une latence similaire pour tous les joueurs d'un groupe, vous pouvez [Amazon GameLift ServersFlexMatch](#) créer des règles de matchmaking basées sur la latence.

Par exemple, considérez cette file d'attente avec un délai d'attente de 5 minutes et les politiques de latence des joueurs suivantes :

1. Passez 120 secondes à rechercher un endroit où toutes les latences des joueurs sont inférieures à 50 millisecondes.
2. Passez 120 secondes à rechercher un endroit où toutes les latences des joueurs sont inférieures à 100 millisecondes.
3. Passez le temps d'attente restant jusqu'à la fin du délai imparti à rechercher un endroit où toutes les latences des joueurs sont inférieures à 200 millisecondes.

Évaluer les métriques relatives aux files

Utilisez les métriques pour évaluer les performances de vos files d'attente. Vous pouvez consulter les statistiques relatives aux files d'attente dans le [Amazon GameLift Servers console](#) ou sur Amazon CloudWatch. Pour obtenir une liste et une description des métriques de file d'attente, consultez [Métriques Amazon GameLift Servers pour les files d'attente](#).

Les métriques de file d'attente peuvent fournir des informations sur les points suivants :

- Performances globales des files d'attente : les métriques des files d'attente indiquent dans quelle mesure une file d'attente répond aux demandes de placement. Ces indicateurs peuvent également

vous aider à identifier quand et pourquoi les placements échouent. Pour les files d'attente dont les flottes sont redimensionnées manuellement, les QueueDepth indicateurs AverageWaitTime et peuvent indiquer quand vous devez ajuster la capacité d'une file d'attente.

- **FleetIQ performance de l'algorithme** — Pour les demandes de placement utilisant le FleetIQ algorithme, les métriques indiquent à quelle fréquence l'algorithme trouve le placement idéal des sessions de jeu. Le placement peut donner la priorité à l'utilisation des ressources présentant la latence la plus faible pour les joueurs ou des ressources les moins coûteuses. Il existe également des mesures d'erreur qui identifient les raisons courantes pour lesquelles Amazon GameLift Servers ne n'arrive pas à trouver le placement idéal. Pour plus d'informations sur les métriques, consultez la section [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#).
- **Emplacements spécifiques à un lieu** : pour les files d'attente comportant plusieurs sites, les statistiques indiquent les placements réussis par site. Pour les files d'attente utilisant le FleetIQ algorithme, ces données fournissent un aperçu utile de l'endroit où se produit l'activité des joueurs.

Lors de l'évaluation des métriques pour FleetIQ performances de l'algorithme, tenez compte des conseils suivants :

- Pour suivre le taux de recherche du placement idéal par la file d'attente, utilisez la PlacementsSucceeded métrique en combinaison avec FleetIQ métriques pour la latence la plus faible et le prix le plus bas.
- Pour accélérer le taux de recherche du placement idéal par une file d'attente, passez en revue les indicateurs d'erreur suivants :
 - S'il FirstChoiceOutOfCapacity est élevé, ajustez l'échelle de capacité pour les flottes de la file d'attente.
 - Si la métrique FirstChoiceNotViable d'erreur est élevée, examinez vos flottes d'instances Spot. Les flottes d'instances ponctuelles sont considérées comme non viables lorsque le taux d'interruption pour un type d'instance particulier est trop élevé. Pour résoudre ce problème, modifiez la file d'attente afin d'utiliser des flottes d'instances Spot avec différents types d'instances. Nous vous recommandons d'inclure des flottes d'instances Spot avec différents types d'instances dans chaque emplacement.

Concevoir une file d'attente pour les instances Spot

Vous pouvez réaliser d'importantes économies sur les coûts d'hébergement en utilisant des flottes Spot. Pour en savoir plus, consultez [Instances à la demande et instances ponctuelles](#). Pour ajouter

des flottes Spot à votre solution d'hébergement, vous devez configurer une file d'attente de sessions de jeu combinant des flottes Spot et des flottes à la demande. Amazon GameLift Servers utilise une file d'attente pendant le processus de placement des sessions de jeu pour effectuer des recherches dans plusieurs flottes et trouver les meilleurs hôtes disponibles pour les nouvelles sessions de jeu. Cette rubrique fournit des conseils sur la façon de commencer à utiliser les flottes Spot.

Utilisez-vous FlexMatch pour le matchmaking ? Vous pouvez suivre les étapes suivantes pour ajouter des flottes de Spot à vos files d'attente de session de jeu existantes pour les emplacements de matchmaking.

1. Déterminez les destinations de votre file d'attente de session de jeu.

La gestion du placement des sessions de jeu par le biais d'une file d'attente est une bonne pratique, et elle est obligatoire lors de l'utilisation d'instances Spot. Étant donné que les instances Spot ne sont pas toujours disponibles lorsque vous en avez besoin, vous devez concevoir une file d'attente résiliente comprenant à la fois des flottes Spot et des flottes à la demande afin d'offrir une capacité de sauvegarde. Vous pouvez réduire la taille de vos flottes à la demande jusqu'à ce qu'elles soient nécessaires. Pour concevoir votre file d'attente, tenez compte des points suivants :

- **Emplacements** — Si possible, vos flottes Spot et vos flottes à la demande doivent se trouver dans la même région que les joueurs. Positionnez les ressources ponctuelles et les ressources à la demande dans chaque site que vous souhaitez prendre en charge. Les flottes multisites prennent en charge les instances ponctuelles et à la demande.
- **Types d'instances** — Tenez compte de la configuration matérielle requise par votre serveur de jeu et de la disponibilité des instances dans les emplacements que vous choisissez.

Pour essayer une file d'attente qui optimise la disponibilité et la résilience de Spot, consultez [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#) Pour connaître les meilleures pratiques en matière de conception de Spot, voir [Bonnes pratiques en matière de files d'attente pour les sessions de Amazon GameLift Servers jeu](#).

2. Créez les flottes pour votre file d'attente optimisée pour les instances Spot.

En fonction de la conception de votre file d'attente, créez des flottes pour déployer vos serveurs de jeu aux emplacements et aux types d'instances souhaités. Consultez [Création d'une EC2 flotte Amazon GameLift Servers gérée](#) pour plus de détails sur la création et la configuration de nouvelles flottes.

3. Créez votre file d'attente de session de jeu.

Ajoutez les destinations de la flotte, configurez le processus de placement des sessions de jeu et définissez les priorités de placement. Consultez [Création d'une file d'attente de session de jeu](#) pour plus de détails sur la création et la configuration de la nouvelle file d'attente.

4. Mettez à jour le service client de votre jeu pour utiliser la file d'attente.

Lorsque votre client de jeu utilise une file d'attente pour demander des ressources, celle-ci évite les ressources présentant un risque élevé d'interruption et sélectionne l'emplacement qui correspond à vos priorités définies. Pour obtenir de l'aide sur la mise en œuvre des placements de sessions de jeu dans votre client de jeu, consultez [Créez des sessions de jeu](#).

5. Mettez à jour votre serveur de jeu pour qu'il puisse gérer une interruption ponctuelle.

AWS peut interrompre les instances Spot avec une notification de 2 minutes, lorsqu'elles ont besoin de récupérer leur capacité. Configurez votre serveur de jeu pour gérer les interruptions afin de minimiser l'impact sur les joueurs.

Avant AWS de récupérer une instance Spot, celle-ci envoie une notification de résiliation. Amazon GameLift Servers transmet la notification à tous les processus du serveur concernés en invoquant le Amazon GameLift Servers Fonction de rappel du SDK du serveur. `onProcessTerminate()` Implémentez ce rappel pour mettre fin à la session de jeu ou déplacer la session de jeu et les joueurs vers une nouvelle instance. Consultez [Répondre à une notification d'arrêt du processus du serveur](#) pour plus de détails sur l'implémentation de `onProcessTerminate()`.

Note

AWS met tout en œuvre pour fournir la notification avant de récupérer une instance, mais il est possible qu' AWS elle récupère l'instance Spot avant l'arrivée de l'avertissement. Préparez votre serveur de jeu à faire face aux interruptions inattendues.

6. Vérifiez les performances de vos flottes et files d'attente Spot.

Vue Amazon GameLift Servers métriques dans le Amazon GameLift Servers console ou avec Amazon CloudWatch pour évaluer les performances. Pour plus d'informations sur Amazon GameLift Servers métriques, voir [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#). Les métriques clés incluent :

- Taux d'interruption : utilisez les `GameSessionInterruptions` indicateurs `InstanceInterruptions` et pour suivre le nombre et la fréquence des interruptions liées à Spot pour les instances et les sessions de jeu. Les sessions de jeu récupérées par AWS ont un statut `TERMINATED` et une raison de statut de `INTERRUPTED`.
- Efficacité des files d'attente : suivez les taux de réussite du placement, le temps d'attente moyen et la profondeur des files d'attente pour vous assurer que les flottes Spot n'ont aucun impact sur les performances de votre file d'attente.
- Utilisation de la flotte : surveillez les données relatives aux instances, aux sessions de jeu et aux sessions des joueurs. L'utilisation de vos flottes à la demande peut indiquer que les files d'attente évitent de se placer dans vos flottes Spot afin d'éviter toute interruption.

Bonnes pratiques pour les files d'attente avec les flottes Spot

Si votre file d'attente inclut des flottes Spot, configurez une file d'attente résiliente. Cela permet de tirer parti des économies réalisées grâce aux flottes Spot tout en minimisant l'effet des interruptions de session de jeu. Pour obtenir de l'aide sur la création correcte de flottes et de files d'attente pour les sessions de jeu à utiliser avec les flottes Spot, consultez [Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot](#). Pour plus d'informations sur les instances Spot, consultez [Concevoir une file d'attente pour les instances Spot](#).

Outre les meilleures pratiques générales décrites dans la section précédente, prenez en compte les meilleures pratiques spécifiques à SPOT :

- Créez au moins une flotte à la demande dans chaque site. Les flottes à la demande fournissent des serveurs de jeu de sauvegarde pour vos joueurs. Vous pouvez réduire la taille de vos flottes de sauvegarde jusqu'à ce qu'elles soient nécessaires et utiliser le dimensionnement automatique pour augmenter la capacité à la demande lorsque les flottes Spot ne sont pas disponibles.
- Sélectionnez différents types d'instances dans plusieurs flottes Spot d'un même emplacement. Si un type d'instance ponctuelle devient temporairement indisponible, l'interruption n'affecte qu'un seul parc d'instances ponctuelles sur le site. La meilleure pratique consiste à choisir des types d'instances largement disponibles et à utiliser des types d'instance appartenant à la même famille (par exemple, `m5.large`, `m5.xlarge`, `m5.2xlarge`). Utilisez la [Amazon GameLift Servers console](#) pour consulter les données de tarification historiques pour les types d'instances.

Configurer la notification des événements pour le placement des sessions de jeu

Vous pouvez utiliser les notifications d'événements pour suivre l'état des demandes de placement individuelles. Nous vous recommandons de configurer des notifications d'événements pour tous les jeux présentant un volume élevé d'activités de placement.

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

- Amazon GameLift Servers Publiez des notifications d'événements sur une rubrique Amazon Simple Notification Service (Amazon SNS) à l'aide d'une file d'attente.
- Utilisez les EventBridge événements Amazon publiés automatiquement et sa suite d'outils pour gérer les événements.

Pour une liste des événements de placement de session de jeu émis par Amazon GameLift Servers, consultez [Événements de placement lors des sessions de jeu](#).

Important

Pour les systèmes de placement de gros volumes, nous recommandons d'utiliser des rubriques Amazon SNS standard (non FIFO) plutôt que des rubriques FIFO. Les rubriques FIFO ont des limites de publication inférieures à celles des rubriques standard, ce qui peut entraîner des exceptions de limitation en cas de charge élevée. Si vous rencontrez des difficultés avec les sujets FIFO, vous risquez de perdre les notifications de placement dans les files d'attente.

Configurer une rubrique SNS

Amazon GameLift Servers Pour publier tous les événements générés par une file d'attente de session de jeu dans un sujet, définissez le champ cible de notification sur un sujet.

Pour configurer une rubrique SNS pour une notification d'événement Amazon GameLift Servers

1. [Connectez-vous à la console Amazon SNS AWS Management Console et ouvrez-la sur v3/home. https://console.aws.amazon.com/sns/](https://console.aws.amazon.com/sns/)

2. Sur la page Sujets SNS, choisissez Créer un sujet et suivez les instructions pour créer votre sujet.
3. Sous Politique d'accès, procédez comme suit :
 - a. Choisissez la méthode avancée.
 - b. Ajoutez la section en gras suivante de l'objet JSON à la politique existante.

JSON

```
{
  "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:us-east-1:111122223333: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:us-east-1:111122223333:your_topic_name",
"Condition": {
  "ArnLike": {
    "aws:SourceArn": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/your_queue_name"
  }
}
}
```

- c. (Facultatif) Ajoutez un contrôle d'accès supplémentaire à la rubrique en ajoutant des conditions à la politique de ressources.
4. Choisissez Créer une rubrique.
5. Après avoir créé votre rubrique SNS, ajoutez-la aux files d'attente lors de la création de la file d'attente ou modifiez une file d'attente existante pour l'ajouter.

Configuration d'une rubrique SNS avec chiffrement côté serveur

Le chiffrement côté serveur (SSE) vous permet de stocker des données sensibles dans des rubriques chiffrées. Le chiffrement SSE protège le contenu des messages présents dans les rubriques Amazon SNS à l'aide de clés gérées dans AWS Key Management Service (AWS KMS). Pour plus d'informations sur le chiffrement côté serveur avec Amazon SNS, [consultez la section Encryption at rest](#) du manuel Amazon Simple Notification Service Developer Guide.

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

- [Création d'une clé](#) dans le guide du AWS Key Management Service développeur
- [Activation de SSE pour un sujet](#) du manuel Amazon Simple Notification Service Developer Guide

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

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "gamelift.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
```

```

    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn":
"arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
    },
    "StringEquals": {
      "kms:EncryptionContext:aws:sns:topicArn":
"arn:aws:sns:your_region:your_account:your_sns_topic_name"
    }
  }
}

```

Configurez EventBridge

Amazon GameLift Servers publie automatiquement tous les événements de placement des sessions de jeu sur EventBridge. Vous pouvez définir des règles pour que les événements soient acheminés vers des cibles pour être traités. Par exemple, vous pouvez définir une règle pour acheminer l'événement `PlacementFulfilled` vers une AWS Lambda fonction qui gère les tâches qui précèdent la connexion à une session de jeu. Pour plus d'informations EventBridge, consultez [Qu'est-ce qu'Amazon EventBridge ?](#) dans le guide de EventBridge l'utilisateur Amazon.

Voici quelques exemples de EventBridge règles à utiliser avec les Amazon GameLift Servers files d'attente :

Correspond aux événements de toutes les Amazon GameLift Servers files d'attente

```

{
  "source": [
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ]
}

```

Correspond aux événements d'une file d'attente spécifique

```

{
  "source": [

```



```
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ],
  "resources": [
    "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
  ]
}
```

Tutoriel : Création d'une Amazon GameLift Servers file d'attente avec des instances Spot

Introduction

Ce didacticiel explique comment configurer le placement des sessions de jeu pour les jeux déployés sur des flottes Spot à faible coût. Les flottes de machines à sous nécessitent des étapes supplémentaires afin de garantir la disponibilité continue des serveurs de jeu pour vos joueurs.

Public visé

Ce didacticiel est destiné aux développeurs de jeux qui souhaitent utiliser des flottes Spot pour héberger des serveurs de jeux personnalisés ou Amazon GameLift ServersRealtime.

Ce que vous allez apprendre

- Définissez le groupe de joueurs desservi par votre file d'attente de session de jeu.
- Construisez une infrastructure de flotte adaptée à l'étendue de la file d'attente des sessions de jeu.
- Attribuez un alias à chaque flotte pour en déduire l'identifiant de la flotte.
- Créez une file d'attente, ajoutez des flottes et hiérarchisez les Amazon GameLift Servers emplacements des sessions de jeu.
- Ajoutez des politiques de latence pour les joueurs afin de minimiser les problèmes de latence.

Prérequis

Avant de créer des flottes et des files d'attente pour le placement des sessions de jeu, effectuez les tâches suivantes :

- Consultez [Fonctionnement d'Amazon GameLift Servers](#).
- [Intégrez votre serveur de jeu à Amazon GameLift Servers](#).

- [Téléchargez le build ou le Realtime script de votre serveur de jeu](#) sur Amazon GameLift Servers.
- [Planifiez la configuration de votre flotte.](#)

Étape 1 : Définissez l'étendue de votre file d'attente

Dans ce didacticiel, nous concevons une file d'attente pour un jeu comportant une variante de build de serveur de jeu. Au moment de sa sortie, le jeu sera disponible sur deux sites : Asie-Pacifique (Séoul) et Asie-Pacifique (Singapour). Comme ces emplacements sont proches les uns des autres, la latence n'est pas un problème pour nos joueurs.

Dans cet exemple, il y a un segment de joueur, ce qui signifie que nous créons une file d'attente. À l'avenir, lorsque nous sortirons le jeu en Amérique du Nord, nous pourrions créer une deuxième file d'attente réservée aux joueurs nord-américains.

Pour de plus amples informations, veuillez consulter [Définition de l'étendue d'une file d'attente](#).

Étape 2 : Création d'une infrastructure de flotte Spot

Créez des flottes dans des lieux et à l'aide de versions ou de scripts de serveurs de jeu adaptés au champ d'application que vous avez défini dans [Étape 1 : Définissez l'étendue de votre file d'attente](#).

Dans ce didacticiel, nous créons une infrastructure à deux sites avec au moins une flotte Spot et une flotte à la demande dans chaque site. Chaque flotte déploie le même build de serveur de jeu. En outre, nous prévoyons que le trafic de joueurs sera plus important sur le site de Séoul, c'est pourquoi nous y ajoutons d'autres flottes de Spot.

Le schéma suivant montre l'exemple d'infrastructure de flotte Spot, avec 3 flottes sur le site ap-northeast-2 (Séoul) et 2 flottes sur le site ap-southeast-1 (Singapour). Toutes les instances des deux flottes utilisent le build MBG_PROD_v1. La flotte d'ap-northeast-2 contient les configurations de flotte suivantes : fleet 1234_spot_1 avec un type d'instance c5.large, fleet 1234_spot_2 avec un type d'instance c5.xlarge et fleet 1234_ondemand avec un type d'instance c5.large. La flotte d'ap-southeast-1 contient les configurations de flotte suivantes : fleet 1234_spot_1 avec un type d'instance c5.large et fleet 1234_ondemand avec un type d'instance c5.large.

Étape 3 : Attribuer des alias à chaque flotte

Créez un nouvel alias pour chaque flotte de votre infrastructure. Les alias font abstraction des identités des flottes, ce qui rend le remplacement périodique des flottes efficace. Pour plus d'informations sur la création d'alias, consultez [Créez un Amazon GameLift Servers alias](#).

Notre infrastructure de flotte compte cinq flottes, c'est pourquoi nous créons cinq alias en utilisant la stratégie de routage. Nous avons besoin de trois alias pour le site Asie-Pacifique (Séoul) et de deux alias pour le site Asie-Pacifique (Singapour).

Le schéma suivant montre l'infrastructure de flotte Spot décrite à la deuxième étape avec des alias ajoutés à chaque flotte. La flotte 1234_spot_1 porte l'alias MBG_spot_1, la flotte 1234_spot_2 porte l'alias MBG_spot_2 et la flotte 1234_ondemand porte l'alias MBG_OnDemand.

Pour de plus amples informations, veuillez consulter [Créez une file d'attente multi-sites](#).

Étape 4 : Création d'une file d'attente avec des destinations

Créez la file d'attente des sessions de jeu et ajoutez les destinations de votre flotte. Pour plus d'informations sur la création d'une file d'attente, consultez [Création d'une file d'attente de session de jeu](#).

Lors de la création de votre file d'attente :

- Définissez le délai d'expiration par défaut sur 10 minutes. Plus tard, vous pourrez tester l'impact du délai d'attente sur les temps d'attente de vos joueurs pour accéder aux parties.
- Pour le moment, ignorez la section sur les politiques de latence des joueurs. Nous aborderons ce point à l'étape suivante.
- Classez par ordre de priorité les flottes de votre file d'attente. Lorsque vous travaillez avec des flottes Spot, nous recommandons l'une des approches suivantes :
 - Si votre infrastructure utilise un site principal avec des flottes sur un deuxième site pour la sauvegarde, hiérarchisez les flottes d'abord par emplacement, puis par type de flotte.
 - Si votre infrastructure utilise plusieurs sites de la même manière, hiérarchisez les flottes par type de flotte, en plaçant les flottes Spot en haut de la file d'attente.

Pour ce didacticiel, nous créons une nouvelle file d'attente portant **MBG_spot_queue** le nom et ajoutons les alias de nos cinq flottes. Nous hiérarchisons ensuite les emplacements d'abord par emplacement et ensuite par type de flotte.

Sur la base de cette configuration, cette file d'attente tente toujours de placer de nouvelles sessions de jeu dans une flotte de Spot à Séoul. Lorsque ces flottes sont pleines, la file d'attente utilise la capacité disponible de la flotte Seoul On-Demand comme solution de secours. Si les trois flottes de Séoul ne sont pas disponibles, Amazon GameLift Servers place des sessions de jeu sur les flottes de Singapour.

Le schéma suivant montre une file d'attente dont le délai d'expiration est de 300 secondes et les destinations prioritaires. Les destinations sont classées dans l'ordre suivant : 1234_spot_1 dans ap-northeast-2, 1234_spot_2 dans ap-northeast-2, 1234_ondemand dans ap-northeast-2, 1234_spot_1 dans ap-southeast-1, et 1234_ondemand dans ap-southeast-1.

Étape 5 : ajouter des limites de latence à la file d'attente

Notre jeu inclut des informations de latence dans les demandes de placement de session de jeu. Nous avons également une fonctionnalité de groupe de joueurs qui crée une session de jeu pour un groupe de joueurs. Nous pouvons faire attendre les joueurs un peu plus longtemps pour accéder à des jeux offrant une expérience de jeu idéale. Nos tests de jeu montrent les observations suivantes :

- Une latence inférieure à 50 millisecondes est idéale.
- Le jeu est injouable à des latences supérieures à 250 millisecondes.
- Les joueurs s'impatientent au bout d'une minute environ.

Pour notre file d'attente, avec un délai d'attente de 300 secondes, nous ajoutons des déclarations de politique limitant la latence autorisée. Les déclarations de politique autorisent progressivement des valeurs de latence plus élevées allant jusqu'à 250 millisecondes de latence.

Grâce à cette politique, notre file d'attente recherche les emplacements présentant une latence idéale (inférieure à 50 millisecondes) pendant la première minute, puis assouplit la limite. La file d'attente ne permet pas d'effectuer des placements où la latence des joueurs est de 250 millisecondes ou plus.

Le schéma suivant montre la file d'attente à partir de l'étape 4 avec les politiques de latence des joueurs ajoutées. Les politiques de latence des joueurs stipulent, appliquent la limite de 50 ms pendant 60 secondes, appliquent la limite de 125 ms pendant 30 secondes et appliquent la limite de 250 ms jusqu'à l'expiration du délai imparti.

Récapitulatif

Félicitations ! Voici ce que vous avez accompli :

- Vous avez une file d'attente de sessions de jeu limitée à un segment de votre population de joueurs.
- Votre file d'attente utilise efficacement les flottes Spot et est résiliente en cas d'interruption de Spot.
- Votre file d'attente donne la priorité aux flottes pour une expérience de jeu optimale.
- La file d'attente comporte des limites de latence pour protéger les joueurs des mauvaises expériences de jeu.

Vous pouvez désormais utiliser la file d'attente pour placer des sessions de jeu pour les joueurs qu'elle dessert. Lorsque vous faites des demandes de placement de session de jeu pour ces joueurs, faites référence au nom de cette file d'attente de session de jeu dans la demande. Pour plus d'informations sur les demandes de placement de session de jeu [Créez des sessions de jeu](#), consultez ou [Intégration d'un client de jeu pour Amazon GameLift ServersRealtime](#).

Étapes suivantes :

- [Concevez votre propre file d'attente.](#)
- [Créez une file d'attente.](#)
- [Utilisez une file d'attente avec votre client de jeu.](#)

Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers

La capacité d'hébergement, mesurée en instances, représente le nombre de sessions de jeu Amazon GameLift Servers pouvant être hébergées simultanément et le nombre de joueurs simultanés que ces sessions de jeu peuvent accueillir. L'une des tâches les plus difficiles de l'hébergement de jeux consiste à augmenter la capacité pour répondre à la demande des joueurs sans gaspiller de l'argent en ressources dont vous n'avez pas besoin. Pour de plus amples informations, veuillez consulter [Dimensionnement de la capacité d'une flotte](#).

La capacité est ajustée au niveau de l'emplacement de la flotte. Toutes les flottes ont au moins un emplacement : la AWS région d'origine de la flotte. Lorsque vous consultez ou augmentez la capacité, les informations sont répertoriées par emplacement, y compris la région d'origine de la flotte et tout autre site distant.

Vous pouvez définir manuellement le nombre d'instances à maintenir ou configurer le dimensionnement automatique pour ajuster dynamiquement la capacité en fonction de l'évolution de la demande des joueurs. Nous vous recommandons de commencer par activer l'option de mise à l'échelle automatique basée sur les cibles. L'objectif de la mise à l'échelle automatique basée sur les cibles est de maintenir suffisamment de ressources d'hébergement pour accueillir les joueurs actuels, ainsi qu'un petit supplément pour faire face à des pics inattendus de demande de joueurs. Pour la plupart des jeux, la mise à l'échelle automatique basée sur les cibles constitue une solution de mise à l'échelle très efficace.

Vous pouvez effectuer la plupart des activités de dimensionnement de flotte à l'aide de la Amazon GameLift Servers console. Vous pouvez également utiliser un AWS SDK ou le AWS Command Line Interface (AWS CLI) avec l'[API de service pour Amazon GameLift Servers](#).

Rubriques

- [Pour gérer la capacité du parc dans la console](#)
- [Définissez des limites Amazon GameLift Servers de capacité](#)
- [Définissez manuellement la capacité d'une Amazon GameLift Servers flotte](#)
- [Augmentez automatiquement la capacité de votre flotte avec Amazon GameLift Servers](#)
- [Échelle des flottes de Amazon GameLift Servers conteneurs](#)

Pour gérer la capacité du parc dans la console

1. Ouvrez la [Amazon GameLift Servers console](#).
2. Dans le volet de navigation, choisissez Hosting, Fleets.
3. Sur la page Flottes, choisissez le nom d'une flotte active pour ouvrir la page détaillée de la flotte.
4. Choisissez l'onglet Mise à l'échelle. Sur cet onglet, vous pouvez :
 - Consultez les indicateurs de dimensionnement historiques pour l'ensemble de la flotte.
 - Consultez et mettez à jour les paramètres de capacité pour chaque emplacement du parc, y compris les limites d'échelle et les paramètres de capacité actuels.
 - Mettez à jour le dimensionnement automatique basé sur les cibles, consultez les politiques de dimensionnement automatique basées sur des règles appliquées à l'ensemble du parc et suspendez les activités de dimensionnement automatique pour chaque site.

Définissez des limites Amazon GameLift Servers de capacité

Lorsque vous augmentez la capacité d'hébergement d'un emplacement de Amazon GameLift Servers flotte, manuellement ou automatiquement, tenez compte des limites d'échelle du site. Tous les sites du parc ont une limite minimale et maximale qui définissent la plage autorisée pour la capacité du site. Par défaut, les limites relatives à l'emplacement du parc sont limitées à 0 instances au minimum et à 1 instance au maximum. Avant de pouvoir redimensionner l'emplacement d'une flotte, ajustez les limites.

Si vous utilisez la mise à l'échelle automatique, la limite maximale permet d'Amazon GameLift Servers étendre l'emplacement d'une flotte pour répondre à la demande des joueurs, tout en évitant les coûts d'hébergement exorbitants, par exemple lors d'une attaque DDOS. Configurez une [CloudWatch alarme Amazon](#) pour vous avertir lorsque la capacité approche de la limite maximale, afin que vous puissiez évaluer la situation et ajuster manuellement le cas échéant. (Vous pouvez également [créer une alarme de facturation](#) pour surveiller AWS les coûts.) La limite minimale est utile pour maintenir la disponibilité de l'hébergement, même lorsque la demande des joueurs est faible.

Vous pouvez définir des limites de capacité pour les emplacements d'une flotte dans la [Amazon GameLift Servers console](#) ou en utilisant le AWS Command Line Interface (AWS CLI).

Pour définir les limites de capacité

Console

1. Ouvrez la [Amazon GameLift Servers console](#).
2. Dans le volet de navigation, choisissez Hosting, Fleets.
3. Sur la page Flottes, choisissez le nom d'une flotte active pour ouvrir la page détaillée de la flotte.
4. Dans l'onglet Scaling, sous Scaling capacity, sélectionnez un emplacement de flotte, puis choisissez Modifier.
5. Dans la boîte de dialogue Modifier la capacité de dimensionnement, définissez le nombre d'instances pour la taille minimale, les instances souhaitées et la taille maximale.
6. Choisissez Confirmer.

AWS CLI

1. Vérifiez les paramètres de capacité actuels. Dans une fenêtre de ligne de commande, utilisez la [describe-fleet-location-capacity](#) commande avec l'ID de flotte et l'emplacement pour lesquels vous souhaitez modifier la capacité. Cette commande renvoie un [FleetCapacity](#) objet qui inclut les paramètres de capacité actuels de l'emplacement. Déterminez si les nouvelles limites d'instances peuvent s'adapter au paramètre d'instances actuellement souhaité.

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
  --location <location name>
```

2. Mettez à jour les paramètres de limite. Dans une fenêtre de ligne de commande, utilisez la [update-fleet-capacity](#) commande avec les paramètres suivants. Vous pouvez ajuster les deux limites d'instance et le nombre d'instances souhaitées avec la même commande.

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
--desired-instances <fleet capacity goal>
```

Exemple :


```
aws gamelift update-fleet-capacity \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --location us-west-2 \  
  --max-size 10 \  
  --min-size 1 \  
  --desired-instances 10
```

Si votre demande est acceptée, Amazon GameLift Servers renvoie le numéro de flotte. Si la nouvelle min-size valeur max-size ou la nouvelle valeur entre en conflit avec le desired-instances paramètre actuel, Amazon GameLift Servers renvoie une erreur.

Définissez manuellement la capacité d'une Amazon GameLift Servers flotte

Lorsque vous créez un nouveau parc, définissez Amazon GameLift Servers automatiquement les instances souhaitées sur une instance dans chaque emplacement du parc. Amazon GameLift ServersDéploie ensuite une nouvelle instance dans chaque emplacement. Pour modifier la capacité du parc, vous pouvez ajouter une politique de dimensionnement automatique basée sur des cibles, ou vous pouvez définir manuellement le nombre d'instances que vous souhaitez pour un emplacement. Pour de plus amples informations, veuillez consulter [Dimensionnement de la capacité d'une flotte](#).

La définition manuelle de la capacité d'une flotte peut être utile lorsque vous n'avez pas besoin d'une mise à l'échelle automatique ou lorsque vous devez maintenir la capacité à un niveau spécifié. La définition manuelle de la capacité ne fonctionne que si vous n'utilisez pas de politique de dimensionnement automatique basée sur des cibles. Si vous avez une politique de dimensionnement automatique basée sur des cibles, elle réinitialise immédiatement la capacité souhaitée en fonction de ses propres règles de dimensionnement.

Vous pouvez définir la capacité manuellement dans la Amazon GameLift Servers console ou en utilisant le AWS Command Line Interface (AWS CLI). Le statut de la flotte doit être actif.

Suspendre le dimensionnement automatique

Vous pouvez suspendre toutes les activités de mise à l'échelle automatique pour chaque emplacement du parc. Lorsque le dimensionnement automatique est suspendu, le nombre

d'instances souhaité dans l'emplacement du parc reste le même, sauf modification manuelle. Lorsque vous suspendez le dimensionnement automatique pour un site, cela affecte les politiques actuelles de la flotte et les politiques que vous pourriez définir à l'avenir.

Pour définir manuellement la capacité de la flotte

Console

1. Ouvrez la [Amazon GameLift Servers console](#).
2. Dans le volet de navigation, choisissez Hosting, Fleets.
3. Sur la page Flottes, choisissez le nom d'une flotte active pour ouvrir la page détaillée de la flotte.
4. Dans l'onglet Mise à l'échelle, sous Emplacements de mise à l'échelle automatique suspendus, sélectionnez chaque emplacement pour lequel vous souhaitez suspendre le dimensionnement automatique, puis choisissez Suspendre.
5. Sous Capacité de mise à l'échelle, sélectionnez un emplacement que vous souhaitez définir manuellement, puis choisissez Modifier.
6. Dans la boîte de dialogue Modifier la capacité de dimensionnement, définissez votre valeur préférée pour les instances souhaitées, puis choisissez Confirmer. Cela indique Amazon GameLift Servers le nombre d'instances à maintenir actives, prêtes à héberger des sessions de jeu.

Amazon GameLift Servers répond aux modifications en déployant des instances supplémentaires ou en fermant celles qui ne sont pas nécessaires. À la Amazon GameLift Servers fin de ce processus, le nombre d'instances actives dans l'emplacement change pour correspondre à la valeur d'instances souhaitée mise à jour. Ce processus peut prendre un certain temps.

AWS CLI

1. Vérifiez les paramètres de capacité actuels. Dans une fenêtre de ligne de commande, utilisez la [describe-fleet-location-capacity](#) commande avec l'ID de flotte et l'emplacement pour lesquels vous souhaitez modifier la capacité. Cette commande renvoie un [FleetCapacity](#) objet qui inclut les paramètres de capacité actuels de l'emplacement. Déterminez si les limites d'instances peuvent s'adapter au nouveau paramètre d'instances souhaité.

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
```

```
--location <location name>
```

2. Mettez à jour la capacité souhaitée. Utilisez la [update-fleet-capacity](#) commande avec l'ID de flotte, l'emplacement et une nouvelle valeur pour les instances souhaitées. Si cette valeur se situe en dehors de la plage limite actuelle, vous pouvez ajuster les valeurs limites dans la même commande.

```
--fleet-id <fleet identifier>
--location <location name>
--desired-instances <fleet capacity as an integer>
--max-size <maximum capacity> [Optional]
--min-size <minimum capacity> [Optional]
```

Exemple :

```
aws gamelift update-fleet-capacity \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --location us-west-2 \
  --desired-instances 5 \
  --max-size 10 \
  --min-size 1
```

Si votre demande est acceptée, Amazon GameLift Servers renvoie le numéro de flotte. Si le nouveau paramètre d'instances souhaité dépasse les limites minimale et maximale, Amazon GameLift Servers renvoie une erreur.

Augmentez automatiquement la capacité de votre flotte avec Amazon GameLift Servers

Utiliser l'auto-scaling dans Amazon GameLift Servers pour augmenter dynamiquement la capacité de votre flotte en fonction de l'activité du serveur de jeu. Au fur et à mesure que les joueurs arrivent et commencent des sessions de jeu, le scalage automatique peut ajouter de nouvelles instances ; à mesure que la demande des joueurs diminue, le scalage automatique peut mettre fin aux instances inutiles. La mise à l'échelle automatique est un moyen efficace de minimiser vos ressources et vos coûts d'hébergement, tout en offrant une expérience de jeu fluide et rapide.

Pour utiliser le dimensionnement automatique, vous devez créer des politiques de dimensionnement qui indiquent Amazon GameLift Servers quand augmenter ou diminuer. Il existe deux types de politiques de dimensionnement : basées sur des cibles et basées sur des règles. L'approche basée sur les cibles (suivi des cibles) est une solution complète. Nous le recommandons comme l'option la plus simple et la plus efficace. Les politiques de dimensionnement basées sur des règles vous obligent à définir chaque aspect du processus décisionnel relatif au dimensionnement automatique, ce qui est utile pour résoudre des problèmes spécifiques. Cette solution fonctionne parfaitement en complément de la mise à l'échelle automatique basée sur les cibles.

Vous pouvez gérer le dimensionnement automatique basé sur les cibles à l'aide du Amazon GameLift Servers console, the AWS Command Line Interface (AWS CLI) ou un AWS SDK. Vous pouvez gérer le dimensionnement automatique basé sur des règles à l'aide du AWS CLI ou d'un AWS SDK uniquement, mais vous pouvez consulter les politiques de dimensionnement basées sur des règles dans la console.

Rubriques

- [Mise à l'échelle automatique basée sur les cibles](#)
- [Mise à l'échelle automatique avec des politiques basées sur des règles](#)

Mise à l'échelle automatique basée sur les cibles

Mise à l'échelle automatique basée sur les cibles pour Amazon GameLift Servers ajuste les niveaux de capacité en fonction de la métrique `PercentAvailableGameSessions` du parc. Cette métrique représente la marge de manœuvre disponible de la flotte en cas d'augmentation soudaine de la demande des joueurs.

La raison principale du maintien d'une capacité de mémoire tampon est le temps d'attente du joueur. Lorsque les machines à sous sont prêtes et attendent, il faut quelques secondes pour que les nouveaux joueurs puissent participer aux sessions de jeu. Si aucune ressource n'est disponible, les joueurs doivent patienter que les sessions de jeu existantes se terminent ou que de nouvelles ressources deviennent disponibles. Le démarrage de nouvelles instances et de nouveaux processus de serveur peut prendre quelques minutes.

Lorsque vous configurez le dimensionnement automatique basé sur les cibles, spécifiez la taille de la zone tampon que vous souhaitez que le parc conserve. Comme elle `PercentAvailableGameSessions` mesure le pourcentage des ressources disponibles, la taille réelle de la zone tampon est un pourcentage de la capacité totale de la flotte. Amazon GameLift

Servers ajoute ou supprime des instances pour maintenir la taille de la mémoire tampon cible. Avec une grande mémoire tampon, vous réduisez le temps d'attente, mais vous payez également pour des ressources supplémentaires que vous n'utiliserez peut-être pas. Si vos joueurs sont plus tolérant en ce qui concerne les temps d'attente, vous pouvez réduire les coûts en définissant un tampon plus petit.

Pour définir la mise à l'échelle automatique basée sur les cibles

Console

1. Ouvrez le fichier [Amazon GameLift Servers console](#).
2. Dans le volet de navigation, choisissez Hosting, Fleets.
3. Sur la page Flottes, choisissez le nom d'une flotte active pour ouvrir la page détaillée de la flotte.
4. Choisissez l'onglet Mise à l'échelle. Cet onglet affiche les métriques de mise à l'échelle de l'historique du parc et contient des contrôles pour l'ajustement des paramètres de mise à l'échelle actuels.
5. Sous Capacité de mise à l'échelle, vérifiez que les limites de taille minimale et maximale sont adaptées à la flotte. Lorsque la mise à l'échelle automatique est activée, la capacité s'ajuste entre ces deux limites.
6. Dans la politique d'auto-scaling basée sur la cible, choisissez Modifier.
7. Dans la boîte de dialogue Modifier la politique d'auto-scaling basée sur les cibles, pour Pourcentage de sessions de jeu disponibles, définissez le pourcentage que vous souhaitez maintenir, puis choisissez Confirmer. Après avoir confirmé les paramètres, Amazon GameLift Servers ajoute une nouvelle politique basée sur les cibles sous la politique d'auto-scaling basée sur les cibles.

AWS CLI

1. Définissez les limites de capacité. Définissez les valeurs limites à l'aide de la [update-fleet-capacity](#) commande. Pour de plus amples informations, veuillez consulter [Définissez des limites Amazon GameLift Servers de capacité](#).
2. Créez une stratégie. Ouvrez une fenêtre de ligne de commande et utilisez la [put-scaling-policy](#) commande avec les paramètres de votre politique. Pour mettre à jour une stratégie existante, spécifiez le nom de la stratégie et indiquez une version complète de la stratégie mise à jour.

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--target-configuration <buffer size>
```

Exemple :

```
aws gamelift put-scaling-policy \
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \
  --name "My_Target_Policy_1" \
  --policy-type "TargetBased" \
  --metric-name "PercentAvailableGameSessions" \
  --target-configuration "TargetValue=5"
```

Mise à l'échelle automatique avec des politiques basées sur des règles

Les politiques de dimensionnement basées sur des règles Amazon GameLift Servers fournissent un contrôle précis lors de l'adaptation automatique de la capacité d'une flotte en fonction de l'activité des joueurs. Pour chaque politique, vous pouvez associer le dimensionnement à l'un des nombreux indicateurs de flotte, identifier un point de déclenchement et personnaliser l'événement de montée en charge ou de réduction correspondant. Les politiques basées sur des règles sont utiles pour compléter la mise à l'[échelle basée sur des cibles](#) afin de faire face à des circonstances particulières.

Une politique basée sur des règles stipule ce qui suit : « Si une métrique de flotte atteint ou dépasse une valeur seuil pendant un certain temps, modifiez la capacité de la flotte d'un montant spécifié. » Cette rubrique décrit la syntaxe utilisée pour construire une déclaration de stratégie et fournit une aide pour la création et la gestion de vos règles basées sur des règles.

Gérer les stratégies basées sur des règles

Créez, mettez à jour ou supprimez des politiques basées sur des règles à l'aide d'un AWS SDK ou du AWS Command Line Interface (AWS CLI) avec l'API de [service](#) pour Amazon GameLift Servers. Vous pouvez afficher tous les stratégies actives sur la console Amazon GameLift Servers.

Pour arrêter temporairement toutes les politiques de dimensionnement d'une flotte, utilisez la AWS CLI commande [stop-fleet-actions](#).

Pour créer ou mettre à jour une politique de dimensionnement basée sur des règles (AWS CLI) :

1. Définissez les limites de capacité. Définissez l'une ou les deux valeurs limites à l'aide de la [update-fleet-capacity](#) commande. Pour de plus amples informations, veuillez consulter [Définissez des limites Amazon GameLift Servers de capacité](#).
2. Créez une stratégie. Ouvrez une fenêtre de ligne de commande et utilisez la [put-scaling-policy](#) commande avec les paramètres de votre politique. Pour mettre à jour une stratégie existante, spécifiez le nom de la stratégie et indiquez une version complète de la stratégie mise à jour.

```
--fleet-id <unique fleet identifier>
--name "<unique policy name>"
--policy-type <target- or rule-based policy>
--metric-name <name of metric>
--comparison-operator <comparison operator>
--threshold <threshold integer value>
--evaluation-periods <number of minutes>
--scaling-adjustment-type <adjustment type>
--scaling-adjustment <adjustment amount>
```

Exemple :

```
aws gamelift put-scaling-policy \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --name "Scale up when AGS<50" \
  --policy-type RuleBased \
  --metric-name AvailableGameSessions \
  --comparison-operator LessThanThreshold \
  --threshold 50 \
  --evaluation-periods 10 \
  --scaling-adjustment-type ChangeInCapacity \
  --scaling-adjustment 1
```

Pour supprimer une politique de dimensionnement basée sur des règles à l'aide de : AWS CLI

- Ouvrez une fenêtre de ligne de commande et utilisez la [delete-scaling-policy](#) commande avec l'ID de flotte et le nom de la politique.

Exemple :

```
aws gamelift delete-scaling-policy \  
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
  --name "Scale up when AGS<50"
```

Syntaxe des règles de mise à l'échelle automatique

Pour élaborer une déclaration de politique de dimensionnement basée sur des règles, spécifiez six variables :

S'il *<metric name>* reste *<comparison operator>* *<threshold value>* pour *<evaluation period>*, modifiez la capacité de la flotte en utilisant *<adjustment type>* to/by *<adjustment value>*.

Par exemple, cette déclaration de politique déclenche un événement de mise à l'échelle chaque fois que la capacité supplémentaire d'une flotte est inférieure à ce qui est nécessaire pour gérer 50 nouvelles sessions de jeu :

Si AvailableGameSessions reste à less than 50 pendant 10 minutes, modifier à la capacité de la flotte avec ChangeInCapacity par 1 instances.

Nom de la métrique

Pour démarrer un événement de dimensionnement, associez une politique de dimensionnement automatique à l'une des mesures spécifiques à la flotte suivantes. Pour une description complète des métriques, voir [Métriques Amazon GameLift Servers pour les flottes](#).

- Activation de sessions de jeu
- Sessions de jeu actives
- Sessions de jeu disponibles
- Pourcentage de sessions de jeu disponibles
- Instances actives
- Sessions de joueur disponibles
- Sessions de joueur actuelles
- Instances inactives
- Pourcentage d'instances inactives

Si la flotte se trouve dans une file d'attente de session de jeu, vous pouvez utiliser les statistiques suivantes :

- Profondeur de la file d'attente — Le nombre de demandes de session de jeu en attente pour lesquelles cette flotte est le meilleur site d'hébergement disponible.
- Temps d'attente : temps d'attente spécifique à la flotte. Durée pendant laquelle la demande de session de jeu plus ancienne attend d'être traitée. Le temps d'attente d'une flotte correspond à la durée passée en file d'attente de la plus ancienne demande actuelle.

Opérateur de comparaison

Indique Amazon GameLift Servers comment comparer les données métriques à la valeur du seuil. Les opérateurs de comparaison valides sont les suivants : supérieur à (>), inférieur à (= <), greater than or equal (>) et inférieur ou égal (<=).

Valeur de seuil

Lorsque la valeur métrique spécifiée atteint ou dépasse la valeur seuil, elle déclenche un événement de mise à l'échelle. Cette valeur est toujours un nombre entier positif.

Période d'évaluation

La métrique doit atteindre ou dépasser le seuil pendant toute la durée de la période d'évaluation avant de démarrer un événement de dimensionnement. La longueur de la période d'évaluation est consécutive ; si la métrique s'écarte du seuil, la période d'évaluation recommence.

Type et valeur d'ajustement

Cet ensemble de variables fonctionne ensemble pour spécifier comment Amazon GameLift Servers ajuster la capacité de la flotte lorsqu'un événement de dimensionnement commence. Choisissez parmi trois types de réglage possibles :

- Modification de la capacité — Augmentez ou diminuez la capacité actuelle d'un nombre spécifié d'instances. Définissez la valeur d'ajustement en fonction du nombre d'instances à ajouter ou à retirer du parc. Les valeurs positives ajoutent des instances, tandis que les valeurs négatives retirent des instances. Par exemple, une valeur de « -10 » réduit la flotte de 10 instances, quelle que soit la taille totale de la flotte.
- Variation en pourcentage de la capacité — Augmentez ou diminuez la capacité actuelle d'un pourcentage spécifié. Définissez la valeur d'ajustement selon le pourcentage par lequel vous souhaitez augmenter ou diminuer la capacité de la flotte. Les valeurs positives ajoutent des instances, tandis que les valeurs négatives retirent des instances. Par exemple, pour un parc de 50 instances, une variation en pourcentage de « 20 » ajoute 10 instances au parc.

- **Capacité exacte** — Augmentez ou diminuez la capacité actuelle jusqu'à une valeur spécifique. Définissez la valeur d'ajustement sur le nombre exact d'instances que vous voulez maintenir dans le parc.

Conseils pour une mise à l'échelle automatique basée sur des règles

Les suggestions suivantes peuvent vous aider à tirer le meilleur parti de la mise à l'échelle automatique grâce à des politiques basées sur des règles.

Utilisation de plusieurs stratégies

Vous pouvez avoir plusieurs politiques de dimensionnement automatique pour une flotte en même temps. Le scénario le plus courant consiste à avoir une stratégie basée sur la cible pour gérer la plupart des besoins de dimensionnement et d'utiliser des stratégies basées sur des règles pour gérer les cas limites. Il n'y a aucune limite à l'utilisation de plusieurs politiques.

Dans le cas de plusieurs politiques, chaque stratégie se comporte de manière indépendante. Il n'existe aucun moyen de contrôler la séquence des événements de dimensionnement. Par exemple, si plusieurs règles régissent la montée en gamme, il est possible que l'activité des joueurs déclenche plusieurs événements de mise à l'échelle simultanément. Évitez les politiques qui démarrent les unes les autres. Par exemple, vous pouvez créer une boucle infinie si vous créez des politiques de mise à l'échelle et de réduction qui fixent la capacité au-delà du seuil défini l'une par rapport à l'autre.

Définir une capacité maximale et minimale

Chaque parc comporte une limite de capacité maximale et minimale. Cette fonctionnalité est importante lors de l'utilisation de la mise à l'échelle automatique. La mise à l'échelle automatique ne définit jamais la capacité à une valeur située en dehors de cette plage. Par défaut, les flottes nouvellement créées ont un minimum de 0 et un maximum de 1. Pour que votre politique de dimensionnement automatique affecte la capacité comme prévu, augmentez la valeur maximale.

La capacité de la flotte est également limitée par les limites relatives au type d'instance de la flotte et par les quotas de service dans votre Compte AWS parc. Vous ne pouvez pas définir de minimum et de maximum en dehors de ces limites et quotas de compte.

Métriques de suivi après une modification de la capacité

Après avoir modifié la capacité en réponse à une politique de dimensionnement automatique, Amazon GameLift Servers attendez 10 minutes avant de répondre aux déclencheurs de la même

politique. Cette attente donne le Amazon GameLift Servers temps d'ajouter les nouvelles instances, de lancer les serveurs de jeu, de connecter les joueurs et de commencer à collecter des données à partir des nouvelles instances. Pendant ce temps, Amazon GameLift Servers évalue la stratégie par rapport à la métrique et suit la période d'évaluation de la politique, qui redémarre après la survenue d'un événement de dimensionnement. Cela signifie qu'une politique de dimensionnement peut déclencher un autre événement de dimensionnement immédiatement après la fin du temps d'attente.

Il n'y a aucun temps d'attente entre les événements de dimensionnement et le lancement de différentes politiques de dimensionnement automatique.

Échelle des flottes de Amazon GameLift Servers conteneurs

L'une des tâches les plus difficiles de l'hébergement de jeux consiste à augmenter la capacité pour répondre à la demande des joueurs sans gaspiller de l'argent en ressources dont vous n'avez pas besoin. Dans un parc de conteneurs géré, vous augmentez la capacité de votre flotte en ajoutant ou en supprimant des instances de flotte.

Lorsque vous créez une nouvelle flotte, définissez Amazon GameLift Servers la capacité souhaitée de la flotte sur une instance et déployez une instance dans la région d'origine de la flotte. Dans le cas d'un parc multisite, Amazon GameLift Servers déploie une instance dans la région d'origine et sur chaque site distant. Une fois l'état de la flotte atteint **ACTIVE**, vous pouvez augmenter la capacité souhaitée pour augmenter ou diminuer la capacité souhaitée pour la réduire.

Vous pouvez utiliser les fonctionnalités de Amazon GameLift Servers dimensionnement pour modifier la capacité manuellement ou configurer une mise à l'échelle automatique en fonction de la demande des joueurs :

- Configurez le dimensionnement automatique avec le suivi des cibles. Consultez [Mise à l'échelle automatique basée sur les cibles](#).
- Modifiez manuellement la capacité de votre flotte. Consultez [Définissez manuellement la capacité d'une Amazon GameLift Servers flotte](#).

Lorsque vous agrandissez une flotte de conteneurs, réfléchissez à l'impact de l'ajout ou de la suppression d'instances sur la capacité de la flotte à accueillir des sessions de jeu et des joueurs.

- Sessions de jeu par instance
 - Chaque processus de serveur de jeu exécuté sur une instance représente la capacité d'héberger une session de jeu.

- Utilisez cette formule pour calculer le nombre de sessions de jeu exécutées simultanément sur une instance de flotte de conteneurs :

```
[Game sessions per instance] = [# of game server processes per game server container] * [# of game server container groups per instance]
```

Si votre architecture de conteneur exécute simultanément un processus de serveur de jeu dans le conteneur de serveur de jeu, les sessions de jeu par instance sont égales au nombre de groupes de conteneurs de serveurs de jeu par instance.

- Pour les groupes de conteneurs de serveurs de jeu par instance, appelez [DescribeContainerFleet](#) pour obtenir la `MaximumGameServerContainerGroupsPerInstance` valeur `GameServerContainerGroupsPerInstance` or.
- Joueurs par instance
 - Vous décidez du nombre de machines à sous à autoriser lors de chaque session de jeu. Selon la façon dont votre solution d'hébergement gère le placement des sessions de jeu, vous pouvez définir les joueurs par session de jeu dans votre configuration de matchmaking ou dans vos appels pour démarrer un placement de session de jeu.
 - Utilisez cette formule pour calculer le nombre de joueurs qui peuvent jouer à votre jeu simultanément sur une instance de flotte de conteneurs :

```
[Players per instance] = [# of game sessions per instance] * [# of player slots per game session]
```

Pour connaître la capacité totale actuelle d'une flotte de conteneurs, appelez [DescribeFleetCapacity](#) ou [DescribeFleetLocation Capacity](#) pour obtenir le nombre de groupes de conteneurs de serveurs de jeu que compte la flotte. Les groupes actifs sont ceux qui hébergent actuellement des sessions de jeu. Les groupes inactifs sont prêts à héberger une nouvelle session de jeu. Multipliez ces valeurs par le nombre de processus de serveur par groupe de conteneurs de serveurs de jeu.

Préparation de votre jeu pour le lancement grâce à l'Amazon GameLift Servershébergement

Utilisez les listes de contrôle suivantes pour valider chaque phase de déploiement de votre jeu. Les éléments marqués [Critique] sont essentiels pour le lancement de votre production.

Téléchargez et complétez le questionnaire de Amazon GameLift Servers lancement, disponible dans la [Amazon GameLift Serversconsole](#). Nous voulons que chaque développeur de jeux Amazon GameLift Servers passe une journée de lancement fluide, et les informations demandées nous aident à vous préparer aux prochains tests de charge, au lancement progressif ou au lancement public à venir. Prévoyez de soumettre le questionnaire rempli au moins trois (3) mois avant d'effectuer votre premier test de charge.

Rubriques

- [Préparez votre jeu](#)
- [Préparez-vous pour les tests](#)
- [Préparer le lancement](#)
- [Planifiez les mises à jour après le lancement](#)

Préparez votre jeu

- [Critique] Vérifiez que vous avez terminé toutes les [étapes du plan de développement](#) de votre solution d'hébergement et que vous disposez de tous les composants requis, notamment un serveur de jeu intégré, un service principal pour les clients de jeu, des flottes d'hébergement et une méthode de placement des sessions de jeu (telle qu'une file d'attente).
- [Critique] [Créez des rôles AWS Identity and Access Management \(IAM\)](#) qui permettent à votre serveur de jeu d'accéder à d'autres AWS ressources pendant son exécution.
- [Critique] Concevez et implémentez le basculement vers d'autres ressources d'hébergement selon les besoins.
- [Planifiez le déploiement des flottes vers vos destinations cibles](#), en tenant compte de la file d'attente et de la structure des flottes de votre jeu.
- [Automatisez votre déploiement](#) en utilisant l'infrastructure sous forme de code (IaC) avec AWS CloudFormation et le AWS Cloud Development Kit (AWS CDK).

- [Collectez des journaux et des analyses](#) à l'aide d'Amazon CloudWatch et d'Amazon Simple Storage Service (Amazon S3).

Préparez-vous pour les tests

- [Critique] [Demandez des augmentations pour les quotas de Amazon GameLift Servers service](#) et autres Service AWS quotas afin que votre environnement réel puisse s'adapter aux besoins de production.
- [Critique] Vérifiez que les ports ouverts sur les flottes actives correspondent à la gamme de ports que vos serveurs peuvent utiliser.
- [Critique] Fermez le port RDP 3389 et le port SSH 22.
- Élaborez un plan pour la DevOps gestion de votre jeu. Si vous utilisez Amazon CloudWatch Logs ou les métriques CloudWatch personnalisées d'Amazon, définissez des alarmes en cas de problèmes graves ou critiques sur le parc de serveurs. Simulez les défaillances et testez les runbooks.
- Vérifiez que les ressources de calcul que vous utilisez peuvent prendre en charge le nombre de processus serveur que vous souhaitez exécuter simultanément sur chaque calcul.
- [Ajustez votre politique de dimensionnement](#) pour qu'elle soit plus prudente dans un premier temps et offrez une capacité d'inactivité supérieure à ce dont vous pensez avoir besoin. Vous pourrez optimiser les coûts ultérieurement. Envisagez d'utiliser une politique de dimensionnement basée sur des cibles avec une capacité inutilisée de 20 %.
- En effetFlexMatch, utilisez des [règles de latence](#) pour associer des joueurs géographiquement proches les uns des autres. Testez son comportement sous charge à l'aide des données de latence synthétiques de votre client de test de charge.
- Testez la charge de votre infrastructure d'authentification des joueurs et de session de jeu pour voir si elle s'adapte efficacement à la demande.
- Vérifiez qu'un serveur qui fonctionne depuis plusieurs jours peut toujours accepter des connexions.
- Élevez le niveau de votre Support forfait à Business ou Entreprise afin de AWS pouvoir vous répondre en cas de problème ou de panne.

Préparer le lancement

- [Critique] [Définissez la politique de protection de la flotte](#) pour une protection complète de toutes les flottes actives afin que la réduction des effectifs n'interrompe pas les sessions de jeu actives.

- [Critique] [Fixez des tailles maximales de flotte](#) suffisamment élevées pour répondre au minimum à la demande maximale prévue. Nous vous recommandons de doubler votre taille maximale en cas de demande imprévue.
- Encouragez l'ensemble de votre équipe de développement à participer à l'événement de lancement et à suivre le lancement de votre jeu dans une salle de lancement.
- Surveillez la latence et l'expérience des joueurs.

Planifiez les mises à jour après le lancement

- [Ajustez la politique de dimensionnement](#) pour minimiser la capacité d'inactivité en fonction de l'utilisation des joueurs.
- [Modifiez FlexMatch les règles](#) ou [ajoutez des emplacements d'hébergement en fonction des](#) données de latence des joueurs et des exigences révisées.
- Optimisez la configuration d'exécution pour exécuter autant de sessions de jeu que possible sur chaque ressource informatique. L'optimisation de l'efficacité des performances de cette manière peut avoir une incidence directe sur les coûts de votre parc, car vous pourriez être en mesure d'exécuter davantage de processus de serveur avec les mêmes ressources de calcul.
- [Utilisez vos données d'analyse](#) pour favoriser le développement continu, améliorer l'expérience des joueurs et la longévité du jeu, et optimiser la monétisation.

Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation

Vous pouvez utiliser AWS CloudFormation pour gérer votre Amazon GameLift Servers ressources. Dans AWS CloudFormation, vous créez un modèle qui modélise chaque ressource, puis vous l'utilisez pour créer vos ressources. Pour mettre à jour les ressources, vous devez apporter les modifications nécessaires à votre modèle et l'utiliser AWS CloudFormation pour implémenter les mises à jour. Vous pouvez organiser vos ressources en groupes logiques, appelés piles et ensembles de piles.

Utilisation AWS CloudFormation pour entretenir votre Amazon GameLift Servers l'hébergement de ressources offre un moyen plus efficace de gérer des ensembles de AWS ressources. Vous pouvez utiliser le contrôle de version pour suivre les modifications apportées au modèle au fil du temps et coordonner les mises à jour effectuées par plusieurs membres de l'équipe. Vous pouvez également réutiliser des modèles. Par exemple, lors du déploiement d'un jeu dans plusieurs régions, vous pouvez utiliser le même modèle pour créer des ressources identiques dans chaque région. Vous pouvez également utiliser ces modèles pour déployer les mêmes ensembles de ressources dans une autre partition.

Pour plus d'informations AWS CloudFormation, consultez le [guide de AWS CloudFormation l'utilisateur](#). Pour afficher les informations du modèle pour Amazon GameLift Servers ressources, consultez le [Amazon GameLift Servers référence au type de ressource](#).

Bonnes pratiques

Pour obtenir des instructions détaillées sur l'utilisation AWS CloudFormation, consultez les [AWS CloudFormation meilleures pratiques](#) du Guide de AWS CloudFormation l'utilisateur. En outre, ces meilleures pratiques revêtent une importance particulière pour Amazon GameLift Servers.

- Gérez constamment vos ressources grâce à AWS CloudFormation. Si vous modifiez vos ressources en dehors de AWS CloudFormation vos ressources, vous serez désynchronisé avec vos modèles de ressources.
- Utilisez des AWS CloudFormation piles et des ensembles de piles pour gérer efficacement plusieurs ressources.
 - Utilisez des piles pour gérer des groupes de ressources connectées. Par exemple, une pile contenant une version, une flotte faisant référence à la version et un alias faisant référence à

la flotte. Si vous mettez à jour votre modèle pour remplacer une version, AWS CloudFormation remplace les flottes connectées à la version. AWS CloudFormation met ensuite à jour les alias existants pour qu'ils pointent vers les nouvelles flottes. Pour plus d'informations, consultez la section [Utilisation des piles](#) dans le guide de l'AWS CloudFormation utilisateur.

- Utilisez des ensembles de AWS CloudFormation piles si vous déployez des piles identiques dans plusieurs régions ou AWS comptes. Pour plus d'informations, consultez la section [Travailler avec des ensembles de piles](#) dans le Guide de AWS CloudFormation l'utilisateur.
- Si vous utilisez des instances Spot, incluez une flotte à la demande comme sauvegarde. Nous vous recommandons de configurer vos modèles avec deux flottes dans chaque région, une flotte avec des instances Spot et une flotte avec des instances à la demande.
- Regroupez vos ressources spécifiques à un site et vos ressources globales dans des piles distinctes lorsque vous gérez des ressources sur plusieurs sites.
- Placez vos ressources mondiales à proximité des services qui les utilisent. Les ressources telles que les files d'attente et les configurations de matchmaking ont tendance à recevoir un volume élevé de demandes provenant de sources spécifiques. En plaçant vos ressources à proximité de la source de ces demandes, vous réduisez le temps de trajet des demandes et pouvez améliorer les performances globales.
- Placez votre configuration de mise en relation dans la même région que la file d'attente de session de jeu qu'elle utilise.
- Créez un alias distinct pour chaque flotte de la pile.

Utiliser des AWS CloudFormation piles

Nous recommandons d'utiliser les structures suivantes lors de la configuration des AWS CloudFormation piles pour Amazon GameLift Servers ressources. La structure de stack optimale varie selon que vous déployez votre jeu à un ou plusieurs endroits.

Des piles pour un seul emplacement

Pour gérer Amazon GameLift Servers ressources réunies en un seul endroit, nous recommandons une structure à deux piles :

- Pile de support — Cette pile contient les ressources que votre Amazon GameLift Servers les ressources dépendent de. Au minimum, cette pile doit inclure le compartiment S3 dans lequel vous stockez votre serveur de jeu personnalisé ou vos fichiers de script Realtime. La pile doit également inclure un rôle IAM qui donne Amazon GameLift Servers autorisation de récupérer vos

fichiers depuis le compartiment S3 lors de la création d'un Amazon GameLift Servers ressource de construction ou de script. Cette pile peut également contenir d'autres AWS ressources utilisées avec votre jeu, telles que des tables DynamoDB, des clusters Amazon Redshift et des fonctions Lambda.

- **Amazon GameLift Servers pile** — Cette pile contient tous vos Amazon GameLift Servers des ressources, notamment le build ou le script, un ensemble de flottes, des alias et une file d'attente de session de jeu. AWS CloudFormation crée une ressource de build ou de script avec des fichiers stockés dans l'emplacement du compartiment S3 et déploie le build ou le script sur une ou plusieurs ressources de flotte. Chaque flotte doit avoir un alias correspondant. La file d'attente de session de jeu fait référence à tout ou partie des alias de flotte. Si vous l'utilisez FlexMatch pour le matchmaking, cette pile contient également une configuration de matchmaking et un ensemble de règles.

Le schéma ci-dessous illustre une structure à deux piles permettant de déployer des ressources dans une seule AWS région.

Des piles pour plusieurs régions

Lorsque vous déployez votre jeu dans plusieurs régions, gardez présent à l'esprit la manière dont les ressources peuvent interagir entre les régions. Certaines ressources, telles que Amazon GameLift Servers flottes, ne peuvent référencer que d'autres ressources de la même région. D'autres ressources, telles qu'un Amazon GameLift Servers file d'attente, sont indépendants de la région. Pour gérer Amazon GameLift Servers ressources dans plusieurs régions, nous recommandons la structure suivante.

- **Stacks d'assistance régionaux** — Ces piles contiennent des ressources que votre Amazon GameLift Servers les ressources dépendent de. Cette pile doit inclure le compartiment S3 dans lequel vous stockez votre serveur de jeu personnalisé ou vos fichiers de script Realtime. Il peut également contenir d'autres AWS ressources pour votre jeu, telles que des tables DynamoDB, des clusters Amazon Redshift et des fonctions Lambda. Bon nombre de ces ressources sont spécifiques à une région, vous devez donc les créer dans chaque région. Amazon GameLift Servers a également besoin d'un rôle IAM qui permet d'accéder à ces ressources de support. Comme un rôle IAM est indépendant de la région, vous n'avez besoin que d'une seule ressource de rôle, placée dans n'importe quelle région et référencée dans toutes les autres piles de support.
- **Régional Amazon GameLift Servers stacks** — Cette pile contient les Amazon GameLift Servers des ressources qui doivent exister dans chaque région où votre jeu est déployé, notamment le

build ou le script, un ensemble de flottes et des alias. AWS CloudFormation crée une ressource de build ou de script avec des fichiers dans un emplacement de compartiment S3, et déploie la build ou le script sur une ou plusieurs ressources de flotte. Chaque flotte doit avoir un alias correspondant. La file d'attente de session de jeu fait référence à tout ou partie des alias de flotte. Vous pouvez gérer un modèle pour décrire ce type de pile et l'utiliser pour créer des ensembles de ressources identiques dans chaque région.

- **Mondial Amazon GameLift Servers pile** — Cette pile contient votre file d'attente de session de jeu et vos ressources de matchmaking. Ces ressources peuvent être situées dans n'importe quelle région et sont généralement placées dans la même région. La file d'attente peut faire référence à des flottes ou à des alias situés dans n'importe quelle région. Pour placer des files d'attente supplémentaires dans différentes régions, créez des piles globales supplémentaires.

Les diagrammes ci-dessous illustrent une structure multistack permettant de déployer des ressources dans plusieurs AWS régions. Le premier diagramme montre une structure pour une file d'attente de session de jeu unique. Le deuxième diagramme montre une structure avec plusieurs files d'attente.

Mettre à jour les versions

Amazon GameLift Servers les builds sont immuables, tout comme la relation entre un build et une flotte. Par conséquent, lorsque vous mettez à jour vos ressources d'hébergement pour utiliser un nouvel ensemble de fichiers de build de jeu, les étapes suivantes doivent être exécutées :

- Créez une nouvelle build en utilisant le nouvel ensemble de fichiers (remplacement).
- Créez un nouvel ensemble de flottes pour déployer la nouvelle build de jeu (remplacement).
- Redirigez les alias pour pointer vers les nouvelles flottes (mise à jour sans interruption).

Pour plus d'informations, voir [Mettre à jour les comportements des ressources de pile](#) dans le Guide de AWS CloudFormation l'utilisateur.

Déployez les mises à jour de build

Lors de la mise à jour d'une pile contenant des ressources de build, de flotte et d'alias associées, le AWS CloudFormation comportement par défaut consiste à exécuter automatiquement ces étapes dans l'ordre. Vous déclenchez cette mise à jour en téléchargeant d'abord les nouveaux fichiers de

build vers un nouvel emplacement S3. Vous modifiez ensuite votre modèle de AWS CloudFormation build pour qu'il pointe vers le nouvel emplacement S3. Lorsque vous mettez à jour votre pile avec le nouvel emplacement S3, la séquence AWS CloudFormation suivante se déclenche :

1. Récupère les nouveaux fichiers de S3, valide les fichiers et crée un nouveau Amazon GameLift Servers construire.
2. Mise à jour de la référence de build dans le modèle de flotte, ce qui déclenche la création d'une nouvelle flotte.
3. Lorsque les nouvelles flottes sont actives, mise à jour de la référence de flotte dans l'alias, ce qui déclenche la mise à jour de l'alias pour cibler les nouvelles flottes.
4. Suppression de l'ancienne flotte.
5. Suppression de l'ancienne build.

Si votre file d'attente de session de jeu utilise des alias de flotte, le trafic des joueurs est automatiquement basculé vers les nouvelles flottes dès que les alias sont mis à jour. Les anciennes flottes sont progressivement vidées de leurs joueurs à mesure que les sessions de jeu se terminent. La mise à l'échelle automatique gère la tâche d'ajout et de suppression des instances à partir de chaque ensemble de flottes lorsque le trafic des joueurs fluctue. Vous pouvez également spécifier un nombre d'instances initial souhaité pour accélérer rapidement la montée en charge du commutateur et activer la mise à l'échelle automatique ultérieurement.

Vous pouvez également AWS CloudFormation conserver les ressources au lieu de les supprimer. Pour plus d'informations, consultez [RetainResources](#) dans la Référence d'API AWS CloudFormation .

Déployer les mises à jour de build

Si vous souhaitez contrôler davantage le moment où les nouvelles flottes sont en ligne pour les joueurs, vous disposez de plusieurs options. Vous pouvez choisir de gérer les alias manuellement à l'aide du Amazon GameLift Servers console ou CLI. Au lieu de mettre à jour votre modèle de build pour remplacer la build et les flottes, vous pouvez ajouter un deuxième ensemble de définitions de build et de flotte à votre modèle. Lorsque vous mettez à jour le modèle, il AWS CloudFormation crée une deuxième ressource de construction et les flottes correspondantes. Les ressources existantes n'étant pas remplacées, elles ne sont pas supprimées et les alias continuent de pointer vers les flottes d'origine.

Le principal avantage de cette approche est qu'elle vous donne de la flexibilité. Vous pouvez créer des ressources distinctes pour la nouvelle version de votre build, tester les nouvelles ressources et

contrôler le moment où les nouvelles flottes sont mises en ligne pour les joueurs. Un inconvénient potentiel est que cela nécessite deux fois plus de ressources dans chaque région pendant une brève période.

Le schéma suivant illustre ce processus.

Comment fonctionnent les annulations

Lors de l'exécution d'une mise à jour des ressources, si une étape ne se termine pas correctement, AWS CloudFormation lance automatiquement une restauration. Ce processus inverse chaque étape dans l'ordre, en supprimant les ressources nouvellement créées.

Si vous devez déclencher manuellement une restauration, restaurez l'emplacement d'origine de la clé d'emplacement S3 du modèle de build, puis mettez à jour votre pile. Un nouveau Amazon GameLift Servers build et fleet sont créés, et l'alias passe à la nouvelle flotte une fois que celle-ci est active. Si vous gérez des alias séparément, vous devez les modifier de sorte qu'ils pointent vers les nouvelles flottes.

Pour plus d'informations sur la gestion d'une annulation qui échoue ou se bloque, voir [Poursuivre l'annulation d'une mise à jour](#) dans le Guide de l'AWS CloudFormation utilisateur.

Surveillance des Amazon GameLift Servers

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que fonctionnalité autonome avec Amazon EC2, consultez la section [Sécurité dans Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité Amazon GameLift Servers et des performances de vos autres AWS solutions. Les métriques sont utilisées avec Amazon GameLift Servers pour trois raisons principales : pour surveiller l'état du système et configurer des alarmes, pour suivre les performances et l'utilisation du serveur de jeux, et pour gérer la capacité à l'aide de la scalabilité automatique ou manuelle.

AWS fournit les outils de surveillance suivants pour surveiller Amazon GameLift Servers, signaler tout problème et prendre des mesures automatiques le cas échéant :

- Amazon GameLift Servers console — Utilisez l'interface graphique pour gérer vos Amazon GameLift Servers ressources et suivre l'activité d'hébergement de jeux.
- Amazon CloudWatch — Vous pouvez surveiller Amazon GameLift Servers les statistiques en temps réel, ainsi que les mesures relatives aux autres AWS ressources et applications que vous exécutez sur AWS des services. CloudWatch propose une suite de fonctionnalités de surveillance, notamment des outils permettant de créer des tableaux de bord personnalisés et la possibilité de définir des alarmes qui signalent ou prennent des mesures lorsqu'une métrique atteint un seuil spécifié.
- AWS CloudTrail— capture tous les appels d'API et les événements connexes effectués par ou au nom de votre AWS compte pour Amazon GameLift Servers d'autres AWS services. Les données sont fournies sous forme de fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels.
- Journaux de session de jeu : vous pouvez générer des messages de serveur personnalisés pour vos sessions de jeu dans des fichiers journaux stockés dans Amazon S3.

Comparaison des métriques entre les sources de surveillance

Amazon GameLift Servers fournit des métriques via trois sources principales : les métriques Fleet Activity de la Amazon GameLift Servers console, le plug-in Amazon GameLift Servers Servers Metrics pour Unreal et CloudWatch Amazon GameLift Servers les métriques Amazon. Comprendre

le chevauchement et les capacités uniques de chaque source vous aide à choisir l'approche de surveillance adaptée à vos besoins.

Disponibilité des métriques par source

Les sections suivantes décrivent les indicateurs disponibles dans les différentes sources de surveillance, en mettant en évidence les chevauchements et les fonctionnalités uniques.

Métriques disponibles dans toutes les sources

La métrique suivante est toujours disponible sur la console, le plugin et CloudWatch :

- `Active Instances/ActiveInstances`— Le nombre d'instances avec le statut ACTIVE qui exécutent des processus de serveur actifs

Métriques disponibles dans la console et CloudWatch

Les métriques suivantes sont disponibles à la fois sur la Amazon GameLift Servers console et sur Amazon CloudWatch, ce qui permet une gestion cohérente du parc et un suivi des capacités :

- `Idle instances/IdleInstances`— Instances actives n'hébergeant actuellement aucune session de jeu
- `Percent idle instances/PercentIdleInstances`— Pourcentage d'instances actives inactives
- `Desired instances/DesiredInstances`— Nombre cible d'instances actives Amazon GameLift Servers maintenues
- `Max instances/MaxInstances`— Nombre maximum d'instances autorisées pour la flotte
- `Min instances/MinInstances`— Nombre minimum d'instances autorisées pour la flotte
- `Instance interruptions/InstanceSpotInterruptions`— Nombre d'instances ponctuelles qui ont été interrompues
- `Recycled instances/RecycledSpotInstances`— Nombre d'instances ponctuelles qui ont été recyclées et remplacées
- `Game session interruptions/GameSessionSpotInterruptions`— Sessions de jeu interrompues en raison de l'interruption d'une instance ponctuelle
- `Unhealthy instances replaced/UnhealthyInstancesReplaced`— Instances remplacées en raison de problèmes de santé
- `Activating game sessions`— Les sessions de jeu sont en cours de démarrage

- **Active game sessions**— Sessions de jeu en cours et hébergeant des joueurs
- **Available game sessions**— Sessions de jeu pouvant accepter des joueurs supplémentaires
- **Concurrent activatable game sessions**— Sessions de jeu pouvant être activées simultanément
- **Percent available game sessions**— Pourcentage de sessions de jeu disponibles pour les nouveaux joueurs

Métriques propres à CloudWatch

Amazon CloudWatch fournit des indicateurs opérationnels spécifiques :

- **Unhealthy instances replaced**— Instances remplacées en raison de problèmes de santé

Mesures propres aux flottes de conteneurs

Les flottes de conteneurs fournissent des indicateurs supplémentaires spécifiques aux serveurs de jeux conteneurisés :

Métriques relatives aux groupes de conteneurs

- **ActiveGameServerContainerGroups**— Nombre de groupes de conteneurs de serveurs de jeu actifs
- **IdleGameServerContainerGroups**— Nombre de groupes de conteneurs de serveurs de jeu inactifs
- **PendingGameServerContainerGroups**— Nombre de groupes de conteneurs de serveurs de jeu en attente
- **TerminatingGameServerContainerGroups**— Nombre de groupes de conteneurs de serveurs de jeu résilients
- **UnhealthyGameServerContainerGroupsReplaced**— Nombre de groupes de contenants insalubres qui ont été remplacés

Mesures de performance des conteneurs

- **ContainerCPUUtilizationPerInstance**— Utilisation du processeur par instance de conteneur
- **ContainerCPUReservation**— Capacité CPU réservée aux conteneurs

- `ContainerMemoryReservation`— Capacité de mémoire réservée aux conteneurs
- `ContainerMemoryUtilization`— Utilisation de la mémoire par les conteneurs
- `ContainerNetworkIn`— Trafic réseau entrant pour les conteneurs
- `ContainerNetworkOut`— Trafic réseau sortant pour les conteneurs
- `ContainerStorageReadBytes`— Stockage des octets lus pour les conteneurs
- `ContainerStorageWriteBytes`— Stockage : octets d'écriture pour les conteneurs

Métriques d'instance renommées

- `InstanceCPUUtilization`— Utilisation du processeur au niveau de l'instance (renommé en) `CPUUtilization`
- `InstanceNetworkIn`— Trafic réseau entrant au niveau de l'instance (renommé en) `NetworkIn`
- `InstanceNetworkOut`— Trafic réseau sortant au niveau de l'instance (renommé en) `NetworkOut`
- `InstanceStorageReadBytes`— Octets de lecture du stockage au niveau de l'instance (renommé en) `DiskReadBytes`
- `InstanceStorageWriteBytes`— Octets d'écriture pour le stockage au niveau de l'instance (renommés en) `DiskWriteBytes`
- `InstanceStorageReadOps`— Opérations de lecture du stockage au niveau de l'instance (renommées en) `DiskReadOps`
- `InstanceStorageWriteOps`— Opérations d'écriture sur le stockage au niveau de l'instance (renommées en) `DiskWriteOps`

Choisir la bonne source de surveillance

Sélectionnez votre approche de surveillance en fonction de vos besoins spécifiques :

- Utilisez la Amazon GameLift Servers console pour la gestion de flotte, la planification des capacités et la supervision opérationnelle générale. La console fournit une vue intégrée de l'état de santé de la flotte et de l'activité des joueurs. Pour de plus amples informations, veuillez consulter [the section called “Suivez l'hébergement des jeux sur la console”](#).
- Utilisez Amazon CloudWatch pour automatiser la surveillance, les alertes et l'intégration avec d'autres AWS services. CloudWatch permet des tableaux de bord personnalisés et une automatisation basée sur les alarmes. Pour de plus amples informations, veuillez consulter [the section called “Moniteur avec CloudWatch”](#).

- Utilisez plusieurs sources pour une surveillance complète. Combinez la supervision de la console, les détails techniques des plugins et CloudWatch l'automatisation pour une visibilité complète de votre infrastructure d'hébergement de jeux.

Pour des fonctionnalités de surveillance supplémentaires, vous pouvez également utiliser :

- [the section called “Journalisation des appels d'API”](#)— Suivez les appels d'API et les événements connexes à des fins d'audit et de conformité.
- [the section called “Journalisation des messages du serveur”](#)— Capturez les messages personnalisés du serveur et les journaux de session de jeu.

Rubriques

- [the section called “Suivez l'hébergement des jeux sur la console”](#)
- [the section called “Moniteur avec CloudWatch”](#)
- [the section called “Journalisation des appels d'API”](#)
- [the section called “Journalisation des messages du serveur”](#)

Suivez l'hébergement de jeux dans le Amazon GameLift Servers console

Utilisez la commande Amazon GameLift Servers console pour afficher et gérer les ressources d'hébergement de vos jeux et votre activité d'hébergement continue en temps quasi réel. La console propose une interface graphique pour la plupart des fonctionnalités de l'API de service pour Amazon GameLift Servers. Vous pouvez utiliser la console pour :

- Utilisez le tableau de bord pour obtenir un instantané de haut niveau. Vous pouvez voir les numéros et l'état actuel de tous vos Amazon GameLift Servers héberger des ressources et suivre les liens pour obtenir des détails sur les ressources individuelles.
- Gérez les ressources d'hébergement individuelles. Vous pouvez tout créer, afficher et supprimer Amazon GameLift Servers ressources, et mettez à jour leurs propriétés mutables. Vous pouvez également consulter certains types d'activités d'hébergement, tels que les événements et les indicateurs de performance.

- Interagissez avec le jeu et l'activité des sessions des joueurs. Vous pouvez suivre l'activité des sessions de jeu et des sessions des joueurs par flotte, et utiliser ces informations pour résoudre les problèmes liés aux sessions de jeu. Consultez les détails d'une session de jeu, consultez les sessions des joueurs pour chaque session de jeu et recherchez l'activité des joueurs au cours de plusieurs sessions de jeu. Vous pouvez également arrêter des sessions de jeu individuelles si nécessaire.

Rubriques

- [Tableau de bord d'hébergement dans le Amazon GameLift Servers tableau de bord](#)
- [Le serveur de jeu est intégré au Amazon GameLift Servers tableau de bord](#)
- [Amazon GameLift ServersRealtimescripts dans la Amazon GameLift Servers console](#)
- [Flottes dans la console Amazon GameLift Servers](#)
- [Détails du parc dans la Amazon GameLift Servers console](#)
- [Sessions de jeu et de jeu dans la Amazon GameLift Servers console](#)
- [Alias dans le Amazon GameLift Servers console](#)
- [files d'attente pour les sessions de jeu dans le Amazon GameLift Servers console](#)

Tableau de bord d'hébergement dans le Amazon GameLift Servers tableau de bord

Utilisez la commande Amazon GameLift Servers tableau de bord de console pour obtenir un aperçu de haut niveau de l'état actuel du Amazon GameLift Servers héberger des ressources sur votre AWS compte. La Amazon GameLift Servers le tableau de bord fournit une vue des éléments suivants :

- Le nombre de builds aux statuts Prêt, Initialisé et Échoué. Choisissez Afficher les versions pour plus de détails sur les versions de votre région actuelle.
- Le nombre de flottes dans tous les statuts. Choisissez Afficher les flottes pour obtenir des informations sur les flottes de votre région actuelle.
- Vos ressources actuelles.
- Nouvelles annonces de fonctionnalités et de services.

Pour ouvrir le Amazon GameLift Servers tableau de bord

- Dans [Amazon GameLift Servers console](#), dans le volet de navigation, choisissez Dashboard.

Depuis le tableau de bord, vous pouvez :

- Préparez votre jeu pour le lancement en choisissant Préparer le lancement et en remplissant le questionnaire de lancement correspondant.
- Demandez des augmentations de quotas de service en vue des lancements ou en réponse aux lancements en choisissant Afficher les quotas de service.
- Consultez les articles de blog et les informations détaillées sur les nouvelles fonctionnalités en cliquant sur le lien dans la section Fonctionnalités.

Le serveur de jeu est intégré au Amazon GameLift Servers tableau de bord

Sur la page Builds du [Amazon GameLift Servers console](#), vous pouvez consulter et gérer toutes les versions de serveur de jeu que vous avez téléchargées sur Amazon GameLift Servers pour un déploiement sur des EC2 flottes gérées. Dans le volet de navigation, choisissez Hosting, Managed EC2, Builds.

La page Builds affiche les informations suivantes pour chaque build. Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir l'

icône

dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.

Note

La page Builds affiche uniquement les builds de votre AWS région actuelle.

- Nom — Le nom associé à la version téléchargée.
- État : statut de la version. Affiche l'un des trois messages d'état suivants :
 - Initialisé : le téléchargement n'a pas commencé ou est toujours en cours.
 - Prêt — La version est prête pour la création d'une flotte.
 - Échec : le délai de compilation a expiré avant Amazon GameLift Servers a reçu les fichiers binaires.
- Heure de création : date et heure auxquelles vous avez chargé le build Amazon GameLift Servers.

- ID de build — L'identifiant unique attribué au build lors du téléchargement.
- Version : étiquette de version associée à la version téléchargée.
- Système d'exploitation : système d'exploitation sur lequel s'exécute le build. Le système d'exploitation de compilation détermine quel système d'exploitation Amazon GameLift Servers s'installe sur les instances d'une flotte.
- Taille — La taille, en mégaoctets (Mo), du fichier de compilation chargé sur Amazon GameLift Servers.
- Flottes : nombre de flottes déployées avec le build.

À partir de cette page, vous pouvez effectuer l'une des opérations suivantes :

- Consulter les détails d'une version de génération. Choisissez le nom d'un build pour ouvrir sa page de détails de build.
- Créer une nouvelle flotte à partir d'une version de génération. Sélectionnez un build, puis choisissez Create fleet.
- Filtrer et trier la liste de versions de génération. Utiliser les commandes situées dans la partie supérieure tableau.
- Supprimer une build. Sélectionnez une version, puis choisissez Supprimer.

Détails de la construction

Sur la page Builds, choisissez le nom d'un build pour ouvrir sa page de détails. La section Vue d'ensemble de la page de détails affiche les mêmes informations récapitulatives sur les builds que sur la page Builds. La section Flottes affiche une liste des flottes créées avec le build, y compris les mêmes informations récapitulatives que celles de la page [Flottes](#).

Amazon GameLift ServersRealtimescripts dans la Amazon GameLift Servers console

Sur la page Scripts de la [Amazon GameLift Serversconsole](#), vous pouvez consulter les informations relatives à tous les Amazon GameLift Servers Realtime scripts que vous avez chargés et les gérer Amazon GameLift Servers pour les déployer sur EC2 des flottes gérées. Dans le volet de navigation, choisissez Hosting, Scripts.

La page Scripts affiche les informations suivantes pour chaque script. Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir

l'icône dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.

Note

La page Scripts affiche uniquement les scripts de votre AWS région actuelle.

- Nom : nom associé au script chargé.
- ID — L'identifiant unique attribué au script lors du téléchargement.
- Version : étiquette de version associée au script chargé.
- Taille : taille, en mégaoctets (Mo), du fichier de script chargé vers Amazon GameLift Servers
- Heure de création : date et heure auxquelles vous avez chargé le script Amazon GameLift Servers.
- Flottes : nombre de flottes déployées avec le script.

À partir de cette page, vous pouvez effectuer l'une des opérations suivantes :

- Afficher les détails du script. Choisissez le nom d'une version pour ouvrir la page de détails de son script.
- Créez une nouvelle flotte à partir d'un script. Sélectionnez un script, puis choisissez Create fleet.
- Filtrez et trie la liste des scripts. Utiliser les commandes situées dans la partie supérieure tableau.
- Supprimez un script. Sélectionnez un script, puis choisissez Supprimer.

Détails du script

Sur la page Scripts, choisissez le nom d'un script pour ouvrir sa page de détails. La section Vue d'ensemble de la page de détails affiche les mêmes informations de synthèse du script que la page Builds. La section Flottes affiche la liste des flottes créées avec le script, y compris les mêmes informations récapitulatives que celles de la page [Flottes](#).

Flottes dans la console Amazon GameLift Servers

Vous pouvez consulter les informations relatives à toutes les flottes créées pour héberger vos jeux sur Amazon GameLift Servers votre AWS compte. La liste indique les flottes créées dans votre région

actuelle. Sur la page Flottes, vous pouvez créer une nouvelle flotte ou consulter des informations supplémentaires sur une flotte. La [page détaillée](#) d'une flotte contient des informations d'utilisation, des statistiques, des données de session de jeu et des données de session de joueur. Vous pouvez également modifier un enregistrement de flotte ou supprimer une flotte.

Pour afficher la page Flottes, choisissez Fleets dans le volet de navigation.

La page Flottes affiche les informations récapitulatives suivantes par défaut : Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir l'icône dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.

- ID — Identifiant attribué à la flotte. Cet identifiant est unique au sein de l' Région AWS endroit où la flotte est créée.
- Nom : nom convivial donné à la flotte.
- État : statut de la flotte : nouvelle, en cours de téléchargement, en construction et active.
- Heure de création : date et heure de création de la flotte.

Note

Une flotte affiche une icône d'avertissement pour les flottes créées il y a plus de 90 jours. Il est recommandé de remplacer les flottes tous les 30 jours afin de garantir un environnement up-to-date d'exécution sécurisé pour vos serveurs de jeux hébergés. Pour de plus amples informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

- Type de flotte : disponibilité des instances utilisées pour héberger vos jeux, ce qui peut avoir un impact sur les coûts d'hébergement. Une flotte peut utiliser des instances à la demande (toujours disponibles) ou ponctuelles (la disponibilité varie).
- Instances actives : nombre d' EC2 instances utilisées par le parc.
- Instances souhaitées : nombre d' EC2 instances à maintenir actives.
- Sessions de jeu : nombre de sessions de jeu actives en cours dans la flotte. Les données sont retardées de cinq minutes.
- Sessions de joueurs : nombre de sessions de joueurs actives dans la flotte. Les données sont retardées de cinq minutes.

Détails du parc dans la Amazon GameLift Servers console

Accédez à la page détaillée d'une flotte depuis le tableau de bord ou la page Flottes en choisissant le nom de la flotte.

Sur la page des détails de la flotte, vous pouvez effectuer les actions suivantes :

- Mettez à jour les attributs, les paramètres de port et la configuration d'exécution d'une flotte.
- Ajoutez ou supprimez des emplacements de flotte.
- Modifier les paramètres de capacité de la flotte.
- Définissez ou modifiez l'auto-scaling du suivi des cibles.
- Supprimer une flotte.

Détails

Réglages du parc

- Numéro de flotte : identifiant attribué à la flotte. Cet identifiant est unique au sein de l' Région AWS endroit où la flotte est créée.
- Nom de la flotte : nom convivial donné à la flotte.
- ARN — Identifiant unique attribué à cette flotte. L'ARN d'une flotte l'identifie comme une Amazon GameLift Servers ressource et indique la région et le AWS compte.
- Description — Une brève description identifiable de la flotte.
- État : état actuel de la flotte, qui peut être nouveau, en cours de téléchargement, en construction ou actif.
- Heure de création : date et heure de création de la flotte.

Note

Une flotte affiche une icône d'avertissement pour les flottes créées il y a plus de 90 jours. À titre de bonne pratique, nous vous recommandons de remplacer les flottes tous les 30 jours afin de garantir un environnement d' up-to-date exécution sécurisé pour vos serveurs de jeux hébergés. Pour de plus amples informations, consultez [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#).

- Type de calcul : type de calcul utilisé pour héberger vos jeux. Une flotte peut être une flotte gérée EC2, une flotte de conteneurs gérés ou une flotte n'importe où.
- Système d'exploitation : Amazon Machine Image (AMI) qui est déployée sur chaque instance du parc. La version AMI est la dernière version disponible au moment de la création de la flotte.
- Type de certificat : état de la capacité de la flotte à utiliser un certificat TLS pour authentifier un serveur de jeu et chiffrer toutes les communications. client/serveur
- Type de flotte : disponibilité des instances utilisées pour héberger vos jeux, ce qui peut avoir un impact sur les coûts d'hébergement. Une flotte peut utiliser des instances à la demande (toujours disponibles) ou ponctuelles (la disponibilité varie).
- EC2 type d'instance : [type d' EC2 instance](#) Amazon sélectionné pour le parc lors de sa création.
- ARN du rôle d'instance : rôle AWS IAM qui gère l'accès à vos autres AWS ressources, si une telle ressource a été fournie lors de la création de la flotte.
- Fournisseur d'informations d'identification du rôle d'instance : rôle AWS IAM qui gère l'accès à vos autres AWS ressources, si un tel rôle a été fourni lors de la création de la flotte.
- Nom du groupe de mesures : groupe utilisé pour agréger les métriques de plusieurs flottes.
- Politique de protection : paramètre actuel de protection des [sessions de jeu de la](#) flotte.
- Nombre maximum de sessions de jeu par joueur : nombre maximum de sessions qu'un joueur peut créer pendant la période de politique.
- Période politique — Durée utilisée pour limiter le nombre de sessions créées par un joueur.

Détails de la construction

La section Détails de la construction affiche la version hébergée dans la flotte. Sélectionnez le nom de la version pour voir la page détaillée complète de la version.

Configuration d'exécution

La section Configuration du runtime affiche les processus du serveur à lancer sur chaque instance. Il inclut le chemin d'accès à l'exécutable du serveur de jeux et des paramètres de lancement en option.

Activation de session de jeu

La section d'activation de la session de jeu indique le nombre de processus du serveur qui se lancent en même temps et le temps qu'il faut attendre pour que le processus soit activé avant de le terminer.

EC2 paramètres du port

La section Ports affiche les autorisations de connexion de la flotte, y compris l'adresse IP et les plages de paramètres de port.

Métriques

L'onglet Métriques affiche une représentation graphique des métriques d'une flotte dans le temps. Pour plus d'informations sur l'utilisation des métriques dans Amazon GameLift Servers, consultez [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#).

Événements

L'onglet Events comporte un journal de tous les événements qui se sont produits sur la flotte, avec le code d'événement, le message et l'horodatage. Consultez les descriptions des [événements](#) dans la référence de Amazon GameLift Servers l'API.

Mise à l'échelle

L'onglet Mise à l'échelle contient des informations sur la capacité du parc, notamment l'état actuel et l'évolution de la capacité au fil du temps. Il propose aussi des outils pour la mise à jour des limites de capacité et la gestion de la scalabilité automatique.

Capacité d'évolutivité

Consultez les paramètres actuels de capacité du parc pour chaque emplacement du parc. Pour plus d'informations sur la modification des limites et de la capacité, consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).

- AWS Emplacement : nom de l'emplacement où les instances de flotte sont déployées.
- État — État de l'hébergement de l'emplacement de la flotte. Le statut de l'emplacement doit ACTIVE être le même pour pouvoir héberger des jeux.
- Taille minimale : le plus petit nombre d'instances devant être déployées sur place.
- Instances souhaitées : nombre cible d'instances actives pour conserver l'emplacement. Lorsque les instances actives et les instances souhaitées ne sont pas identiques, un événement de dimensionnement est lancé pour démarrer ou arrêter les instances selon les besoins jusqu'à ce que les instances actives soient égales aux instances souhaitées.
- Taille maximale : nombre maximal d'instances pouvant être déployées sur place.
- Disponible : limite de service pour les instances moins le nombre d'instances utilisées. Cette valeur indique le nombre maximum d'instances que vous pouvez ajouter à l'emplacement.

Politiques autoscaling

Cette section contient des informations sur les politiques d'auto-scaling appliquées à la flotte. Vous pouvez configurer ou mettre à jour une politique basée sur des cibles. Les politiques basées sur des règles de la flotte, qui doivent être définies à l'aide du AWS SDK ou de la CLI, sont affichées ici. Pour plus d'informations sur la mise à l'échelle, consultez [Augmentez automatiquement la capacité de votre flotte avec Amazon GameLift Servers](#).

Historique du dimensionnement

Consultez des graphiques de l'évolution de la capacité au fil du temps.

Emplacements

L'onglet Emplacements répertorie tous les emplacements où les instances de flotte sont déployées. Les emplacements incluent la région d'origine de la flotte et tous les sites distants qui ont été ajoutés. Vous pouvez ajouter ou supprimer des emplacements directement dans cet onglet.

- **Emplacement** : nom de l'emplacement où les instances de flotte sont déployées.
- **État** — État de l'hébergement de l'emplacement de la flotte. L'état de l'emplacement suit le processus d'activation des premières instances sur le site. Le statut de l'emplacement doit **ACTIVE** être le même pour pouvoir héberger des jeux.
- **Instances actives** : nombre d'instances avec des processus serveur exécutés sur l'emplacement du parc.
- **Serveurs actifs** : nombre de processus de serveur de jeu capables d'héberger des sessions de jeu dans l'emplacement de la flotte.
- **Sessions de jeu** : nombre de sessions de jeu actives sur les instances situées dans l'emplacement de la flotte.
- **Sessions de joueurs** : nombre de sessions de joueurs, qui représentent des joueurs individuels, participant à des sessions de jeu actives dans l'emplacement de la flotte.

Sessions de jeu

L'onglet Game sessions affiche la liste des sessions de jeu passées et présentes hébergées dans la flotte, y compris certaines informations détaillées. Choisissez un identifiant de session de jeu pour accéder à des informations supplémentaires sur les sessions de jeu, y compris les sessions des joueurs. Pour plus d'informations sur les sessions des joueurs, consultez [Sessions de jeu et de jeu dans la Amazon GameLift Servers console](#).

Sessions de jeu et de jeu dans la Amazon GameLift Servers console

Vous pouvez utiliser la Amazon GameLift Servers console pour travailler avec des sessions de jeu et des sessions de joueurs. Pour plus d'informations sur les sessions de jeu et les sessions de joueur, consultez [Comment les joueurs se connectent aux jeux](#). La Amazon GameLift Servers console fournit des informations et des outils pour vous aider à résoudre les problèmes liés à vos sessions de jeu.

Ce que tu peux faire :

- Explorez les sessions de jeu et les activités des sessions des joueurs hébergées sur une flotte spécifique.
- Recherchez l'activité des sessions de jeu d'un joueur spécifique dans plusieurs flottes.
- Arrêtez une session de jeu spécifique.

Afficher le détail de la session de jeu

Les données de session de jeu et de session de joueur sont organisées par la flotte qui héberge la session de jeu.

Pour accéder aux informations relatives à la session de jeu et à la session du joueur

1. Dans la [Amazon GameLift Servers console](#), ouvrez le volet de navigation de gauche. Sélectionnez un type de solution d'hébergement et ouvrez la page Flottes. Par exemple :
 - Hébergement, n'importe où, flottes
 - Hébergement, gestion EC2, flottes
 - Hébergement, conteneurs gérés, flottes
2. Chaque page Flottes affiche la liste des flottes actuellement sélectionnées. Région AWS Choisissez la flotte pour laquelle vous souhaitez consulter les données de session de jeu.
3. Sur la page détaillée de la flotte, ouvrez l'onglet Sessions de jeu. Cet onglet répertorie toutes les sessions de jeu hébergées sur la flotte, ainsi que des informations récapitulatives. Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir l'icône dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.
4. Choisissez une session de jeu dans la liste pour afficher des informations supplémentaires.

5. Si la session de jeu inclut les données de session des joueurs, choisissez Afficher les sessions des joueurs pour ouvrir l'outil de recherche des sessions des joueurs avec l'identifiant de session de jeu automatiquement renseigné.

Le détail des sessions de jeu inclut les informations suivantes :

- État — État de la session de jeu.
 - Activation : l'instance lance une session de jeu.
 - Active — Une session de jeu est en cours et peut accueillir des joueurs, en fonction de la [politique de création de joueurs](#) de la session.
 - Terminé : la session de jeu est terminée.
- ARN — Le nom de ressource Amazon de la session de jeu.
- Nom : nom généré pour la session de jeu.
- Emplacement : emplacement dans lequel la session de jeu a été Amazon GameLift Servers hébergée.
- Heure de création : date et heure de Amazon GameLift Servers création de la session de diffusion.
- Heure de fin : date et heure de fin de la session de jeu.
- Nom DNS : nom d'hôte de la session de jeu.
- Adresse IP : adresse IP spécifiée pour la session de jeu.
- Port : numéro de port utilisé pour se connecter à la session de jeu.
- Creator ID : identifiant unique du joueur qui a lancé la session de jeu.
- Politique de création de session de joueur — Indique si la session de jeu accepte de nouveaux joueurs.
- Politique de protection en matière de scalabilité du jeu : type de protection de session de jeu à définir pour toutes les nouvelles Amazon GameLift Servers instances créées dans le parc.

Données du jeu

Données relatives aux propriétés du jeu, sous forme de chaîne, à envoyer à votre session de jeu au démarrage.

Propriétés du jeu

Données relatives aux propriétés du jeu, formatées key/value par paires, à envoyer à votre session de jeu au démarrage.

Données de matchmaking

Si la session de jeu a été créée avec FlexMatch, les données de matchmaking décrivent des informations sur la configuration du matchmaking et l'ensemble de règles. Cela inclut les attributs des joueurs et les affectations des équipes pour chaque match. Les données sont au format JSON. Pour plus d'informations sur le FlexMatch matchmaking, voir [Créer un entremetteur](#).

Rechercher les données de session du joueur

Si votre solution d'hébergement de jeux utilise des sessions de jeu et propose un joueur unique IDs, vous pouvez explorer les activités spécifiques aux joueurs pour les sessions de jeu passées ou présentes sur plusieurs flottes. Ouvrez l'outil de recherche de session du lecteur en utilisant l'une des méthodes suivantes :

- Dans la Amazon GameLift Servers console, ouvrez le volet de navigation de gauche, choisissez Recherche de session de lecteur et sélectionnez le type de filtre à utiliser.
- Lorsque vous consultez les détails des sessions de jeu d'une flotte, choisissez Afficher les sessions des joueurs. L'outil de recherche s'ouvre avec la session de jeu avec le filtre d'ID de session de jeu présélectionné et la valeur de la session de jeu renseignée.

Lorsque vous utilisez l'outil de recherche, vous pouvez fournir les informations suivantes :

- Un identifiant de session de joueur pour obtenir des informations sur une session de joueur spécifique.
- Un identifiant de session de jeu permettant d'obtenir des informations sur toutes les sessions des joueurs pour la session de jeu demandée. Les résultats représentent tous les joueurs qui ont réservé une place ou qui se sont connectés à la session de jeu. Vous pouvez éventuellement filtrer les résultats en fonction de l'état de la session du joueur.
- Un identifiant de joueur pour obtenir des informations sur toutes les sessions de joueur pour le joueur demandé. Les résultats représentent toutes les sessions de jeu auxquelles le joueur a participé.

Note

L'outil de recherche recherche toutes les activités des sessions des joueurs dans la zone actuellement sélectionnée Région AWS. Si vous avez plusieurs flottes dans la Région, les résultats incluent l'activité des sessions des joueurs dans toutes les flottes. Pour les flottes

multisites, les résultats incluent également l'activité des sessions des joueurs sur les sites distants des flottes.

Les données de session de joueur suivantes sont collectées pour chaque session de jeu :

- ID de session du joueur : identifiant attribué à la session du joueur.
- ID du joueur — Identifiant unique du joueur. Choisissez cet identifiant pour obtenir des informations supplémentaires sur le joueur.
- ID de session de jeu : identifiant attribué à la session de jeu.
- ID de flotte : identifiant attribué à la flotte qui a accueilli la session de jeu.
- État : statut de la session du joueur. Les statuts possibles sont les suivants :
 - Réservé — La session du joueur a été réservée, mais les joueurs ne sont pas connectés.
 - Active — La session du joueur est connectée au serveur de jeu.
 - Terminé — La session du joueur est terminée ; le joueur n'est plus connecté.
 - Expiration du délai — Le joueur n'a pas réussi à se connecter.
- Heure de création : heure à laquelle le joueur s'est connecté à la session de jeu.
- Heure de fin : heure à laquelle le joueur s'est déconnecté de la session de jeu.
- Données de connexion : adresse IP, nom DNS et port utilisés par le joueur pour se connecter à la session de jeu.
- Données du joueur : informations sur le joueur fournies lors de la création de la session de joueur.

Arrêter une session de jeu

Utilisez la Amazon GameLift Servers console pour arrêter une session de jeu spécifique. Cette fonctionnalité vous offre une méthode simple et rapide pour localiser une session de jeu et envoyer un signal pour y mettre fin. Une autre méthode de résiliation nécessite que vous trouviez l'instance de flotte sur laquelle la session de jeu est en cours d'exécution, que vous y accédiez à distance et que vous arrêtiez manuellement la session de jeu.

Vous pouvez arrêter une session de jeu pour n'importe quelle raison. La raison la plus courante est de résoudre une session de jeu qui ne s'arrête pas naturellement. Par conséquent, la ressource d'hébergement de la session de jeu ne peut pas être libérée pour héberger une nouvelle session de jeu, et la capacité d'hébergement de la flotte est dégradée.

 Note

Cette fonctionnalité repose sur certains paramètres de configuration de votre solution d'hébergement. Il présente les limites suivantes :

- La session de jeu doit être hébergée sur une flotte exécutant un serveur de jeu compilé avec le SDK du serveur pour la version Amazon GameLift Servers 5 ou une version ultérieure. Si vos serveurs de jeu sont déployés avec une ancienne version, vous devez utiliser l'accès à distance pour supprimer la session de jeu.
- Si la session de jeu est hébergée sur une flotte Anywhere, la flotte doit utiliser l'Amazon GameLift Serversagent pour gérer les processus du serveur de jeu.

Pour mettre fin à une session de jeu

1. Dans la [Amazon GameLift Serversconsole](#), ouvrez le volet de navigation de gauche. Sélectionnez un type de solution d'hébergement et ouvrez la page Flottes. Par exemple :
 - Hébergement, n'importe où, flottes
 - Hébergement, gestion EC2, flottes
 - Hébergement, conteneurs gérés, flottes
2. Chaque page Flottes affiche la liste des flottes actuellement sélectionnées. Région AWS Choisissez la flotte qui héberge la session de jeu à laquelle vous souhaitez mettre fin.
3. Sur la page détaillée de la flotte, ouvrez l'onglet Sessions de jeu. Dans la liste des sessions de jeu, sélectionnez celle à laquelle vous souhaitez mettre fin, puis cliquez sur le bouton Terminer.
4. Dans la session de jeu Terminate ? fenêtre, vérifiez que vous fermez la bonne session de jeu et choisissez une méthode de résiliation.
 - Arrêt normal de la session de jeu : cette option envoie un signal au processus serveur hébergeant la session de jeu pour qu'il s'arrête. Si la version de votre serveur de jeu a été correctement intégréeAmazon GameLift Servers, le processus du serveur lance la séquence d'arrêt de sa session de jeu, indique Amazon GameLift Servers qu'elle se termine et s'arrête. En fonction de la conception de votre jeu, la séquence d'arrêt peut inclure des étapes permettant de terminer la session de jeu en douceur, telles que la sauvegarde des données et la notification des joueurs actifs. Cette méthode peut nécessiter un petit délai pour terminer la séquence d'arrêt de la session de jeu.

- Arrêt immédiat de la session de jeu : cette option envoie un signal à un gestionnaire de processus pour arrêter le processus du serveur qui héberge la session de jeu. Cette option permet de contourner l'arrêt normal de la session de jeu. Il est capable de mettre fin à la session de jeu même lorsque le processus du serveur est incapable de répondre.
5. Confirmez la fin de la session de jeu. Vous pouvez suivre la progression de l'arrêt sur la page de console des sessions de jeu. Le statut de la session de jeu passe à « Terminé », puis à « Terminé » une fois l'arrêt terminé.

Voir aussi

- Vous pouvez également arrêter les sessions de jeu à l'aide du AWS SDK et du AWS CLI. Pour plus de détails et d'exemples, consultez la rubrique de référence des Amazon GameLift Servers API [TerminateGameSession](#).
- Pour plus d'informations sur l'intégration d'un serveur de jeu et sur la façon dont un processus serveur répond aux signaux du Amazon GameLift Servers service, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Alias dans le Amazon GameLift Servers console

La page Alias affiche des informations sur Amazon GameLift Servers alias qui dirigent le trafic vers des destinations d'hébergement spécifiques. Pour les alias, choisissez Hosting, Aliases dans le volet de navigation.

Vous pouvez effectuer les opérations suivantes sur la page des alias :

- Créez un nouvel alias. Choisissez Créer un alias.
- Filtrez et triez le tableau des alias. Utiliser les commandes situées dans la partie supérieure tableau. Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir l'icône dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.
- Afficher des informations sur les alias. Choisissez un nom d'alias pour ouvrir la page détaillée de l'alias.
- Supprimer un alias. Choisissez un alias, puis cliquez sur Supprimer.

Détails de l'alias

La page de détails de l'alias affiche des informations sur l'alias.

À partir de cette page, vous pouvez :

- Modifiez un alias. Choisissez Modifier.
- Affichez les flottes que vous avez associées à l'alias.
- Supprimer un alias. Sélectionnez Delete (Supprimer).

Les informations détaillées d'alias sont les suivantes :

- ID — Le numéro unique utilisé pour identifier l'alias.
- Description — Description de l'alias.
- ARN — Le nom de ressource Amazon de l'alias.
- Création : date et heure de création de l'alias.
- Dernière mise à jour : date et heure de dernière mise à jour de l'alias.
- Type de routage : type de routage pour l'alias, qui peut être l'un des suivants :
 - Simple : achemine le trafic des joueurs vers un numéro de flotte spécifique. Vous pouvez mettre à jour l'ID de flotte d'un alias à tout moment.
 - Terminal : renvoie un message au client. Par exemple, vous pouvez diriger les joueurs qui utilisent un out-of-date client vers un endroit où ils peuvent obtenir une mise à niveau.
- Tags — Paires de clés et de valeurs utilisées pour identifier l'alias.

files d'attente pour les sessions de jeu dans le Amazon GameLift Servers console

Vous pouvez consulter les informations relatives à toutes les files d'attente utilisées pour traiter les demandes de nouvelles sessions de jeu. La page des files d'attente affiche les files d'attente des sessions de jeu actuellement sélectionnées. Région AWS Depuis la page Queues, vous pouvez créer une file d'attente, supprimer les files d'attente existantes ou ouvrir une page d'informations détaillées pour une file d'attente sélectionnée. Chaque page de détails de file d'attente contient les données de configuration et de métriques de la file d'attente. Pour plus d'informations sur les files d'attente, consultez [Gestion du placement des sessions de jeu avec des Amazon GameLift Servers files d'attente](#).

La page des files d'attente affiche les informations récapitulatives suivantes pour chaque file d'attente. Vous pouvez ajuster le contenu du tableau selon vos besoins à l'aide de l'outil Préférences (voir l'icône

dans le coin supérieur droit du tableau). Les préférences personnalisées sont enregistrées auprès de l'utilisateur de votre AWS compte et sont automatiquement appliquées chaque fois que vous consultez cette page.

- **Nom de la file d'attente** : nom attribué à la file d'attente. Les demandes de nouvelles sessions de jeu utilisent ce nom pour spécifier une file d'attente.
- **Délai d'attente** : durée maximale, en secondes, pendant laquelle une demande de placement de session de jeu reste dans la file d'attente avant l'expiration du délai imparti.
- **Destinations en file d'attente** : nombre de flottes répertoriées dans la configuration de la file d'attente. Amazon GameLift Servers place de nouvelles sessions de jeu sur n'importe quelle flotte de la file d'attente.

Afficher les détails de la file

Vous pouvez accéder aux informations détaillées relatives à n'importe quelle file d'attente, notamment sur la configuration et les métriques de la file d'attente. Pour ouvrir une page de détails sur une file d'attente, rendez-vous sur la page Files d'attente et choisissez un nom de file d'attente.

La page des détails d'une file d'attente affiche un tableau récapitulatif et des onglets comportant des informations supplémentaires. Sur cette page, vous pouvez effectuer les opérations suivantes :

- Mettre à jour la configuration de la file d'attente, la liste des destinations et les stratégies de latence pour les joueurs. Choisissez Modifier.
- Supprimez une file d'attente. Une fois que vous avez supprimé une file d'attente, toutes les demandes de nouvelles sessions de jeu faisant référence à ce nom de file d'attente échoueront. Sélectionnez Delete (Supprimer).

Note

Pour restaurer une file d'attente supprimée, créez-en une nouvelle avec le nom de la file supprimée.

Détails

Présentation

La section Vue d'ensemble affiche le nom de ressource Amazon (ARN) de la file d'attente et le délai d'expiration. Vous pouvez utiliser l'ARN pour référencer la file d'attente dans d'autres actions ou domaines de Amazon GameLift Servers. Le délai d'attente est la durée maximale, en secondes, pendant laquelle une demande de placement de session de jeu reste dans la file d'attente avant son expiration.

Notification d'événement

La section Notification d'événements répertorie la rubrique SNS. Amazon GameLift Servers publie les notifications d'événements et les données d'événement ajoutées à tous les événements créés par cette file d'attente.

Balises

Le tableau Tags affiche les clés et les valeurs utilisées pour étiqueter la ressource. Pour plus d'informations sur le balisage, consultez la section Ressources relatives au [balisage. AWS](#)

Métriques

L'onglet Metrics affiche une représentation graphique des métriques d'une file d'attente au fil du temps.

Les statistiques relatives aux files d'attente incluent une série d'informations décrivant les activités de placement dans la file d'attente, y compris les placements réussis organisés par région. Vous pouvez utiliser les données de région pour comprendre où vous hébergez vos jeux. Les indicateurs de placement régionaux peuvent aider à détecter les problèmes liés à la conception globale de la file d'attente.

Les statistiques de file d'attente sont également disponibles sur Amazon CloudWatch. Pour une description des mesures disponibles, voir [Métriques Amazon GameLift Servers pour les files d'attente](#).

Destinations

L'onglet Destinations affiche toutes les flottes ou tous les alias répertoriés pour la file d'attente.

Lorsque Amazon GameLift Servers recherche dans les destinations les ressources disponibles pour héberger une nouvelle session de jeu, il recherche l'ordre par défaut indiqué ici. Tant qu'il

y a de la capacité sur la première destination indiquée, Amazon GameLift Servers y place de nouvelles sessions de jeu. Vous pouvez faire en sorte que les demandes de placement de session de jeu remplacent l'ordre par défaut en fournissant les données de latence du joueur. Ces données indiquent Amazon GameLift Servers pour rechercher une destination disponible présentant le temps de latence moyen le plus faible pour les joueurs. Pour plus d'informations sur la conception de vos files d'attente, consultez [Personnaliser une file d'attente de session de jeu](#).

Placement des sessions

Politiques de latence des joueurs

La section Politiques de latence des joueurs affiche toutes les politiques utilisées par la file d'attente. Les tableaux répertorient les politiques dans l'ordre dans lequel elles sont appliquées.

Emplacements

La section Emplacements indique les emplacements dans lesquels cette file d'attente peut placer une session de jeu.

Priorité

La section Priorité indique l'ordre dans lequel la file d'attente évalue les détails d'une session de jeu.

Ordre de localisation

La section Ordre de localisation indique l'ordre par défaut utilisé par la file d'attente pour placer des sessions de jeu. La file d'attente utilise cet ordre si vous n'avez pas défini d'autres types de priorité.

Surveillez Amazon GameLift Servers avec Amazon CloudWatch

Vous pouvez effectuer un suivi Amazon GameLift Servers à l'aide d'Amazon CloudWatch, un AWS service qui collecte des données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques sont conservées pendant 15 mois pour fournir une perspective historique sur la performance de votre serveur de jeux avec Amazon GameLift Servers. Vous pouvez définir des alarmes qui surveillent certains seuils et envoient des notifications ou effectuent des actions spécifiques lorsque ces seuils sont atteints. Pour plus d'informations, consultez le [guide de CloudWatch l'utilisateur Amazon](#).

Les tableaux suivants répertorient les métriques et les dimensions d'Amazon GameLift Servers. Toutes les métriques disponibles dans le CloudWatch sont également dans la Amazon

GameLift Servers console, qui fournit les données sous la forme d'un ensemble de graphiques personnalisables. Pour accéder aux CloudWatch statistiques de vos jeux, utilisez l'API AWS Management Console AWS CLI, le ou l' CloudWatch API.

Si une métrique n'a pas d'emplacement, elle utilise l'emplacement d'origine.

Dimensions pour les métriques Amazon GameLift Servers

Amazon GameLift Servers prend en charge le filtrage des métriques selon les dimensions suivantes.

Dimension	Description
Location	Filtrez les métriques pour un lieu de déploiement de flotte. Si une métrique n'a pas d'emplacement, elle utilise l'emplacement d'origine.
FleetId	Filtrez les métriques d'une seule flotte. Cette dimension est utilisée avec toutes les métriques de flotte relatives aux instances, aux processus serveur, aux sessions de jeu et aux sessions de joueur.
MetricGroup	Filtrez les métriques pour une collection de flottes. Ajoutez une flotte à un groupe de métriques en ajoutant le nom du groupe de métriques aux attributs de la flotte (voir UpdateFleetAttributes()). Cette dimension est utilisée avec toutes les métriques de flotte relatives aux instances, aux processus serveur, aux sessions de jeu et aux sessions de joueur.
QueueName	Filtrez les métriques pour une seule file d'attente . Cette dimension est utilisée avec les métriques relatives aux files d'attente de placement de session de jeu uniquement.
ConfigurationName	Filtrez les métriques pour une configuration de mise en relation unique. Cette dimension est utilisée avec les métriques pour les configurations de mise en relation.

Dimension	Description
ConfigurationName-RuleName	Filtrez les métriques pour obtenir l'intersection d'une configuration de mise en relation et d'une règle de mise en relation. Cette dimension est utilisée avec les métriques pour les règles de mise en relation uniquement.
InstanceType	Filtrez les métriques pour une désignation de type d'EC2 instance, telle que « c4.large ». Cette dimension est utilisée avec les métriques pour les instances Spot.
OperatingSystem	Filtrez les métriques du système d'exploitation d'une instance Cette dimension est utilisée avec les métriques des instances Spot.
GameServerGroup	Filtrer les métriques FleetIQ pour un groupe de serveurs de jeu.
ContainerGroupType	Filtrez les indicateurs du parc de conteneurs pour faire la distinction entre GameServer Container Group les indicateurs relatifs à l'emplacement du parc et PerInstance ContainerGroup ceux relatifs à l'emplacement du parc. Cette dimension est incluse pour certaines métriques du parc de conteneurs (telles que ContainerNetworkIn ,Container StorageReadBytes) et vous permet de faire la distinction entre les métriques pour différents types de groupes de conteneurs.

Métriques Amazon GameLift Servers pour les flottes

L'espace de nom `AWS/GameLift` inclut les métriques suivantes associées à une activité sur un parc ou un groupe de parcs. Les flottes sont utilisées avec une solution Amazon GameLift Servers gérée. Le Amazon GameLift Servers service envoie des métriques à CloudWatch chaque minute.

instances

Métrique	Description
ActiveInstances	Instances avec l'état ACTIVE, ce qui signifie qu'elles exécutent des processus serveur actifs. Le compte inclut des instances inactives et celles qui hébergent une ou plusieurs sessions de jeu. Cette métrique mesure la capacité totale actuelle de l'instance. Cette métrique peut être utilisée avec le dimensionnement automatique. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
DesiredInstances	Nombre cible d'instances actives gérées par Amazon GameLift Servers dans la flotte. Grâce au dimensionnement automatique, cette valeur est déterminée en fonction des stratégies de dimensionnement en vigueur actuellement. Sans dimensionnement automatique, cette valeur est définie manuellement. Cette métrique n'est pas disponible lorsque vous affichez les données pour les groupes de métriques du parc. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
IdleInstances	Instances actives qui hébergent actuellement zéro (0) session de jeu. Cette métrique mesure la capacité

Métrique	Description
	disponible mais non utilisée. Cette métrique peut être utilisée avec le dimensionnement automatique. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
MaxInstances	Nombre maximal d'instances autorisées pour le parc. Le nombre maximal d'instances d'un parc détermine le plafond de capacité lors du dimensionnement à la hausse manuel ou automatique. Cette métrique n'est pas disponible lorsque vous affichez les données pour les groupes de métriques du parc. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
MinInstances	Nombre minimal d'instances autorisées pour le parc. Le nombre minimal d'instances d'une flotte détermine le plancher de capacité lors de la réduction manuelle ou automatique de la capacité. Cette métrique n'est pas disponible lorsque vous affichez les données pour les groupes de métriques du parc. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement

Métrique	Description
PercentIdleInstances	Pourcentage d'instances actives qui sont inactives (calculé selon la formule $\text{IdleInstances} / \text{ActiveInstances}$). Cette métrique peut être utilisée dans le cadre d'un dimensionnement automatique. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
RecycledSpotInstances	Nombre d'instances ponctuelles qui ont été recyclées et remplacées. Amazon GameLift Servers recycle les instances ponctuelles qui n'hébergent pas actuellement de sessions de jeu et qui présentent une forte probabilité d'interruption. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement
InstanceSpotInterruptions	Nombre d'instances Spot qui ont été interrompues. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement

Métrique	Description
CPUUtilization	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Pourcentage de temps processeur physique EC2 utilisé par Amazon pour exécuter l'instance, qui inclut le temps passé à exécuter à la fois le code utilisateur et le EC2 code Amazon. Les outils de votre système d'exploitation peuvent afficher un pourcentage différent de celui CloudWatch dû à des facteurs tels que la simulation d'appareils existants, la configuration d'appareils non existants, les charges de travail nécessitant de nombreuses interruptions, la migration en direct et la mise à jour en direct. Unités : pourcentage
NetworkIn	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Nombre d'octets reçus par l'instance sur toutes les interfaces réseau. Cette métrique identifie le volume du trafic réseau entrant d'une application sur une seule instance. Unités : octets
NetworkOut	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Nombre d'octets envoyés par l'instance sur toutes les interfaces réseau. Cette métrique identifie le volume du trafic réseau sortant d'une application sur une seule instance. Unités : octets

Métrique	Description
DiskReadBytes	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Octets lus à partir de tous les volumes de stockage d'instance disponibles pour l'instance. Cette métrique permet de déterminer le volume de données que l'application lit à partir du disque dur de l'instance. Vous pouvez l'utiliser pour déterminer la vitesse de l'application. Unités : octets
DiskWriteBytes	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Octets écrits dans tous les volumes de stockage d'instance disponibles pour l'instance. Cette métrique permet de déterminer le volume de données que l'application écrit sur le disque dur de l'instance. Vous pouvez l'utiliser pour déterminer la vitesse de l'application. Unités : octets
DiskReadOps	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Opérations de lecture terminées de tous les volumes de stockage d'instance disponibles pour l'instance, au cours de la période spécifiée. Pour calculer la moyenne I/O des opérations par seconde (IOPS) pour la période, divisez le total des opérations de la période par le nombre de secondes de cette période. Unités : nombre

Métrique	Description
DiskWriteOps	EC2 métrique. Car Amazon GameLift Servers cette métrique représente les performances matérielles de toutes les instances actives d'un parc. Opérations d'écriture terminées dans tous les volumes de stockage d'instance disponibles pour l'instance, au cours de la période spécifiée. Pour calculer la moyenne I/O des opérations par seconde (IOPS) pour la période, divisez le total des opérations de la période par le nombre de secondes de cette période. Unités : nombre

Processus serveur

Métrique	Description
ActiveServerProcesses	Processus serveur avec l'état ACTIVE, ce qui signifie qu'ils fonctionnent et sont en mesure d'héberger des sessions de jeu. Le compte inclut des processus serveur inactifs et ceux qui hébergent des sessions de jeu. Cette métrique mesure la capacité totale actuelle de processus serveur. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
HealthyServerProcesses	Processus serveur actifs signalés comme sains. Cette métrique est utile pour le suivi de la santé générale des serveurs de jeu du parc. Unités : nombre

Métrique	Description
	CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
PercentHealthyServerProcesses	Pourcentage de l'ensemble des processus serveur actifs signalés comme sains (calculé selon la formule $\text{HealthyServerProcesses} / \text{ActiveServerProcesses}$). Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
ServerProcessAbnormalTerminations	Processus serveur ayant été arrêtés en raison de circonstances anormales depuis le dernier rapport. Cette métrique inclut les mises hors service initiées par le service Amazon GameLift Servers. Cela se produit lorsqu'un processus serveur cesse de répondre, signale régulièrement l'échec des tests de santé ou ne s'arrête pas correctement (en appelant ProcessEnding()). Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement

Métrique	Description
ServerProcessActivations	Processus serveur passés avec succès de l'état ACTIVATING à l'état ACTIVE depuis le dernier rapport. Les processus serveur ne peuvent pas héberger de sessions de jeu tant qu'ils ne sont pas actifs. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement
ServerProcessTerminations	Processus serveur arrêtés depuis le dernier rapport. Cela concerne tous les processus serveur qui, pour n'importe quelle raison, ont opéré une transition vers l'état TERMINATED, y compris en cas de mise hors service normale ou anormale du processus. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement

Sessions de jeu

Métrique	Description
ActivatingGameSessions	Sessions de jeu avec l'état ACTIVATING, ce qui signifie qu'elles sont en cours de démarrage. Les sessions de jeu ne peuvent pas héberger des joueurs tant qu'elles ne sont pas actives. Des chiffres élevés pendant une période de temps continue peuvent indiquer que les sessions de jeu

Métrique	Description
	ne parviennent pas à passer de l'état ACTIVATING à l'état ACTIVE. Cette métrique peut être utilisée avec le dimensionnement automatique. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement
ActiveGameSessions	Sessions de jeu avec l'état ACTIVE, ce qui signifie qu'elles sont en mesure d'héberger des joueurs, et qu'elles hébergent actuellement zéro joueur ou plus. Cette métrique mesure le nombre total de sessions de jeu hébergées actuellement. Cette métrique peut être utilisée avec le dimensionnement automatique. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement

Métrique	Description
AvailableGameSessions	Processus serveur actifs et sains qui ne sont pas actuellement utilisés pour héberger une session de jeu et peuvent démarrer une nouvelle session de jeu sans délai pour lancer de nouveaux processus ou instances de serveur. Cette métrique peut être utilisée avec le dimensionnement automatique.  Note Pour les flottes qui limitent les activations de sessions de jeu simultanées, utilisez la métrique. <code>ConcurrentActiveGameSessions</code> Cette métrique représente plus précisément le nombre de nouvelles sessions de jeu pouvant démarrer sans aucun délai. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement

Métrique	Description
ConcurrentActivatableGameSessions	Processus de serveur actifs et sains qui ne sont pas actuellement utilisés pour héberger une session de jeu et peuvent démarrer immédiatement une nouvelle session de jeu. Cette métrique diffère de la <code>AvailableGameSessions</code> manière suivante : elle ne prend pas en compte les processus du serveur qui ne peuvent actuellement pas activer une nouvelle session de jeu en raison des limites d'activation des sessions de jeu. (Voir le paramètre RuntimeConfiguration optionnel de la flotte <code>MaxConcurrentGameSessionActivations</code>). Pour les flottes qui ne limitent pas les activations de sessions de jeu, cette métrique est identique à <code>AvailableGameSessions</code>. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : Emplacement

Métrique	Description
PercentAvailableGameSessions	Pourcentage d'emplacements de session de jeu actifs sur l'ensemble des processus serveur actifs (sains ou non sains) actuellement inutilisés (calculé selon la formule $\text{AvailableGameSessions} / [\text{ActiveGameSessions} + \text{AvailableGameSessions} + \text{unhealthy server processes}]$). Cette métrique peut être utilisée avec le dimensionnement automatique. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne Dimensions : Emplacement
GameSessionInterruptions	Nombre de sessions de jeu ou d'instances Spot qui ont été interrompues. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : Emplacement

Sessions de joueur

Métrique	Description
CurrentPlayerSessions	Sessions de joueur avec soit l'état ACTIVE (le joueur est connecté à une session de jeu active), soit l'état RESERVED (le joueur s'est vu attribuer un emplacement dans une session de jeu mais ne s'est pas encore connecté). Cette métrique peut être utilisée avec le dimensionnement automatique.

Métrique	Description
	Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale
PlayerSessionActivations	Sessions de joueur qui sont passées de l'état RESERVED à l'état ACTIVE depuis le dernier rapport. Cette situation survient lorsqu'un joueur réussit à se connecter à une session de jeu active. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum

Mesures relatives à la flotte de conteneurs

Les métriques de la flotte de conteneurs fournissent des données de surveillance pour l'hébergement de jeux basé sur des conteneurs, notamment le statut des groupes de conteneurs, l'utilisation des ressources et les indicateurs de performance.

Métrique	Description
ActiveGameServerContainerGroups	Groupes de conteneurs actuellement actifs et prêts à accueillir des sessions de jeu. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
IdleGameServerContainerGroups	Groupes de conteneurs actifs mais n'hébergeant pas actuellement de sessions de jeu. Unités : nombre

Métrique	Description
	CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
PendingGameServerContainerGroups	Groupes de conteneurs en cours de démarrage. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
TerminatingGameServerContainerGroups	Groupes de conteneurs en cours de fermeture. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
UnhealthyGameServerContainerGroupsReplaced	Groupes de conteneurs qui ont été remplacés en raison d'échecs liés aux contrôles de santé. Unités : nombre CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : FleetID et localisation

Métrique	Description
ContainerCPUUtilizationPerInstance	Utilisation moyenne du processeur sur tous les conteneurs d'une instance. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
ContainerCPUReservation	Pourcentage de ressources du processeur réservées par les conteneurs. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
ContainerMemoryReservation	Pourcentage de ressources de mémoire réservées par les conteneurs. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation
ContainerMemoryUtilization	Pourcentage de mémoire réservée utilisée par les conteneurs. Unités : pourcentage CloudWatch Statistiques pertinentes : moyenne, minimale, maximale Dimensions : FleetID et localisation

Métrique	Description
ContainerNetworkIn	Octets réseau reçus par les conteneurs. Unités : octets CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : FleetId, Emplacement et Container GroupType
ContainerNetworkOut	Octets réseau envoyés par des conteneurs. Unités : octets CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : FleetId, Emplacement et Container GroupType
ContainerStorageReadBytes	Octets lus depuis le stockage par des conteneurs. Unités : octets CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : FleetId, Emplacement et Container GroupType
ContainerStorageWriteBytes	Octets écrits dans le stockage par des conteneurs. Unités : octets CloudWatch Statistiques pertinentes : somme, moyenne, minimum, maximum Dimensions : FleetId, Emplacement et Container GroupType

Métriques Amazon GameLift Servers pour les files d'attente

L'espace de nom Amazon GameLift inclut les métriques suivantes associées à une activité sur une file d'attente de placement de session de jeu. Les files d'attente sont utilisées avec une solution Amazon GameLift Servers gérée. Le Amazon GameLift Servers service envoie des métriques à CloudWatch chaque minute.

Métrique	Description
AverageWaitTime	Temps d'attente moyen pour satisfaire les demandes de placement de session de jeu dans la file d'attente avec l'état PENDING. Unités : secondes CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme Dimensions : Emplacement
FirstChoiceNotViable	Sessions de jeu correctement placées mais PAS dans la flotte de premier choix, car cette dernière a été considérée comme non viable (comme un parc d'instances Spot avec un taux d'interruption élevé). Cette métrique est basée sur le coût et non sur le temps de latence. La flotte de premier choix est soit la première flotte répertoriée dans la file d'attente, soit, lorsqu'une demande de placement inclut des données de latence des joueurs, c'est la première flotte choisie par ordre de priorité. FleetIQ S'il n'y a pas de parcs d'instances Spot viables, n'importe quel parc de cette région peut être sélectionné. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme

Métrique	Description
FirstChoiceOutOfCapacity	Sessions de jeu correctement placées mais PAS dans la flotte de premier choix, car cette dernière n'avait pas de ressources disponibles. La flotte de premier choix est soit la première flotte répertoriée dans la file d'attente, soit, lorsqu'une demande de placement inclut des données de latence des joueurs, c'est la première flotte choisie selon vos priorités définies. FleetIQ Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
LowestLatencyPlacement	Sessions de jeu correctement placées dans une région qui offre la plus faible latence possible de la file d'attente pour les joueurs. Cette métrique est émis uniquement lorsque les données de latence des joueurs sont incluses dans la requête de placement. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme

Métrique	Description
LowestPricePlacement	Des sessions de jeu qui ont été placées avec succès dans une flotte avec le prix le plus bas possible dans la file d'attente pour la région choisie. Cette flotte peut être un parc d'instances Spot ou une instance à la demande si la file d'attente ne dispose d'aucune instance Spot. Cette métrique est émis uniquement lorsque les données de latence des joueurs sont incluses dans la requête de placement. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
Placement <region name>	Sessions de jeu correctement placées dans des flottes situés dans la région spécifiée. Cette métrique décortique la métrique PlacementsSucceeded par région. Unités : nombre CloudWatchStatistiques pertinentes : Somme
PlacementsCanceled	Demandes de placement de session de jeu ayant été annulées avant expiration depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme

Métrique	Description
PlacementsFailed	Requêtes de placement de session de jeu ayant échoué pour une quelconque raison depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
PlacementsStarted	Nouvelle demandes de placement de session de jeu ayant été ajoutées à la file d'attente depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
PlacementsSucceeded	Demandes de placement de session de jeu ayant abouti au lancement d'une nouvelle session de jeu depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
PlacementsTimedOut	Demandes de placement de session de jeu ayant atteint la limite d'expiration de la file d'attente sans être exécutées depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme

Métrique	Description
QueueDepth	Nombre de demandes de placement de session de jeu dans la file d'attente avec l'état PENDING. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme Dimensions : Emplacement

Métriques Amazon GameLift Servers pour l'activité de mise en relation

L'espace de noms Amazon GameLift Servers inclut des métriques sur l'activité FlexMatch pour les configurations de mise en relation et les règles de mise en relation. La mise en relation FlexMatch est utilisée avec une solution Amazon GameLift Servers gérée. Le Amazon GameLift Servers service envoie des métriques à CloudWatch chaque minute.

Pour plus d'informations sur la séquence des activités de matchmaking, voir [Comment Amazon GameLift Servers FlexMatch fonctionne](#).

Configurations de mise en relation

Métrique	Description
CurrentTickets	Demandes de mise en relation en cours de traitement ou en attente de traitement. Unités : nombre CloudWatch Statistiques pertinentes : moyenne, minimale, maximale, somme
MatchAcceptancesTimedOut	Pour les configurations de mise en relation qui nécessitent l'acceptation, les rencontres éventuelles qui ont dépassé le délai lors de l'acceptation depuis le dernier rapport.

Métrique	Description
	Unités : nombre CloudWatch Statistiques pertinentes : Somme
MatchesAccepted	Pour les configurations de mise en relation qui nécessitent l'acceptation, les rencontres éventuelles qui ont été acceptées depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
MatchesCreated	Mises en relation potentielles ayant été créées depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
MatchesPlaced	Mises en relation placées avec succès dans une session de jeu depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
MatchesRejected	Pour les configurations de mise en relation qui nécessitent l'acceptation, les rencontres éventuelles qui ont été refusées par au moins un joueur depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme

Métrique	Description
PlayersStarted	Joueurs dans des tickets de mise en relation ajoutés depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
TicketsFailed	Demandes de mise en relation ayant entraîné un échec depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
TicketsStarted	Nouvelles demandes de mise en relation ayant été créées depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
TicketsTimedOut	Demandes de mise en relation ayant dépassé le délai depuis le dernier rapport. Unités : nombre CloudWatch Statistiques pertinentes : Somme
TimeToMatch	Pour les demandes de mise en relation qui ont été placées dans une mise en relation potentielle avant le dernier rapport, la quantité de temps entre la création du ticket et la création de la mise en relation potentielle. Unités : secondes CloudWatch Statistiques pertinentes : échantillons de données, moyenne, minimale, maximale

Métrique	Description
TimeToTicketCancel	Pour les demandes de mise en relation qui ont été annulées avant le dernier rapport, la quantité de temps entre la création du ticket et l'annulation. Unités : secondes CloudWatch Statistiques pertinentes : échantillons de données, moyenne, minimale, maximale
TimeToTicketSuccess	Pour les demandes de mise en relation qui ont réussi avant le dernier rapport, la quantité de temps entre la création du ticket et le placement de mise en relation réussi. Unités : secondes CloudWatch Statistiques pertinentes : échantillons de données, moyenne, minimale, maximale

Règles de mise en relation

Métrique	Description
RuleEvaluationsPassed	Évaluations des règles pendant le processus de mise en relation qui ont réussi depuis le dernier rapport. Cette métrique est limitée aux 50 premières règles. Unités : nombre CloudWatchStatistiques pertinentes : Somme
RuleEvaluationsFailed	Évaluations des règles pendant l'activité de mise en relation qui ont échoué depuis le dernier rapport. Cette métrique est limitée aux 50 premières règles. Unités : nombre

Métrique	Description
	CloudWatch Statistiques pertinentes : Somme

Métriques Amazon GameLift Servers pour un FleetIQ

L'espace de noms Amazon GameLift Servers inclut des métriques pour le groupe de serveurs de jeu FleetIQ et son activité dans le cadre d'une solution FleetIQ autonome pour l'hébergement de jeux. Le Amazon GameLift Servers service envoie des métriques à CloudWatch chaque minute. Consultez également la section [Surveillance de vos groupes et instances Auto Scaling à l'aide d'Amazon CloudWatch](#) dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.

Métrique	Description
AvailableGameServers	Serveurs de jeux qui sont disponibles pour exécuter une exécution de jeu et qui ne sont pas actuellement occupés par le jeu. Ce nombre inclut les serveurs de jeux qui ont été demandés, mais qui sont toujours à l'état AVAILABLE (DISPONIBLE). Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup
UtilizedGameServers	Serveurs de jeux qui sont actuellement occupés avec le gameplay. Ce nombre inclut les serveurs de jeux dont l'état est UTILIZED (UTILISÉ). Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup

Métrique	Description
DrainingAvailableGameServers	Serveurs de jeux sur des instances dont la résiliation est planifiée et qui ne prennent actuellement pas en charge le gameplay. En cas de nouvelle demande, la priorité de demande de ces serveurs de jeux est la plus faible. Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup
DrainingUtilizedGameServers	Serveurs de jeux sur des instances dont la résiliation est planifiée et qui prennent actuellement en charge le gameplay. Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup
PercentUtilizedGameServers	Partie des serveurs de jeux qui prennent actuellement en charge les exécutions de jeux. Cette métrique indique la capacité du serveur de jeux actuellement utilisé. Elle est utile pour mener une stratégie Auto Scaling qui peut ajouter et supprimer dynamiquement des instances afin de correspondre à la demande des joueurs. Unités : pourcentage CloudWatch Statistiques Amazon pertinentes : moyenne, minimale, maximale Dimensions : GameServerGroup

Métrique	Description
GameServerInterruptions	Serveurs de jeux sur des instances Spot qui ont été interrompus en raison d'une disponibilité limitée des instances Spot. Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup, InstanceType
InstanceInterruptions	Instances Spot interrompues en raison d'une disponibilité limitée. Unités : nombre CloudWatch Statistiques Amazon pertinentes : somme Dimensions : GameServerGroup, InstanceType

Journalisation Amazon GameLift Servers Appels d'API avec AWS CloudTrail

Amazon GameLift Servers est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon GameLift Servers. CloudTrail capture tous les appels d'API pour Amazon GameLift Servers en tant qu'événements. Les appels capturés incluent les appels provenant du Amazon GameLift Servers appels de console et de code vers Amazon GameLift Servers Opérations d'API. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Amazon GameLift Servers. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite Amazon GameLift Servers, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le [guide de AWS CloudTrail l'utilisateur](#).

Amazon GameLift Servers informations dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsque l'activité se produit dans Amazon GameLift Servers, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section [Affichage des événements à l'aide de l'historique des CloudTrail événements](#).

Pour un enregistrement continu des événements survenus dans votre environnement Compte AWS, y compris des événements pour Amazon GameLift Servers, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions Régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- [Présentation de la création d'un journal de suivi](#)
- [CloudTrail services et intégrations pris en charge](#)
- [Configuration des notifications Amazon SNS pour CloudTrail](#)
- [Réception de fichiers CloudTrail journaux de plusieurs régions](#) et [réception de fichiers CloudTrail journaux de plusieurs comptes](#)

Tous Amazon GameLift Servers les actions sont enregistrées CloudTrail et documentées dans le [Amazon GameLift Servers Référence d'API](#) . Par exemple `CreateGameSession`, les appels `CreatePlayerSession` et les `UpdateGameSession` actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.

- Si la demande a été faite par un autre AWS service.

Pour de plus amples informations, veuillez consulter l'[élément userIdentity CloudTrail](#) .

Compréhension Amazon GameLift Servers entrées du fichier journal

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal qui illustre les `DescribeFleetAttributes` actions `CreateFleet` et.

```
{
  "Records": [
    {
      "eventVersion": "1.04",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
      },
      "eventTime": "2015-12-29T23:40:15Z",
      "eventSource": "gamelift.amazonaws.com",
      "eventName": "CreateFleet",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "[]",
      "requestParameters": {
        "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d",
        "eC2InboundPermissions": [
          {
            "ipRange": "10.24.34.0/23",
            "fromPort": 1935,
```

```

        "protocol": "TCP",
        "toPort": 1935
    }
],
"logPaths": [
    "C:\\game\\serverErr.log",
    "C:\\game\\serverOut.log"
],
"e2InstanceType": "c5.large",
"serverLaunchPath": "C:\\game\\MyServer.exe",
"description": "Test fleet",
"serverLaunchParameters": "-paramX=baz",
"name": "My_Test_Server_Fleet"
},
"responseElements": {
    "fleetAttributes": {
        "fleetId": "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e",
        "serverLaunchPath": "C:\\game\\MyServer.exe",
        "status": "NEW",
        "logPaths": [
            "C:\\game\\serverErr.log",
            "C:\\game\\serverOut.log"
        ],
        "description": "Test fleet",
        "serverLaunchParameters": "-paramX=baz",
        "creationTime": "Dec 29, 2015 11:40:14 PM",
        "name": "My_Test_Server_Fleet",
        "buildId": "build-92b6e8af-37a2-4c10-93bd-4698ea23de8d"
    }
},
"requestID": "824a2a4b-ae85-11e5-a8d6-61d5cafb25f2",
"eventID": "c8fbea01-fbf9-4c4e-a0fe-ad7dc205ce11",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
},
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
    }
}

```

```
    },
    "eventTime": "2015-12-29T23:40:15Z",
    "eventSource": "gamelift.amazonaws.com",
    "eventName": "DescribeFleetAttributes",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "[]",
    "requestParameters": {
      "fleetIds": [
        "fleet-0bb84136-4f69-4bb2-bfec-a9b9a7c3d52e"
      ]
    },
    "responseElements": null,
    "requestID": "82e7f0ec-ae85-11e5-a8d6-61d5cafb25f2",
    "eventID": "11daabcb-0094-49f2-8b3d-3a63c8bad86f",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  },
]
}
```

Enregistrement des messages du serveur Amazon GameLift Servers

Vous pouvez capturer des messages de serveur personnalisés à partir de vos Amazon GameLift Servers serveurs dans des fichiers journaux. La façon dont vous configurez la journalisation varie selon que vous utilisez des serveurs personnalisés ou Amazon GameLift Servers Realtime (voir les sous-sections appropriées dans ce chapitre).

Rubriques

- [Journalisation des messages du serveur \(serveurs personnalisés\)](#)
- [Journalisation des messages du serveur \(Amazon GameLift ServersRealtime\)](#)

Journalisation des messages du serveur (serveurs personnalisés)

Vous pouvez capturer des messages de serveur personnalisés à partir de vos serveurs Amazon GameLift Servers personnalisés dans des fichiers journaux. Pour en savoir plus sur la journalisation pour Amazon GameLift ServersRealtime, voir [Journalisation des messages du serveur \(Amazon GameLift ServersRealtime\)](#).

⚠ Important

La taille d'un fichier journal par session de jeu est limitée (voir [Amazon GameLift Servers points de terminaison et quotas](#) dans le Références générales AWS). À la fin d'une session de jeu Amazon GameLift Servers, les journaux du serveur sont chargés sur Amazon Simple Storage Service (Amazon S3). Amazon GameLift Servers ne téléchargera pas les journaux dépassant la limite. Les grumes peuvent pousser très rapidement et dépasser la limite de taille. Vous devez surveiller vos journaux et limiter la sortie des journaux aux seuls messages nécessaires.

Configuration de la journalisation pour les serveurs personnalisés

Avec les serveurs Amazon GameLift Servers personnalisés, vous écrivez votre propre code pour effectuer la journalisation, que vous configurez dans le cadre de la configuration des processus de votre serveur. Amazon GameLift Servers utilise votre configuration de journalisation pour identifier les fichiers qu'il doit télécharger sur Amazon S3 à la fin de chaque session de jeu.

Les instructions suivantes montrent comment configurer la journalisation à l'aide d'exemples de code simplifiés :

C++

Pour configurer la journalisation (C++)

1. Créez un vecteur de chaînes qui sont des chemins de répertoire vers les fichiers journaux du serveur de jeu.

```
std::string serverLog("serverOut.log");           // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. Indiquez votre vecteur comme étant celui [LogParameters](#) de votre [ProcessParameters](#) objet.

```
Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
```

```
listenPort,  
    Aws::GameLift::Server::LogParameters(logPaths));
```

3. Fournissez l'[ProcessParameters](#) objet lorsque vous appelez [ProcessReady\(\)](#).

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

Pour un exemple plus complet, voir [ProcessReady\(\)](#).

C#

Pour configurer la journalisation (C#)

1. Créez une liste de chaînes qui sont des chemins de répertoire vers les fichiers journaux du serveur de jeu.

```
List<string> logPaths = new List<string>();  
logPaths.Add("C:\\game\\serverOut.txt");    // Example of a log file that the  
game server writes
```

2. Indiquez votre liste en [LogParameter](#) tant qu'[ProcessParameters](#) objet.

```
var processReadyParameter = new ProcessParameters(  
    this.OnGameSession,  
    this.OnProcessTerminate,  
    this.OnHealthCheck,  
    this.OnGameSessionUpdate,  
    port,  
    new LogParameters(logPaths));
```

3. Fournissez l'[ProcessParameters](#) objet lorsque vous appelez [ProcessReady\(\)](#).

```
var processReadyOutcome =  
    GameLiftServerAPI.ProcessReady(processReadyParameter);
```

Pour un exemple plus complet, voir [ProcessReady\(\)](#).

Écrire dans des journaux

Vos fichiers journaux existent une fois que le processus de votre serveur a démarré. Vous pouvez écrire dans les journaux en utilisant n'importe quelle méthode pour écrire dans des fichiers. Pour capturer toutes les sorties standard et les sorties d'erreur de votre serveur, remappez les flux de sortie en fichiers journaux, comme dans les exemples suivants :

C++

```
std::freopen("serverOut.log", "w+", stdout);  
std::freopen("serverErr.log", "w+", stderr);
```

C#

```
Console.SetOut(new StreamWriter("serverOut.txt"));  
Console.SetError(new StreamWriter("serverErr.txt"));
```

Accès aux journaux du serveur

À la fin d'une session de jeu, les journaux Amazon GameLift Servers sont automatiquement stockés dans un compartiment Amazon S3 et les conservent pendant 14 jours. Pour obtenir l'emplacement des journaux d'une session de jeu, vous pouvez utiliser l'opération [GetGameSessionLogUrl](#) API. Pour télécharger les journaux, utilisez l'URL renvoyée par l'opération.

Journalisation des messages du serveur (Amazon GameLift ServersRealtime)

Vous pouvez capturer des messages de serveur personnalisés à partir de vos Amazon GameLift Servers Realtime fichiers journaux. Pour en savoir plus sur la journalisation pour les serveurs personnalisés, consultez [Journalisation des messages du serveur \(serveurs personnalisés\)](#).

Il existe différents types de messages que vous pouvez enregistrer dans vos fichiers journaux (voir [Enregistrement des messages dans le script de votre serveur](#)). Outre vos messages personnalisés, vos messages système Amazon GameLift Servers Realtime de sortie utilisent les mêmes types de messages et les écrivez dans les mêmes fichiers journaux. Vous pouvez ajuster le niveau de journalisation de votre parc afin de réduire le nombre de messages de journalisation générés par vos serveurs (voir [Réglage du niveau de journalisation](#)).

⚠ Important

La taille d'un fichier journal par session de jeu est limitée (voir [Amazon GameLift Servers points de terminaison et quotas](#) dans le Références générales AWS). À la fin d'une session de jeu Amazon GameLift Servers, les journaux du serveur sont chargés sur Amazon Simple Storage Service (Amazon S3). Amazon GameLift Servers ne téléchargera pas les journaux dépassant la limite. Les grumes peuvent pousser très rapidement et dépasser la limite de taille. Vous devez surveiller vos journaux et limiter la sortie des journaux aux seuls messages nécessaires.

Enregistrement des messages dans le script de votre serveur

Vous pouvez générer des messages personnalisés dans le [script de votre Amazon GameLift Servers Realtime](#). Pour envoyer des messages du serveur à un fichier journal, procédez comme suit :

1. Créez une variable pour contenir la référence à l'objet enregistreur.

```
var logger;
```

2. Dans la `init()` fonction, récupérez l'enregistreur à partir de l'objet de session et attribuez-le à votre variable d'enregistrement.

```
function init(rtSession) {  
    session = rtSession;  
    logger = session.getLogger();  
}
```

3. Appelez la fonction appropriée de l'enregistreur pour afficher un message.

Messages de débogage

```
logger.debug("This is my debug message...");
```

Messages d'information

```
logger.info("This is my info message...");
```

Messages d'avertissement

```
logger.warn("This is my warn message...");
```

Messages d'erreur

```
logger.error("This is my error message...");
```

Messages d'erreur fatals

```
logger.fatal("This is my fatal error message...");
```

Messages d'erreur fatale liés à l'expérience client

```
logger.cxfatal("This is my customer experience fatal error message...");
```

Pour un exemple des instructions de journalisation dans un script, consultez [Amazon GameLift ServersRealtimeexemple de script](#).

La sortie des fichiers journaux indique le type de message (DEBUG,,INFO,WARN, ERRORFATAL,CXFATAL), comme le montrent les lignes suivantes d'un exemple de journal :

```
09 Sep 2021 11:46:32,970 [INFO] (gamelift.js) 215: Calling GameLiftServerAPI.InitSDK...
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 220: GameLiftServerAPI.InitSDK succeeded
09 Sep 2021 11:46:32,993 [INFO] (gamelift.js) 223: Waiting for Realtime server to
start...
09 Sep 2021 11:46:33,15 [WARN] (index.js) 204: Connection is INSECURE. Messages will be
sent/received as plaintext.
```

Accès aux journaux du serveur

À la fin d'une session de jeu, les journaux sont Amazon GameLift Servers automatiquement stockés dans Amazon S3 et conservés pendant 14 jours. Vous pouvez utiliser l'[appel d'GetGameSessionLogUrl API](#) pour obtenir l'emplacement des journaux d'une session de jeu. Utilisez l'URL renvoyée par l'appel d'API pour télécharger les journaux.

Réglage du niveau de journalisation

Les grumes peuvent pousser très rapidement et dépasser la limite de taille. Vous devez surveiller vos journaux et limiter la sortie des journaux aux seuls messages nécessaires. En Amazon GameLift Servers Realtime effet, vous pouvez ajuster le niveau de journalisation en fournissant un paramètre dans la configuration d'exécution de votre flotte sous le formulaire `loggingLevel: LOGGING_LEVEL`, où **LOGGING_LEVEL** figure l'une des valeurs suivantes :

1. debug
2. info(par défaut)
3. warn
4. error
5. fatal
6. cxfatal

Cette liste est ordonnée du moins grave (debug) au plus grave (cxfatal). Vous définissez un seul `loggingLevel` et le serveur n'enregistrera que les messages correspondant à ce niveau de gravité ou à un niveau de gravité supérieur. Par exemple, le paramètre `loggingLevel:error` obligera tous les serveurs de votre parc à écrire `error` et à `cxfatal` envoyer des messages uniquement dans le journal. `fatal`

Vous pouvez définir le niveau de journalisation de votre flotte lorsque vous la créez ou une fois qu'elle est en cours d'exécution. La modification du niveau de journalisation de votre flotte après son exécution n'affectera que les journaux des sessions de jeu créées après la mise à jour. Les journaux des sessions de jeu existantes ne seront pas affectés. Si vous ne définissez pas de niveau de journalisation lors de la création de votre flotte, vos serveurs définiront le niveau de journalisation `info` par défaut. Reportez-vous aux sections suivantes pour obtenir des instructions permettant de définir le niveau de journalisation.

Configuration du niveau de journalisation lors de la création d'une Amazon GameLift ServersRealtime flotte (console)

Suivez les instructions indiquées [Création d'une EC2 flotte Amazon GameLift Servers gérée](#) pour créer votre flotte, en ajoutant ce qui suit :

- Dans la sous-étape d'allocation des processus du serveur de l'étape de gestion des processus, fournissez la paire clé-valeur du niveau de journalisation (telle que **loggingLevel:error**) en tant

que valeur pour les paramètres de lancement. Utilisez un caractère non alphanumérique (sauf une virgule) pour séparer le niveau de journalisation de tout paramètre supplémentaire (par exemple,).
`loggingLevel:error +map Winter444`

Définition du niveau de journalisation lors de la création d'une Amazon GameLift ServersRealtime flotte (AWS CLI)

Suivez les instructions indiquées [Création d'une EC2 flotte Amazon GameLift Servers gérée](#) pour créer votre flotte, en ajoutant ce qui suit :

- Dans l'argument du `--runtime-configuration` paramètre `for`[create-fleet](#), indiquez la paire clé-valeur du niveau de journalisation (telle que `loggingLevel:error`) en tant que valeur pour. Parameters Utilisez un caractère non alphanumérique (sauf une virgule) pour séparer le niveau de journalisation de tout paramètre supplémentaire. Consultez l'exemple suivant:

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,  
                        MaxConcurrentGameSessionActivations=2,  
                        ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,  
                                         Parameters=loggingLevel:error +map Winter444,  
                                         ConcurrentExecutions=10}]"
```

Configuration du niveau de journalisation pour un Amazon GameLift ServersRealtime parc en cours d'exécution (console)

Suivez les instructions de la section [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#) pour mettre à jour votre flotte à l'aide de la Amazon GameLift Servers console, en ajoutant ce qui suit :

- Sur la page Modifier le parc, sous Allocation des processus du serveur, indiquez la paire clé-valeur du niveau de journalisation (telle que `loggingLevel:error`) en tant que valeur pour les paramètres de lancement. Utilisez un caractère non alphanumérique (sauf une virgule) pour séparer le niveau de journalisation de tout paramètre supplémentaire (par exemple,).
`loggingLevel:error +map Winter444`

Configuration du niveau de journalisation pour une Amazon GameLift ServersRealtime flotte en cours de fonctionnement (AWS CLI)

Suivez les instructions indiquées [Mettre à jour la configuration d'une Amazon GameLift Servers flotte](#) pour mettre à jour votre flotte à l'aide du AWS CLI, avec l'ajout suivant :

- Dans l'argument du `--runtime-configuration` paramètre [update-runtime-configuration](#), indiquez la paire clé-valeur du niveau de journalisation (telle que `loggingLevel:error`) en tant que valeur pour `Parameters`. Utilisez un caractère non alphanumérique (sauf une virgule) pour séparer le niveau de journalisation de tout paramètre supplémentaire. Consultez l'exemple suivant:

```
--runtime-configuration "GameSessionActivationTimeoutSeconds=60,  
                        MaxConcurrentGameSessionActivations=2,  
                        ServerProcesses=[{LaunchPath=/local/game/myRealtimeLaunchScript.js,  
                                         Parameters=loggingLevel:error +map Winter444,  
                                         ConcurrentExecutions=10}]"
```

Sécurité dans Amazon GameLift Servers

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que fonctionnalité autonome avec Amazon EC2, consultez la section [Sécurité dans Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

La sécurité du cloud AWS est la priorité absolue. En tant que client AWS , vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des organisations les plus pointilleuses en termes de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le [modèle de responsabilité partagée](#) décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le AWS cadre des [programmes](#) de de). Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon GameLift Servers, consultez la section [AWS services concernés par programme de conformité](#) et .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, notamment la sensibilité de vos données, les exigences de votre entreprise et les lois AWS et réglementations applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Amazon GameLift Servers. Les rubriques suivantes expliquent comment configurer Amazon GameLift Servers pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos Amazon GameLift Servers ressources.

Rubriques

- [Protection des données dans Amazon GameLift Servers](#)
- [Gestion des identités et des accès pour Amazon GameLift Servers](#)
- [Journalisation et surveillance avec Amazon GameLift Servers](#)
- [Validation de conformité pour Amazon GameLift Servers](#)
- [Résilience dans Amazon GameLift Servers](#)

- [Sécurité de l'infrastructure dans Amazon GameLift Servers](#)
- [Configuration et analyse des vulnérabilités dans Amazon GameLift Servers](#)
- [Bonnes pratiques de sécurité pour Amazon GameLift Servers](#)

Protection des données dans Amazon GameLift Servers

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que fonctionnalité autonome avec Amazon EC2, consultez la section [Sécurité dans Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

Le [modèle de responsabilité AWS partagée](#) de s'applique à la protection des données dans Amazon GameLift Servers. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez [Questions fréquentes \(FAQ\) sur la confidentialité des données](#). Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée [AWS et RGPD \(Règlement général sur la protection des données\)](#) sur le Blog de sécurité AWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section [Utilisation des CloudTrail sentiers](#) dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.

- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez [Norme FIPS \(Federal Information Processing Standard\) 140-3](#).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Amazon GameLift Servers ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Les données spécifiques à Amazon GameLift Servers sont traitées comme suit :

- Les versions du serveur de jeu et les scripts vers lesquels vous les téléchargez Amazon GameLift Servers sont stockés dans Amazon S3. Il n'y a pas d'accès client direct à ces données une fois qu'elles sont chargées. Un utilisateur autorisé peut obtenir un accès temporaire pour charger des fichiers, mais ne peut pas afficher ou mettre à jour les fichiers directement dans Amazon S3. Pour supprimer des scripts et des builds, utilisez la console Amazon GameLift Servers ou l'API du service.
- Les données du journal de session de jeu sont stockées dans Amazon S3 pendant une période limitée une fois la session de jeu terminée. Les utilisateurs autorisés peuvent accéder aux données du journal en les téléchargeant via un lien dans la console Amazon GameLift Servers ou des appels à l'API du service.
- Les données de métriques et d'événement sont stockées dans Amazon GameLift Servers et sont accessibles via la console Amazon GameLift Servers ou des appels à l'API du service. Les données pouvant être récupérées concernent les parcs, les instances, les placements de session de jeu, les tickets de mise en relation, les sessions de jeu et les sessions de joueur. Les données sont également accessibles via Amazon CloudWatch et CloudWatch Events.
- Les données fournies par le client sont stockées dans Amazon GameLift Servers. Les utilisateurs autorisés peuvent y accéder via des appels à l'API du service. Les données potentiellement sensibles peuvent inclure des données de joueur, des données de session de joueur et de session de jeu (y compris les informations de connexion), des données de mise en relation, etc.

 Note

Si vous indiquez un joueur personnalisé IDs dans vos demandes, il est prévu que ces valeurs soient anonymisées UUIDs et ne contiennent aucune information permettant d'identifier le joueur.

Pour de plus amples informations sur la protection des données, veuillez consulter le billet de blog [Modèle de responsabilité partagée AWS et RGPD](#) à la page Blog sur la sécuritéAWS .

Chiffrement au repos

Le chiffrement au repos des données spécifiques à Amazon GameLift Servers est géré comme suit :

- Les versions et les scripts des serveurs de jeu sont stockés dans des compartiments Amazon S3 avec chiffrement côté serveur.
- Les données fournies par le client sont stockées dans Amazon GameLift Servers selon un format chiffré.

Chiffrement en transit

Les connexions au Amazon GameLift Servers APIs sont établies via une connexion sécurisée (SSL) et authentifiées à l'aide de [AWS Signature Version 4](#) (lors de la connexion via la AWS CLI ou le AWS SDK, la signature est gérée automatiquement). L'authentification est gérée à l'aide des stratégies d'accès définies par IAM pour les informations d'identification de sécurité utilisées pour établir la connexion.

La communication directe entre les clients de jeu et les serveurs de jeu est la suivante :

- Pour les serveurs de jeux personnalisés hébergés sur des ressources Amazon GameLift Servers, la communication n'implique pas le service Amazon GameLift Servers. Le chiffrement de cette communication est de la responsabilité du client. Vous pouvez utiliser des parcs compatibles TLS pour que vos clients de jeu authentifient le serveur de jeu lors de la connexion et chiffrent toutes les communications entre votre client de jeu et votre serveur de jeu.
- Lorsque la génération Amazon GameLift Servers Realtime de certificats TLS est activée, le trafic entre le client du jeu et Realtime les serveurs utilisant le SDK client pour Realtime est crypté en cours de vol. Le trafic TCP est chiffré avec TLS 1.2 et le trafic UDP avec DTLS 1.2.

Confidentialité du trafic inter-réseau

Vous pouvez accéder à distance à vos instances Amazon GameLift Servers en toute sécurité. Pour les instances qui utilisent Linux, SSH fournit un canal de communication sécurisé pour l'accès à distance. Pour les instances qui exécutent Windows, utilisez un client RDP (Remote Desktop Protocol). Avec Amazon GameLift ServersFleetIQ, l'accès à distance à vos instances à l'aide de AWS Systems Manager Session Manager et Run Command est crypté à l'aide de TLS 1.2, et les demandes de création de connexion sont signées à l'aide de SigV4. Pour obtenir de l'aide sur la connexion à une instance Amazon GameLift Servers gérée, veuillez consulter [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).

Gestion des identités et des accès pour Amazon GameLift Servers

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Des administrateurs IAM contrôlent les personnes qui s'authentifient (sont connectées) et sont autorisées (disposent d'autorisations) à utiliser des ressources Amazon GameLift Servers. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion des accès à l'aide de politiques](#)
- [Fonctionnement d'Amazon GameLift Servers avec IAM](#)
- [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#)
- [Résolution des problèmes d'identité et d'accès avec Amazon GameLift Servers](#)
- [AWS politiques gérées pour Amazon GameLift Servers](#)

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez. Amazon GameLift Servers

Utilisateur du service – Si vous utilisez le service Amazon GameLift Servers pour effectuer votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous

avez besoin. Plus vous utiliserez de fonctions Amazon GameLift Servers pour effectuer votre travail, plus vous pourriez avoir besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Amazon GameLift Servers, consultez [Résolution des problèmes d'identité et d'accès avec Amazon GameLift Servers](#).

Administrateur du service – Si vous êtes le responsable des ressources Amazon GameLift Servers de votre entreprise, vous bénéficiez probablement d'un accès total à Amazon GameLift Servers. Votre responsabilité est de déterminer Amazon GameLift Servers les fonctionnalités ainsi que les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec Amazon GameLift Servers, veuillez consulter [Fonctionnement d'Amazon GameLift Servers avec IAM](#).

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaitez peut-être en savoir plus sur la façon d'écrire des politiques pour gérer l'accès à Amazon GameLift Servers. Pour voir des exemples de politiques Amazon GameLift Servers basées sur l'identité que vous pouvez utiliser dans IAM, veuillez consulter [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#).

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section [Comment vous connecter à votre compte Compte AWS dans](#) le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez [AWS Signature Version 4 pour les demandes d'API](#) dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour plus d'informations, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Authentification multifactorielle AWS dans IAM](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez [Tâches nécessitant des informations d'identification d'utilisateur racine](#) dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez

vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez [Qu'est-ce que IAM Identity Center ?](#) dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité au sein de votre Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez [Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification](#) dans le Guide de l'utilisateur IAM.

Un [groupe IAM](#) est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour plus d'informations, consultez [Cas d'utilisation pour les utilisateurs IAM](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité au sein de votre Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous pouvez [passer d'un rôle d'utilisateur à un rôle IAM \(console\)](#). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez [Méthodes pour endosser un rôle](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- **Accès utilisateur fédéré** : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center .
- **Autorisations d'utilisateur IAM temporaires** : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- **Accès intercompte** : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.
- **Accès multiservices** — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
- **Sessions d'accès direct (FAS)** : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez [Transmission des sessions d'accès](#).
- **Rôle de service** : il s'agit d'un [rôle IAM](#) attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM.

Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.

- **Rôle lié à un service** — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- **Applications exécutées sur Amazon EC2** : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez [Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon](#) dans le guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la [présentation de la liste de contrôle d'accès \(ACL\)](#) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- **Limite d'autorisations** : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- **Politiques de contrôle des services (SCPs)** : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les Organizations et consultez SCPs les [politiques de contrôle des services](#) dans le Guide de AWS Organizations l'utilisateur.
- **Politiques de contrôle des ressources (RCPs)** : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les

autorisations effectives pour les identités, y compris Utilisateur racine d'un compte AWS, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, voir [Politiques de contrôle des ressources \(RCPs\)](#) dans le guide de AWS Organizations l'utilisateur.

- **Politiques de séance** : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez [Politiques de session](#) dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section [Logique d'évaluation des politiques](#) dans le guide de l'utilisateur IAM.

Fonctionnement d'Amazon GameLift Servers avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon GameLift Servers, découvrez les fonctions IAM que vous pouvez utiliser avec Amazon GameLift Servers.

Fonctions IAM que vous pouvez utiliser avec Amazon GameLift Servers

Fonctionnalité IAM	Prise en charge de Amazon GameLift Servers
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui

Fonctionnalité IAM	Prise en charge de Amazon GameLift Servers
ACLs	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Rôles de service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont Amazon GameLift Servers les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez la section [AWS Services compatibles avec IAM](#) dans le Guide de l'utilisateur IAM.

Stratégies basées sur l'identité pour Amazon GameLift Servers

Prend en charge les politiques basées sur l'identité : oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité, car celle-ci s'applique à l'utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez [Références des éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

Exemples de stratégies basées sur l'identité pour Amazon GameLift Servers

Pour voir des exemples de politiques Amazon GameLift Servers basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#).

Politiques basées sur une ressource dans Amazon GameLift Servers

Prend en charge les politiques basées sur les ressources : non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal intercompte à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un administrateur IAM du compte sécurisé doit également accorder à l'entité principale (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Actions de politique pour Amazon GameLift Servers

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec

autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour obtenir la liste des Amazon GameLift Servers actions, voir [Actions définies par Amazon GameLift Servers](#) dans la référence d'autorisation de service.

Les actions de politique dans Amazon GameLift Servers utilisent le préfixe suivant avant l'action :

```
gamelift
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [  
    "gamelift:action1",  
    "gamelift:action2"  
]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot Describe, incluez l'action suivante :

```
"Action": "gamelift:Describe*"
```

Pour voir des exemples de politiques Amazon GameLift Servers basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#).

Ressources de politique pour Amazon GameLift Servers

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON Resource indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément Resource ou NotResource. Il est recommandé de définir

une ressource à l'aide de son [Amazon Resource Name \(ARN\)](#). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

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

Pour une liste des types de Amazon GameLift Servers ressources et de leurs ARNs caractéristiques, voir [Ressources définies par Amazon GameLift Servers](#) dans la référence d'autorisation de service. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez [Actions définies par Amazon GameLift Servers](#).

Certaines Amazon GameLift Servers ressources ont des valeurs ARN, ce qui permet de gérer leur accès à l'aide de politiques IAM. L'ARN de la ressource de parc Amazon GameLift Servers respecte la syntaxe suivante :

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

Pour plus d'informations sur le format de ARNs, consultez [Amazon Resource Names \(ARNs\)](#) dans le Références générales AWS.

Par exemple, pour spécifier le parc `fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa` dans votre déclaration, utilisez l'ARN suivant :

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

Pour spécifier toutes les flottes appartenant à un compte spécifique, utilisez un caractère générique (*) :

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

Pour voir des exemples de politiques Amazon GameLift Servers basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#).

Clés de condition de politique pour Amazon GameLift Servers

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des [opérateurs de condition](#), tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération AND logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une OR opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez [Éléments d'une politique IAM : variables et identifications](#) dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de [contexte de condition AWS globales](#) dans le guide de l'utilisateur IAM.

Pour obtenir la liste des clés de Amazon GameLift Servers condition, voir [Clés de condition pour Amazon GameLift Servers](#) dans la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez [Actions définies par Amazon GameLift Servers](#).

Pour voir des exemples de politiques Amazon GameLift Servers basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour Amazon GameLift Servers](#).

ACLs dans Amazon GameLift Servers

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec Amazon GameLift Servers

Prise en charge d'ABAC (balises dans les politiques) : Oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans [l'élément de condition](#) d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez [Définition d'autorisations avec l'autorisation ABAC](#) dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez [Utilisation du contrôle d'accès par attributs \(ABAC\)](#) dans le Guide de l'utilisateur IAM.

Pour un exemple de politique basée sur l'identité qui limite l'accès à une ressource en fonction des balises associées à cette ressource, voir. [Afficher les flottes Amazon GameLift Servers en fonction des balises](#)

Utilisation des informations d'identification temporaires avec Amazon GameLift Servers

Prend en charge les informations d'identification temporaires : oui

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent

avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation [d'IAM](#) dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, consultez [Passage d'un rôle utilisateur à un rôle IAM \(console\)](#) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez [Informations d'identification de sécurité temporaires dans IAM](#).

Autorisations de principal interservices pour Amazon GameLift Servers

Prend en charge les sessions d'accès direct (FAS) : oui

Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez [Transmission des sessions d'accès](#).

Rôles de service pour Amazon GameLift Servers

Prend en charge les rôles de service : oui

Un rôle de service est un [rôle IAM](#) qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.

 Warning

La modification des autorisations d'un rôle de service peut altérer la fonctionnalité d'Amazon GameLift Servers. Ne modifiez des rôles de service que quand Amazon GameLift Servers vous le conseille.

Autorisez vos serveurs de jeu Amazon GameLift Servers hébergés à accéder à d'autres AWS ressources, telles qu'une AWS Lambda fonction ou une base de données Amazon DynamoDB. Les serveurs de jeu étant hébergés sur des flottes qui les Amazon GameLift Servers gèrent, vous avez besoin d'un rôle de service qui donne un accès Amazon GameLift Servers limité à vos autres AWS ressources. Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

Rôles liés à un service pour Amazon GameLift Servers

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus de détails sur la création ou la gestion des rôles liés à un service, consultez la section [AWS Services compatibles avec IAM dans le Guide de l'utilisateur d'IAM](#). Recherchez un service dans le tableau qui inclut un Yes dans la colonne Rôles liés à un service. Choisissez Oui pour consulter la documentation relative aux rôles liés à un service pour ce service.

Exemples de politiques basées sur l'identité pour Amazon GameLift Servers

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources Amazon GameLift Servers. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez [Création de politiques IAM \(console\)](#) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Amazon GameLift Servers, y compris le format de ARNs pour chacun des types de ressources, voir [Actions, ressources et clés de condition Amazon GameLift Servers](#) dans la référence d'autorisation de service.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console Amazon GameLift Servers](#)
- [Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations](#)
- [Autoriser l'accès de joueurs pour les sessions de jeu](#)
- [Autoriser l'accès à une Amazon GameLift Servers file d'attente](#)
- [Afficher les flottes Amazon GameLift Servers en fonction des balises](#)
- [Accédez à un fichier de compilation de jeu dans Amazon S3](#)

Bonnes pratiques en matière de politiques

Les stratégies basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon GameLift Servers dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [politiques gérées par AWS](#) ou [politiques gérées par AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par

exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politiques avec IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Sécurisation de l'accès aux API avec MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon GameLift Servers

Pour accéder à la console Amazon GameLift Servers, vous devez disposer d'un ensemble minimum d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails Amazon GameLift Servers des ressources de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Pour garantir que ces entités peuvent toujours utiliser la Amazon GameLift Servers console, ajoutez des autorisations aux utilisateurs et aux groupes en utilisant la syntaxe décrite dans les exemples suivants et dans [Exemples d'autorisations d'administration](#). Pour de plus amples informations, veuillez consulter [Définissez les autorisations utilisateur pour Amazon GameLift Servers](#).

Les utilisateurs qui utilisent des opérations Amazon GameLift Servers via AWS CLI ou via AWS l'API n'ont pas besoin d'autorisations minimales sur la console. Au lieu de cela, vous pouvez limiter l'accès aux seules opérations que l'utilisateur doit effectuer. Par exemple, un utilisateur joueur, agissant pour

le compte de clients du jeu, a besoin d'un accès pour demander des sessions de jeu, placer des joueurs dans des jeux et effectuer d'autres tâches.

Pour plus d'informations sur les autorisations requises pour utiliser toutes les fonctionnalités de Amazon GameLift Servers la console, consultez la syntaxe des autorisations pour les administrateurs dans [Exemples d'autorisations d'administration](#).

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
    },
  ],
}
```

```

        "Resource": "*"
    }
]
}

```

Autoriser l'accès de joueurs pour les sessions de jeu

Pour placer des joueurs dans des sessions de jeu, les clients du jeu et les services principaux ont besoin d'autorisations. Pour des exemples de politiques pour ces scénarios, voir [Exemples d'autorisations utilisateur pour les joueurs](#).

Autoriser l'accès à une Amazon GameLift Servers file d'attente

L'exemple suivant fournit à un utilisateur l'accès à une file d'Amazon GameLift Serversattente spécifique.

Cette politique accorde à l'utilisateur l'autorisation d'ajouter, de mettre à jour et de supprimer des destinations de file d'attente avec les actions suivantes : `gamelift:UpdateGameSessionQueue`, `gamelift>DeleteGameSessionQueue`, et `gamelift:DescribeGameSessionQueues`. Comme indiqué, cette politique utilise l'élément `Resource` pour limiter l'accès à une seule file d'attente : `gamesessionqueue/examplequeue123`.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewSpecificQueueInfo",
      "Effect": "Allow",
      "Action": [
        "gamelift:DescribeGameSessionQueues"
      ],
      "Resource": "arn:aws:gamelift:us-east-1:555555555555:gamesessionqueue/examplequeue123"
    },
    {
      "Sid": "ManageSpecificQueue",
      "Effect": "Allow",

```

```

    "Action": [
      "gamelift:UpdateGameSessionQueue",
      "gamelift>DeleteGameSessionQueue"
    ],
    "Resource": "arn:aws:gamelift:us-east-1:111122223333:gamesessionqueue/
examplequeue123"
  }
]
}

```

Afficher les flottes Amazon GameLift Servers en fonction des balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux ressources Amazon GameLift Servers en fonction des balises. Cet exemple montre comment créer une politique qui permet de consulter une flotte si le Owner tag correspond au nom d'utilisateur de l'utilisateur. Cette politique accorde également les autorisations nécessaires pour effectuer cette opération dans la console.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListFleetsInConsole",
      "Effect": "Allow",
      "Action": "gamelift:ListFleets",
      "Resource": "*"
    },
    {
      "Sid": "ViewFleetIfOwner",
      "Effect": "Allow",
      "Action": "gamelift:DescribeFleetAttributes",
      "Resource": "arn:aws:gamelift:*:*:fleet/*",
      "Condition": {
        "StringEquals": {"aws:ResourceAccount": "${aws:username}"}
      }
    }
  ]
}

```


Accédez à un fichier de compilation de jeu dans Amazon S3

Après avoir intégré votre serveur de jeu Amazon GameLift Servers, téléchargez les fichiers de compilation sur Amazon S3. Pour accéder Amazon GameLift Servers aux fichiers de compilation, appliquez la politique suivante.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::bucket-name/object-name"
    }
  ]
}
```

Pour plus d'informations sur le téléchargement de fichiers de Amazon GameLift Servers jeu, consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#).

Résolution des problèmes d'identité et d'accès avec Amazon GameLift Servers

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon GameLift Servers et AWS Identity and Access Management (IAM).

Rubriques

- [Je ne suis pas autorisé à effectuer une action dans Amazon GameLift Servers](#)
- [Je ne suis pas autorisé à effectuer iam : PassRole](#)

- [Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes Amazon GameLift Servers ressources](#)

Je ne suis pas autorisé à effectuer une action dans Amazon GameLift Servers

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, contactez l'administrateur de votre AWS compte pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit lorsque l'utilisateur mateojackson IAM essaie d'utiliser la console pour afficher les détails d'une file d'attente mais ne dispose pas des `gamelift:DescribeGameSessionQueues` autorisations nécessaires :

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder en lecture à la `examplequeue123` ressource utilisant l'`gamelift:DescribeGameSessionQueues` action.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à Amazon GameLift Servers.

Certains vos Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé marymajor essaie d'utiliser la console pour exécuter une action dans Amazon GameLift Servers. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes Amazon GameLift Servers ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon GameLift Servers prend en charge ces fonctionnalités, consultez [Fonctionnement d'Amazon GameLift Servers avec IAM](#).
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section [Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section [Fournir un accès à des ressources Comptes AWS détenues par des tiers](#) dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

AWS politiques gérées pour Amazon GameLift Servers

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation

courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des [politiques gérées par le client](#) qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez [Politiques gérées par AWS](#) dans le Guide de l'utilisateur IAM.

AWS politique gérée : GameLiftContainerFleetPolicy

Vous pouvez les associer GameLiftContainerFleetPolicy à vos rôles IAM.

La politique accorde des autorisations pour les actions de calcul dans une flotte de Amazon GameLift Servers conteneurs. Une flotte de conteneurs est un ensemble de ressources d'hébergement qui se Amazon GameLift Servers gère pour vous. Amazon GameLift Servers a besoin d'autorisations pour se connecter au Amazon GameLift Servers service et à d'autres AWS services en votre nom.

Lorsque vous créez un parc de conteneurs avec Amazon GameLift Servers, fournissez un rôle de service IAM auquel est attachée la politique GameLiftContainerFleetPolicy gérée. Pour obtenir des instructions sur la création du rôle de service, consultez [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Pour de plus amples informations, veuillez consulter [GameLiftContainerFleetPolicy](#).

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `cloudwatch`— Permet d'Amazon GameLift Servers écrire des journaux de session de jeu dans un flux de journal Amazon CloudWatch Events de votre AWS compte.
- `cloudwatch` — Permet Amazon GameLift Servers de créer un groupe de CloudWatch journaux pour organiser les données de session de jeu dans le flux de journal.

- `s3`— Permet d'Amazon GameLift Servers d'écrire des journaux de session de jeu dans un bucket Amazon Simple Storage Service de votre AWS compte.
- `s3`— Permet Amazon GameLift Servers de récupérer l' Région AWS emplacement d'un compartiment Amazon S3 spécifié, à l'aide de l'action de l'API `GetBucketLocation`.
- `gamelift` — Permet Amazon GameLift Servers de récupérer un jeton d'authentification qui permet à un serveur de jeu hébergé de communiquer avec le Amazon GameLift Servers service via votre AWS compte.

Amazon GameLift Servers mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées Amazon GameLift Servers depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la [page des notes de Amazon GameLift Servers version](#).

Modification	Description	Date
GameLiftContainerFleetPolicy — Modification	Amazon GameLift Servers a ajouté de nouvelles autorisations pour récupérer le Région AWS contenu d'un compartiment Amazon S3.	5 février 2024
GameLiftContainerFleetPolicy : nouvelle politique	Amazon GameLift Servers a ajouté de nouvelles autorisations pour permettre aux conteneurs de serveurs de jeux de fonctionner sur des flottes Amazon GameLift Servers gérées.	12 novembre 2024
Amazon GameLift Servers a démarré le suivi des modifications	Amazon GameLift Servers a commencé à suivre les modifications apportées AWS à ses politiques gérées.	12 novembre 2024

Journalisation et surveillance avec Amazon GameLift Servers

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de Amazon GameLift Servers et vos AWS solutions. Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant.

AWS et Amazon GameLift Servers fournissent plusieurs outils pour surveiller les ressources d'hébergement de vos jeux et répondre aux incidents potentiels.

CloudWatch Alarmes Amazon

À l'aide des CloudWatch alarmes Amazon, vous observez une seule métrique sur une période que vous spécifiez. Si la métrique dépasse un seuil donné, une notification est envoyée à une rubrique Amazon SNS ou à une politique AWS Auto Scaling. CloudWatch les alarmes sont déclenchées lorsque leur état change et sont maintenues pendant un certain nombre de périodes, et non lorsqu'elles sont dans un état particulier. Pour de plus amples informations, veuillez consulter [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#).

AWS CloudTrail Journaux

CloudTrail fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon GameLift Servers. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Amazon GameLift Servers, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite et des informations supplémentaires. Pour de plus amples informations, veuillez consulter [Journalisation Amazon GameLift Servers Appels d'API avec AWS CloudTrail](#).

Validation de conformité pour Amazon GameLift Servers

Amazon GameLift Servers ne fait l'objet d'aucun programme de AWS conformité.

Pour savoir si un [programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de [AWS conformité Programmes AWS](#) de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#) .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- [Conformité et gouvernance de la sécurité](#) : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- [Référence des services éligibles HIPAA](#) : liste les services éligibles HIPAA. Tous ne Services AWS sont pas éligibles à la loi HIPAA.
- AWS Ressources de <https://aws.amazon.com/compliance/resources/> de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- [AWS Guides de conformité destinés aux clients](#) — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans plusieurs cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- [Évaluation des ressources à l'aide des règles](#) du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- [AWS Security Hub](#)— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez [Référence des contrôles Security Hub](#).
- [Amazon GuardDuty](#) — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- [AWS Audit Manager](#)— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans Amazon GameLift Servers

Si vous utilisez Amazon GameLift Servers FleetIQ en tant que fonctionnalité autonome avec Amazon EC2, consultez [la section Sécurité sur Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section [Infrastructure AWS globale](#).

Outre l'infrastructure AWS mondiale, Amazon GameLift Servers propose les fonctionnalités suivantes pour répondre à vos besoins en matière de résilience des données :

- **Files d'attente multirégionales** — Amazon GameLift Servers les files d'attente de session de jeu sont utilisées pour placer de nouvelles sessions de jeu avec les ressources d'hébergement disponibles. Les files d'attente couvrant plusieurs régions peuvent rediriger les placements de session de jeu en cas de panne au niveau d'une région. Pour de plus amples informations et connaître les bonnes pratiques relatives à la création de files d'attente de session de jeu, veuillez consulter [Personnaliser une file d'attente de session de jeu](#).
- **Mise à l'échelle automatique de la capacité** : maintenez l'intégrité et la disponibilité de vos ressources d'hébergement en utilisant Amazon GameLift Servers outils de mise à l'échelle. Ces outils offrent une palette d'options qui vous permettent d'ajuster la capacité de votre parc en fonction des besoins de votre jeu et des joueurs. Pour plus d'informations sur le dimensionnement, consultez [Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#).
- **Distribution entre les instances** — Amazon GameLift Servers répartit le trafic entrant sur plusieurs instances, en fonction de la taille du parc. Pour les jeux en production, une bonne pratique consiste à disposer de plusieurs instances pour conserver la disponibilité au cas où une instance devient non saine ou ne répond pas.
- **Stockage Amazon S3** : versions de serveurs de jeu et scripts téléchargés sur Amazon GameLift Servers sont stockés dans Amazon S3 à l'aide de la classe de stockage Standard, qui utilise plusieurs répliquions de centres de données pour améliorer la résilience. Les journaux des

sessions de jeu sont également stockés dans Amazon S3 à l'aide de la classe de stockage Standard.

Sécurité de l'infrastructure dans Amazon GameLift Servers

Si vous utilisez Amazon GameLift Servers FleetIQ en tant que fonctionnalité autonome avec Amazon EC2, consultez [la section Sécurité sur Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

En tant que service géré, Amazon GameLift Servers est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc [Amazon Web Services : présentation des processus de sécurité](#).

Vous utilisez des appels d'API AWS publiés pour accéder Amazon GameLift Servers via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) 1.2 ou version ultérieure. Nous recommandons TLS 1.3 ou version ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Le Amazon GameLift Servers le service place toutes les flottes dans les clouds privés virtuels Amazon (VPCs) afin que chaque flotte existe dans une zone logiquement isolée du AWS cloud. Vous pouvez utiliser ... Amazon GameLift Servers politiques pour contrôler l'accès depuis des points de terminaison VPC spécifiques ou spécifiques. VPCs En fait, cela isole l'accès au réseau à un Amazon GameLift Servers ressource provenant uniquement du VPC spécifique au sein du AWS réseau. Lorsque vous créez un parc, vous spécifiez une plage de numéros de port et d'adresses IP. Ces plages limitent la façon dont le trafic entrant peut accéder aux serveurs de jeux hébergés sur un VPC de parc. Utilisez les bonnes pratiques en matière de sécurité standard lorsque vous choisissez les paramètres d'accès au parc.

Configuration et analyse des vulnérabilités dans Amazon GameLift Servers

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que fonctionnalité autonome avec Amazon EC2, consultez la section [Sécurité dans Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

La configuration et les contrôles informatiques sont une responsabilité partagée entre AWS et vous, notre client. Pour plus d'informations, consultez le [modèle de responsabilité AWS partagée](#). AWS gère les tâches de sécurité de base telles que l'application de correctifs au système d'exploitation client (OS) et aux bases de données, la configuration du pare-feu et la reprise après sinistre. Ces procédures ont été vérifiées et certifiées par les tiers appropriés. Pour plus de détails, consultez la ressource suivante : [Amazon Web Services : présentation des processus de sécurité](#) (livre blanc).

Les bonnes pratiques de sécurité suivantes s'appliquent également à la configuration et à l'analyse des vulnérabilités dans Amazon GameLift Servers :

- Les clients sont responsables de la gestion des logiciels déployés sur les instances Amazon GameLift Servers pour l'hébergement de jeux. En particulier :
 - Les logiciels d'application de serveur de jeux fournis par le client doivent être gérés, y compris les mises à jour et les correctifs de sécurité. Pour mettre à jour le logiciel du serveur de jeu, téléchargez une nouvelle version Amazon GameLift Servers ou fournissez-la dans une nouvelle image de conteneur, créez une nouvelle flotte et redirigez le trafic vers la nouvelle flotte.
 - L'AMI (Amazon Machine Image) de base, qui inclut le système d'exploitation, est mise à jour uniquement lorsqu'un nouveau parc est créé. Pour appliquer des correctifs, mettre à jour et sécuriser le système d'exploitation et les autres applications faisant partie de l'AMI, remplacez régulièrement les flottes, quelles que soient les mises à jour du serveur de jeu. Au minimum, les flottes de production devraient être remplacées au moins une fois par an. Pour plus d'informations, consultez [Maintenir à jour les environnements d'exécution des flottes](#).
- Les clients devraient envisager de mettre régulièrement à jour leurs jeux avec les dernières versions du SDK, notamment le AWS SDK, le SDK Amazon GameLift Servers serveur et le SDK Amazon GameLift Servers client pour serveurs en temps réel.

Bonnes pratiques de sécurité pour Amazon GameLift Servers

Si vous l'utilisez en Amazon GameLift Servers FleetIQ tant que fonctionnalité autonome avec Amazon EC2, consultez la section [Sécurité dans Amazon EC2 dans](#) le guide de l' EC2 utilisateur Amazon.

Amazon GameLift Servers fournit différentes fonctionnalités de sécurité à prendre en compte lorsque vous développez et implémentez vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Maintenir à jour les environnements d'exécution des flottes

Amazon GameLift Servers recommande vivement de remplacer régulièrement les flottes gérées (y compris les flottes de conteneurs gérées EC2 et gérées) afin de garantir des environnements d'exécution sécurisés pour vos serveurs de jeux. Les flottes qui fonctionnent pendant de longues périodes sans mises à jour d'exécution peuvent contenir des dépendances obsolètes et des failles de sécurité susceptibles de compromettre vos serveurs de jeu. Pour plus de détails sur la manière dont la responsabilité est partagée pour les logiciels déployés dans Amazon GameLift Servers les flottes, voir [Configuration et analyse des vulnérabilités dans Amazon GameLift Servers](#).

L'environnement d'exécution d'un parc géré est déterminé par sa version Amazon Machine Image (AMI). Lorsqu'un nouveau parc est créé, Amazon GameLift Servers la dernière version d'AMI disponible lui est attribuée, et toutes les instances de calcul de ce parc sont déployées avec cette version. Pour mettre à jour la version de l'AMI, vous devez créer une nouvelle flotte. Pour plus de détails sur les versions actuelles de l'AMI, consultez [Amazon GameLift Servers Versions de l'AMI](#).

Pratiques recommandées :

- Surveillez l'âge des flottes et remplacez les flottes âgées de plus de 30 jours : vous pouvez suivre la date de création d'une flotte dans la Amazon GameLift Servers console ou utiliser la CLI pour récupérer les attributs de la flotte. Amazon GameLift Servers affiche des avertissements dans la console pour les flottes de plus de 90 jours et avertit les titulaires de compte par e-mail pour les flottes de plus d'un an.

 Note

La mise à jour d'un parc (par exemple en utilisant [UpdateFleetAttributes](#) ou [UpdateContainerFleet](#)) ne modifie pas la version de l'AMI. Vous devez créer une nouvelle flotte.

- Remplacez régulièrement les flottes en fonction de leur état de sécurité : établissez un calendrier régulier pour créer de nouvelles flottes et retirer les anciennes flottes. Envisagez d'utiliser un service tel qu'[Amazon Q](#) pour vérifier le code de votre jeu avec la version actuelle de l'AMI, détecter les problèmes de sécurité et suggérer des mesures correctives.
- Testez les versions de serveur avec les dernières versions d'AMI avant le déploiement : vous devrez peut-être modifier la version de votre serveur et la télécharger Amazon GameLift Servers avant de créer un nouveau parc.
- Gérez les quotas de flotte pour votre AWS compte : vous pouvez demander des augmentations de limites si nécessaire pour créer des flottes de remplacement. Pour de plus amples informations, veuillez consulter [Amazon GameLift Servers points de terminaison et quotas](#).
- Envisagez d'automatiser le remplacement des flottes : vous pouvez automatiser les processus pour créer de nouvelles flottes et migrer le trafic des joueurs depuis les anciennes flottes. Par exemple :
 - AWS CloudFormation À utiliser pour automatiser la création et la gestion de flottes. Conservez les configurations de votre flotte sous forme AWS CloudFormation de modèles et utilisez-les pour lancer des piles de ressources.
 - Tirez parti de la fonction d'Amazon GameLift Servers alias pour extraire une flotte spécifique IDs. Les alias de flotte permettent de transférer facilement le trafic des joueurs d'une flotte existante à une nouvelle, sans perturber les sessions de jeu en cours. Pour en savoir plus, consultez [Abstraire un Amazon GameLift Servers désignation de la flotte avec un alias](#).
 - Utilisez des stratégies blue/green de déploiement pour réduire les risques liés à la migration et éviter toute interruption de service. Avec deux environnements de production identiques, vous pouvez tirer parti d'un environnement de test complet similaire à celui de la production, exercer un meilleur contrôle sur le processus de migration et garantir des annulations instantanées.

Sécurisez vos configurations de ports

Nous vous déconseillons vivement d'ouvrir des ports vers Internet car cela représente un risque pour la sécurité. Par exemple, la configuration suivante ouvre un port de poste de travail distant qui permet à toute personne connectée à Internet d'accéder à l'instance :

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "0.0.0.0/0",
      "Protocol": "RDP",
      "ToPort": 3389
    }
  ]
}
```

Utilisez-le plutôt [UpdateFleetPortSettings](#) pour ouvrir un port avec une adresse IP ou une plage d'adresses spécifique, comme indiqué dans cet exemple :

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "54.186.139.221/32",
      "Protocol": "TCP",
      "ToPort": 3389
    }
  ]
}
```

Ressources de sécurité supplémentaires

Pour plus d'informations sur la manière dont vous pouvez rendre votre utilisation Amazon GameLift Servers plus sûre, consultez le [pilier AWS Well-Architected Tool Sécurité](#).

Amazon GameLift Servers guides de référence

Cette section contient la documentation de référence pour utiliser Amazon GameLift Servers.

Rubriques

- [API de service pour Amazon GameLift Servers](#)
- [SDK de serveur 5.x pour Amazon GameLift Servers](#)
- [SDK de serveur pour Amazon GameLift Servers version 4 et antérieures](#)
- [Amazon GameLift ServersAmazon GameLift ServersréférenceRealtime](#)
- [Événements de placement lors des sessions de jeu](#)
- [Amazon GameLift ServersVersions de l'AMI](#)
- [Amazon GameLift Serverspoints de terminaison et quotas](#)
- [Limites de l'API Amazon GameLift Servers](#)
- [Limites de l'API du SDK Amazon GameLift Servers](#)
- [Amazon GameLift ServersBalises ping UDP](#)

API de service pour Amazon GameLift Servers

Utilisez cette liste basée sur les tâches pour rechercher les opérations d'API lors de la création de vos solutions d'hébergement de Amazon GameLift Servers jeux et d'autres fonctionnalités. Le AWS SDK inclut ces opérations dans l'espace de `aws.gamelift` noms. [Téléchargez le AWS SDK](#) ou [consultez la documentation de référence de l'Amazon GameLift ServersAPI](#). Vous pouvez également utiliser l'API avec l'interface de ligne de commande de AWS (AWS CLI), comme indiqué dans la [référence des AWS CLI commandes](#).

L'API inclut deux ensembles d'opérations pour l'hébergement géré de jeux :

- [Gérer les ressources Amazon GameLift Servers d'hébergement](#)
- [Démarez des sessions de jeu et rejoignez des joueurs](#)

L'API Amazon GameLift Servers de service contient également des opérations à utiliser avec d'autres Amazon GameLift Servers outils et solutions. Pour en obtenir la liste FleetIQ APIs, consultez la section [Opérations de FleetIQ l'API](#). Pour une liste des FlexMatch APIs pour le matchmaking, voir les [opérations de FlexMatch l'API](#).

Gérer les ressources Amazon GameLift Servers d'hébergement

Effectuez ces opérations pour configurer les ressources d'hébergement de vos serveurs de jeu, augmenter la capacité pour répondre à la demande des joueurs, obtenir des indicateurs de performance et d'utilisation, etc. Utilisez ces opérations d'API lorsque vous hébergez des serveurs de jeux avec Amazon GameLift Servers, notamment Amazon GameLift Servers Realtime. Vous pouvez également travailler dans [Amazon GameLift Servers la console](#) pour la plupart des tâches de gestion des ressources ou passer des appels avec l'outil AWS Command Line Interface (AWS CLI).

Préparer les serveurs de jeu pour le déploiement

Téléchargez et configurez le code du serveur de jeu de votre jeu en vue du déploiement et du lancement sur les ressources d'hébergement.

Gérez les versions personnalisées de serveurs de jeu

- [upload-build](#) — Télécharge les fichiers de build depuis un chemin local et crée une nouvelle ressource de Amazon GameLift Servers build. Cette opération, disponible sous forme de AWS CLI commande, est la méthode la plus courante pour télécharger des versions de serveurs de jeux.
- [CreateBuild](#) — Créez une nouvelle version à l'aide de fichiers stockés dans un compartiment Amazon S3.
- [ListBuilds](#) — Obtenez une liste de toutes les versions téléchargées Amazon GameLift Servers dans une région.
- [DescribeBuild](#) — Récupère les informations associées à un build.
- [UpdateBuild](#) — Modifiez les métadonnées du build, y compris le nom et la version du build.
- [DeleteBuild](#) — Supprime un build de Amazon GameLift Servers.

Gérer les scripts Amazon GameLift Servers Realtime de configuration

- [CreateScript](#) — Téléchargez JavaScript des fichiers et créez une nouvelle ressource de Amazon GameLift Servers script.
- [ListScripts](#) — Obtenez une liste de tous les Realtime scripts téléchargés Amazon GameLift Servers dans une région.
- [DescribeScript](#) — Récupère les informations associées à un Realtime script.
- [UpdateScript](#) — Modifiez les métadonnées du script et téléchargez le contenu révisé du script.
- [DeleteScript](#) — Supprime un Realtime script de Amazon GameLift Servers.

Configuration des ressources informatiques pour l'hébergement

Configurez les ressources d'hébergement et déployez-les avec le script de compilation ou de Realtime configuration de votre serveur de jeu.

Création et gestion de flottes

- [CreateFleet](#)— Configurez et déployez un nouveau Amazon GameLift Servers parc de ressources informatiques pour faire fonctionner vos serveurs de jeu. Une fois déployés, les serveurs de jeu sont automatiquement lancés tels que configurés et prêts à héberger des sessions de jeu.
- [ListFleets](#)— Obtenez une liste de toutes les flottes d'une Amazon GameLift Servers région.
- [DeleteFleet](#)— Supprimez une flotte qui ne gère plus de serveurs de jeu ou n'héberge plus de joueurs.
- Afficher/mettre à jour les emplacements des flottes.
 - [CreateFleetLocations](#)— Ajoutez des sites distants à une flotte existante qui prend en charge plusieurs sites
 - [DescribeFleetLocationAttributes](#)— Obtenez une liste de tous les sites distants d'une flotte et consultez l'état actuel de chaque site.
 - [DeleteFleetLocations](#)— Supprimez les sites distants d'une flotte qui prend en charge plusieurs sites.
- Afficher/mettre à jour les configurations de flotte.
 - [DescribeFleetAttributes/UpdateFleetAttributes](#)— Affichez ou modifiez les métadonnées et les paramètres d'une flotte pour la protection des sessions de jeu et les limites de création de ressources.
 - [DescribeFleetPortSettings/UpdateFleetPortSettings](#)— Affichez ou modifiez les autorisations entrantes (adresses IP et plages de paramètres de port) autorisées pour une flotte.
 - [DescribeRuntimeConfiguration/UpdateRuntimeConfiguration](#)— Affichez ou modifiez les processus de serveur (et leur nombre) à exécuter sur chaque instance d'un parc.

Gérez la capacité de votre flotte

- [DescribeEC2InstanceLimits](#) — Récupérez le nombre maximum d'instances autorisées pour le AWS compte courant et le niveau d'utilisation actuel.
- [DescribeFleetCapacity](#)— Récupérez les paramètres de capacité actuels pour la région d'origine d'une flotte.

- [DescribeFleetLocationCapacity](#)— Récupérez les paramètres de capacité actuels pour chaque site d'une flotte multi-sites.
- [UpdateFleetCapacity](#)— Ajustez manuellement les paramètres de capacité d'une flotte.
- Configurez :
 - [PutScalingPolicy](#)— Activez l'auto-scaling basé sur les cibles, créez une politique d'auto-scaling personnalisée ou mettez à jour une politique existante.
 - [DescribeScalingPolicies](#)— Récupère une politique d'auto-scaling existante.
 - [DeleteScalingPolicy](#)— Supprimez une politique d'auto-scaling et empêchez qu'elle n'affecte la capacité d'une flotte.
 - [StartFleetActions](#)— Redémarrez les politiques d'auto-scaling d'une flotte.
 - [StopFleetActions](#)— Suspendez les politiques d'auto-scaling d'une flotte.

Surveillance de l'activité de la flotte.

- [DescribeFleetUtilization](#)— Récupérez des statistiques sur le nombre de processus de serveur, de sessions de jeu et de joueurs actuellement actifs sur une flotte.
- [DescribeFleetLocationUtilization](#)— Récupérez les statistiques d'utilisation pour chaque site d'un parc multi-sites.
- [DescribeFleetEvents](#)— Afficher les événements enregistrés pour une flotte pendant une période spécifiée.
- [DescribeGameSessions](#)— Récupérez les métadonnées d'une session de jeu, y compris la durée d'une partie et le nombre actuel de joueurs.

Configurez des files d'attente pour le placement des sessions de jeu

Configurez des files d'attente sur plusieurs flottes et plusieurs régions pour placer des sessions de jeu avec les meilleures ressources d'hébergement disponibles en matière de coût, de latence et de résilience.

- [CreateGameSessionQueue](#)— Créez une file d'attente à utiliser lors du traitement des demandes de placement de session de jeu.
- [DescribeGameSessionQueues](#)— Récupère les files d'attente des sessions de jeu définies dans une Amazon GameLift Servers région.
- [UpdateGameSessionQueue](#)— Modifie la configuration d'une file d'attente de session de jeu.

- [DeleteGameSessionQueue](#)— Supprime une file d'attente de session de jeu de la région.

Gérer les alias

Utilisez des alias pour représenter vos flottes ou créer une destination alternative de terminal. Les alias sont utiles lors de la transition de l'activité du jeu d'une flotte à une autre, par exemple lors des mises à jour des builds du serveur de jeu.

- [CreateAlias](#)— Définissez un nouvel alias et attribuez-le éventuellement à une flotte.
- [ListAliases](#)— Obtenez tous les alias de flotte définis dans une Amazon GameLift Servers région.
- [DescribeAlias](#)— Récupère des informations sur un alias existant.
- [UpdateAlias](#)— Modifiez les paramètres d'un alias, par exemple en le redirigeant d'un parc à un autre.
- [DeleteAlias](#)— Supprime un alias de la région.
- [ResolveAlias](#)— Obtenez l'identifiant de flotte vers lequel pointe un alias spécifié.

Connectez-vous à des instances d'hébergement gérées

Affichez des informations sur les instances individuelles d'une flotte ou demandez un accès à distance à une instance de flotte spécifiée pour le dépannage.

- [DescribeInstances](#)— Obtenez des informations sur chaque instance d'un parc, notamment l'ID, l'adresse IP, l'emplacement et le statut de l'instance.
- [GetInstanceAccess](#)— Demandez les informations d'accès nécessaires pour vous connecter à distance à une instance spécifiée d'un parc.

Configurer l'appairage de VPC

Créez et gérez des connexions de peering VPC entre vos ressources Amazon GameLift Servers d'hébergement et d'autres ressources. AWS

- [CreateVpcPeeringAuthorization](#)— Autorisez une connexion de peering avec l'un de vos VPCs.
- [DescribeVpcPeeringAuthorizations](#)— Récupère les autorisations de connexion d'appairage valides.
- [DeleteVpcPeeringAuthorization](#)— Supprime une autorisation de connexion de peering.
- [CreateVpcPeeringConnection](#)— Établissez une connexion de peering entre le VPC d'Amazon GameLift Servers une flotte et l'un des vôtres. VPCs

- [DescribeVpcPeeringConnections](#)— Récupérez des informations sur les connexions de peering VPC actives ou en attente avec une flotte. Amazon GameLift Servers
- [DeleteVpcPeeringConnection](#)— Supprime une connexion de peering VPC avec une flotte. Amazon GameLift Servers

Démarrez des sessions de jeu et rejoignez des joueurs

Appelez ces opérations depuis un service principal pour démarrer de nouvelles sessions de jeu, obtenir des informations sur les sessions de jeu existantes et rejoindre des joueurs à des sessions de jeu. Ces opérations sont destinées à être utilisées avec des serveurs de jeu personnalisés hébergés sur Amazon GameLift Servers. Si vous utilisez Amazon GameLift ServersRealtime, gérez les sessions de jeu à l'aide du [Amazon GameLift ServersRealtime référence de l'API client \(C#\)](#).

- Démarrage de nouvelles sessions de jeu pour un ou plusieurs joueurs.
 - [StartGameSessionPlacement](#)— Demandez Amazon GameLift Servers à trouver les meilleures ressources d'hébergement disponibles et lancez une nouvelle session de jeu. Il s'agit de la méthode préférée pour créer de nouvelles sessions de jeu. Il s'appuie sur les files d'attente des sessions de jeu pour suivre la disponibilité de l'hébergement dans plusieurs régions et utilise des FleetIQ algorithmes pour hiérarchiser les emplacements en fonction de la latence des joueurs, du coût d'hébergement, de la localisation, etc.
 - [DescribeGameSessionPlacement](#)— Obtenez les détails et le statut d'une demande de placement.
 - [StopGameSessionPlacement](#)— Annulez une demande de placement.
 - [CreateGameSession](#)— Démarrez une nouvelle session de jeu vide sur un emplacement de flotte spécifique. Cette opération vous permet de mieux contrôler le point de départ de la session de jeu, au lieu de l'utiliser FleetIQ pour évaluer les options de placement. Vous devez ajouter des joueurs à la nouvelle session de jeu lors d'une étape séparée.
- Faites participer les joueurs à des sessions de jeu existantes. Trouvez des sessions de jeu avec des emplacements de jeu disponibles et réservez-les aux nouveaux joueurs.
 - [CreatePlayerSession](#)— Réservez une place libre pour qu'un joueur puisse rejoindre une session de jeu.
 - [CreatePlayerSessions](#)— Réservez des places libres pour que plusieurs joueurs puissent participer à une session de jeu.
- Utilisation avec des données de session de jeu et de session de joueur. Gérez les informations relatives aux sessions de jeu et aux sessions des joueurs.

- [SearchGameSessions](#)— Demandez une liste des sessions de jeu actives en fonction d'un ensemble de critères de recherche.
- [DescribeGameSessions](#)— Récupérez les métadonnées pour des sessions de jeu spécifiques, y compris la durée d'activité et le nombre actuel de joueurs.
- [DescribeGameSessionDetails](#)— Récupérez les métadonnées, y compris le paramètre de protection des sessions de jeu, pour une ou plusieurs sessions de jeu.
- [DescribePlayerSessions](#)— Obtenez des informations sur l'activité des joueurs, notamment leur statut, leur temps de jeu et leurs données.
- [UpdateGameSession](#)— Modifiez les paramètres de session de jeu, tels que le nombre maximum de joueurs et la politique d'inscription.
- [GetGameSessionLogUrl](#)— Obtenez l'emplacement des journaux enregistrés pour une session de jeu.

SDK de serveur 5.x pour Amazon GameLift Servers

Cette section fournit une documentation de référence pour le SDK 5.x du serveur pour Amazon GameLift Servers. Le SDK du serveur fournit les fonctionnalités de base que votre serveur de jeu utilise pour interagir avec le Amazon GameLift Servers service. Par exemple, votre serveur de jeu reçoit des instructions du service pour démarrer et arrêter des sessions de jeu et fournit régulièrement des mises à jour de l'état des sessions de jeu au service. Intégrez vos serveurs de jeu au SDK du serveur avant de les déployer pour l'hébergement.

Utilisez cette référence au SDK de serveur pour intégrer vos serveurs de jeux multijoueurs personnalisés à des fins d'Amazon GameLift Servers hébergement. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

La dernière version majeure du SDK du serveur pour Amazon GameLift Servers est la version 5.x. Les fonctionnalités d'hébergement suivantes nécessitent l'utilisation de la version 5.x :

- Amazon GameLift Servers N'importe où
- Amazon GameLift Servers plugin pour Unreal Engine et Unity

Rubriques

- [Mises à jour du SDK 5 du serveur pour Amazon GameLift Servers](#)
- [Migrer vers le SDK 5.x du serveur pour Amazon GameLift Servers](#)

- [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#)
- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)
- [SDK du serveur Go pour Amazon GameLift Servers -- Actions](#)
- [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)

Mises à jour du SDK 5 du serveur pour Amazon GameLift Servers

Vos serveurs de jeu hébergés utilisent le SDK du serveur pour communiquer avec le Amazon GameLift Servers service Amazon GameLift Servers afin de démarrer et de gérer les sessions de jeu pour les joueurs. La dernière version, le SDK 5 pour Amazon GameLift Servers serveurs, propose un certain nombre d'améliorations et prend en charge de nouvelles Amazon GameLift Servers fonctionnalités. Si la version de votre serveur de jeu utilise Amazon GameLift Servers actuellement le SDK 4 ou une version antérieure, suivez les instructions de cette rubrique pour mettre à jour vos jeux.

Amazon GameLift ServersLes versions 5.0.0 et supérieures du SDK du serveur incluent les mises à jour suivantes :

- Langages étendus — Les bibliothèques sont disponibles dans les langages suivants : C++, C#, Go. Vous pouvez créer les bibliothèques C++ à utiliser avec Unreal Engine.
- Support des plugins pour moteurs de jeu — Les plugins Amazon GameLift Servers autonomes pour Unreal Engine et Unity nécessitent des bibliothèques Amazon GameLift Servers du SDK 5 pour le serveur. Ces plugins proposent des flux de travail guidés pour intégrer, tester et déployer vos jeux à des Amazon GameLift Servers fins d'hébergement. Voir [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#) et [Amazon GameLift Serversplugin pour Unreal Engine documentation](#).
- Amazon GameLift ServersAssistance partout — Avec les flottes Anywhere, vous pouvez configurer vos propres ressources d'hébergement pour utiliser les Amazon GameLift Servers fonctionnalités (y compris le matchmaking). Ajoutez l'Amazon GameLift Serversagent pour automatiser la gestion du cycle de vie des sessions de jeu. Utilisez les flottes Anywhere pour l'hébergement de production avec du matériel sur site, ou configurez des environnements de test pour un développement itératif rapide de jeux. Voir [Hébergement n'importe où](#) et l'[Amazon GameLift Serversagent](#).
- Outils de test mis à jour — La fonctionnalité Amazon GameLift Servers Anywhere vous permet de configurer des environnements de test locaux ou basés sur le cloud pour vos jeux. Configurez les tests avec ou sans l'Amazon GameLift Serversagent. Ces outils remplacent Amazon GameLift Servers Local. Consultez [Configurez des tests locaux avec Amazon GameLift Servers Anywhere](#).

- Solution .NET consolidée pour C# — Le SDK 5.1+ du serveur C# prend en charge .NET Framework 4.6.2 (mis à niveau depuis la version 4.6.1) et .NET 6.0 dans une seule solution. .NET Standard 2.1 est disponible avec les bibliothèques créées par Unity.
- Nouvelle Compute ressource — Cette nouvelle ressource combine différents types de ressources d'hébergement. Il inclut des ressources d'hébergement basées sur le cloud (flottes gérées EC2 ou de conteneurs) et des ressources d'hébergement contrôlées par le client (flottes Anywhere). Il inclut les mises à jour suivantes :
 - Les nouveaux appels d'API pour la Compute ressource incluent : [ListCompute\(\)](#), [DescribeCompute\(\)](#) et [GetComputeAccess\(\)](#). Ces actions renvoient des informations sur les ressources d'hébergement pour tout type de Amazon GameLift Servers flotte. En général, pour les flottes dotées de serveurs de jeux utilisant le SDK 5.x du serveur, utilisez les actions spécifiques au calcul pour remplacer les actions spécifiques à l'instance. En outre, ces actions sont destinées à être utilisées dans les flottes Anywhere sans l'Amazon GameLift Serversagent : [RegisterCompute\(\)](#), [DeregisterCompute\(\)](#) et [GetComputeAuthToken\(\)](#).
 - Nouvelle métrique ActiveCompute avec CloudWatch dimensions FleetId, Location, et ComputeType. Cette métrique remplace la métrique précédente ActiveInstances.
- Amazon EC2 Systems Manager (SSM) pour l'accès à distance : pour plus de sécurité, utilisez SSM au lieu de SSH lorsque vous vous connectez à des instances de flottes gérées. Amazon GameLift Servers Consultez [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).

Migrer vers le SDK 5.x du serveur pour Amazon GameLift Servers

Pour mettre à jour un projet de jeu afin d'utiliser la version 5.x du SDK du serveur, apportez les modifications suivantes :

1. Procurez-vous le dernier SDK de serveur pour le Amazon GameLift Servers package pour votre environnement de développement [[Site de téléchargement](#)]. Suivez les instructions d'installation figurant dans le Readme fichier correspondant au package et à la version que vous avez téléchargés. Consultez ces instructions pour utiliser le serveur SDKs avec votre projet de jeu.
 - [Pour les environnements de développement utilisant C++, C# ou Go](#)
 - [Pour les projets Unreal Engine \(SDK du serveur C++ pour les bibliothèques Unreal uniquement\)](#)
 - [Pour les projets Unity \(SDK du serveur C# pour les bibliothèques Unity uniquement\)](#)

- [À utiliser avec le Amazon GameLift Servers plugin pour Unreal Engine](#)
- [À utiliser avec le Amazon GameLift Servers plugin pour Unity](#)

2. Mettez à jour le code de votre serveur comme suit :

- Remplacez la fonction de rappel du code du serveur par `CreateGameSession()`.
`onStartGameSession()`
- Mettez à jour les `InitSDK()` entrées comme il convient :
 - Si vous envisagez de déployer la version du serveur de jeu sur une EC2 flotte Amazon GameLift Servers gérée ou sur une flotte Anywhere avec l'Amazon GameLift Serversagent :

Appel `InitSDK()` sans paramètres ([C++](#)) ([C#](#)) ([Unreal](#)). Cet appel définit l'environnement informatique et une WebSocket connexion au Amazon GameLift Servers service.

- Si vous envisagez de déployer la version du serveur de jeu sur une flotte Anywhere sans l'Amazon GameLift Serversagent :

Appel `InitSDK()` avec les paramètres du serveur ([C++](#)) ([C#](#)) ([Unreal](#)). Un processus de serveur de jeu utilise ces paramètres pour établir une connexion avec le Amazon GameLift Servers service.

3. Si la version de votre serveur de jeu ou d'autres applications hébergées communiquent avec d'autres AWS ressources en cours d'exécution, vous devez modifier la manière dont l'application accède à ces ressources. Remplacez l'utilisation de `AssumeRoleCredentials` par l'action du nouveau SDK du serveur `GetFleetRoleCredentials()` (pour les serveurs de jeux) ou utilisez des informations d'identification partagées (pour les autres applications). Pour en savoir plus sur la mise en œuvre de cette modification, consultez [Communiquez avec les autres AWS ressources de vos flottes](#).
4. Si votre projet a appelé l'action du SDK du serveur `GetInstanceCertificate()` pour récupérer un certificat TLS, modifiez votre code pour utiliser le nouveau `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#)) à la place.
5. Lorsque vous téléchargez la version de votre jeu vers Amazon GameLift Servers (par exemple avec [upload-build](#) ou [CreateBuild\(\)](#)), définissez le `ServerSdkVersion` paramètre sur la version 5.x que vous utilisez (ce paramètre est actuellement 4.0.2 par défaut). Ce paramètre doit correspondre aux bibliothèques du SDK du serveur présentes dans la version du serveur de jeu. Si vous spécifiez une version incorrecte pour une version de serveur de jeu téléchargée, toutes les flottes que vous créerez avec cette version échoueront. Consultez [Déployez un serveur personnalisé conçu pour Amazon GameLift Servers hébergement](#).

L'exemple suivant montre comment spécifier la version du SDK du serveur :

```
aws gamelift upload-build \  
  --operating-system AMAZON_LINUX_2023 \  
  --server-sdk-version "5.0.0" \  
  --build-root "~/mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

6. Si vous utilisez des scripts pour vous connecter à distance à des flottes gérées, mettez-les à jour pour utiliser le nouveau processus, comme décrit dans [Connectez-vous à distance à Amazon GameLift Servers instances de flotte](#).

SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions

Utilisez la référence du SDK 5.x du serveur pour intégrer votre jeu multijoueur à des fins d'hébergement. Amazon GameLift Servers Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette rubrique décrit l'API Amazon GameLift Servers C++ que vous pouvez utiliser lorsque vous créez avec la bibliothèque standard C++ (std). Plus précisément, cette documentation s'applique au code que vous compilez avec l'-DDGAMELIFT_USE_STD=1option.

SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Types de données

Utilisez la référence du SDK 5.x du serveur Amazon GameLift Servers C++ pour intégrer votre jeu multijoueur à des fins d'hébergement. Amazon GameLift Servers Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette rubrique décrit l'API Amazon GameLift Servers C++ que vous pouvez utiliser lorsque vous créez avec la bibliothèque standard C++ (std). Plus précisément, cette documentation s'applique au code que vous compilez avec l'-DDGAMELIFT_USE_STD=1option.

[SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#)

Types de données

- [LogParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Joueur](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [AttributeValue](#)
- [GetFleetRoleCredentialsRequest](#)
- [AwsLongOutcome](#)
- [AwsStringOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [GenericOutcome](#)
- [GenericOutcomeCallable](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [Initialiser SDKOutcome](#)
- [GameLiftError](#)
- [Enums](#)

LogParameters

Objet identifiant les fichiers générés au cours d'une session de jeu que vous Amazon GameLift Servers souhaitez télécharger et stocker à la fin de la session de jeu. Le serveur de jeu fournit `LogParameters` dans Amazon GameLift Servers le cadre d'un `ProcessParameters` objet lors d'un [ProcessReady\(\)](#) appel.

Propriétés	Description
LogPaths	La liste des chemins de répertoire vers les fichiers journaux du serveur de jeu que vous Amazon GameLift Servers souhaitez stocker pour un accès futur. Le processus du serveur génère ces fichiers lors de chaque session de jeu. Vous définissez les chemins et les noms des fichiers sur votre serveur de jeu et vous les stockez dans le répertoire racine du jeu. Les chemins du journal doivent être absolus. Par exemple, si votre build de jeu stocke les journaux de session de jeu dans un chemin tel que <code>MyGame\sessionLogs\</code>, le chemin se <code>c:\game\MyGame\sessionLogs</code> trouve sur une instance Windows. Type : <code>std::vector<std::string></code> Obligatoire : non

ProcessParameters

Ce type de données contient l'ensemble des paramètres envoyés à Amazon GameLift Servers dans un [ProcessReady\(\)](#).

Propriétés	Description
LogParameters	Objet avec des chemins de répertoire vers des fichiers générés au cours d'une session de jeu.

	Amazon GameLift Servers copie et stocke les fichiers pour un accès futur. Type : <code>Aws::GameLift::Server::</code> LogParameters Obligatoire : non
OnHealthCheck	La fonction de rappel qui Amazon GameLift Servers appelle le processus du serveur pour demander un rapport d'état de santé. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes et attend une réponse pendant 60 secondes. Le processus du serveur revient TRUE s'il est sain, FALSE sinon sain. Si aucune réponse n'est renvoyée, Amazon GameLift Servers enregistre le processus du serveur comme non sain. Type : <code>std::function<bool()></code> <code>onHealthCheck</code> Obligatoire : non
OnProcessTerminate	La fonction de rappel qui Amazon GameLift Servers invoque pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attendez 5 minutes que le processus serveur s'arrête et répondez par un ProcessEnding() appel avant d'arrêter le processus serveur. Type : <code>std::function<void()></code> <code>onProcessTerminate</code> Obligatoire : oui

OnStartGameSession

La fonction de rappel qui Amazon GameLift Servers invoque une nouvelle session de jeu pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à une demande du client [CreateGameSession](#). La fonction de rappel transmet un [GameSession](#) objet.

```
Type : const std::function<void  
(Aws::GameLift::Model::Game  
Session)> onStartGameSession
```

Obligatoire : oui

OnUpdateGameSession

La fonction de rappel qui Amazon GameLift Servers appelle pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de remplissage de correspondances a été traitée pour fournir des données de matchmaking mises à jour. Il transmet un [GameSession](#) objet, une mise à jour de statut (`updateReason`) et l'identifiant du ticket de remplacement des matchs.

```
Type : std::function<void(Aws::Gam  
eLift::Server::Model::Updat  
eGameSession)> onUpdateG  
ameSession
```

Obligatoire : non

Port

Numéro de port sur lequel le processus serveur écoute les connexions des nouveaux joueurs. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur.

Type : Integer

Obligatoire : oui

UpdateGameSession

Ce type de données est mis à jour vers un objet de session de jeu, qui inclut la raison pour laquelle la session de jeu a été mise à jour et l'identifiant du ticket de remplissage associé si le remplissage est utilisé pour remplir les sessions des joueurs pendant la session de jeu.

Propriétés	Description
GameSession	Un objet GameSession. L'GameSession objet contient des propriétés décrivant une session de jeu. Type : <code>Aws::GameLift::Server::GameSession</code> Obligatoire : oui
UpdateReason	La raison pour laquelle la session de jeu est mise à jour. Type : <code>Aws::GameLift::Server::UpdateReason</code> Obligatoire : oui

Propriétés	Description
BackfillTicketId	L'identifiant du ticket de remplacement qui tente de mettre à jour la session de jeu. Type : <code>std::string</code> Obligatoire : non

GameSession

Ce type de données fournit les détails d'une session de jeu.

Propriétés	Description
GameSessionId	Identifiant unique pour la session de jeu. L'ARN d'une session de jeu a le format suivant : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Type : <code>std::string</code> Obligatoire : non
Nom	Une étiquette descriptive de la session de jeu. Type : <code>std::string</code> Obligatoire : non
FleetId	Identifiant unique de la flotte sur laquelle s'exécute la session de jeu. Type : <code>std::string</code> Obligatoire : non
MaximumPlayerSessionCount	Le nombre maximum de connexions de joueurs à la session de jeu.

Propriétés	Description
	Type : <code>int</code> Obligatoire : non
Port	Le numéro de port de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : <code>int</code> Obligatoire : non
IpAddress	Adresse IP de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : <code>std::string</code> Obligatoire : non
GameSessionData	Ensemble de propriétés de session de jeu personnalisées, mises en forme en tant que valeur de chaîne unique. Type : <code>std::string</code> Obligatoire : non

Propriétés	Description
MatchmakerData	Informations sur le processus de matchmaking utilisé pour créer la session de jeu, en syntaxe JSON, formatée sous forme de chaîne. En plus de la configuration de matchmaking utilisée, il contient des données sur tous les joueurs affectés au match, y compris les attributs des joueurs et les affectations des équipes. Type : <code>std::string</code> Obligatoire : non
GameProperties	Ensemble de propriétés personnalisées pour une session de jeu, formatées sous forme de paires clé:valeur. Ces propriétés sont transmises avec une demande de démarrage d'une nouvelle session de jeu. Type : <code>std :: vector < GameProperty ></code> Obligatoire : non

Propriétés	Description
DnsName	Identifiant DNS attribué à l'instance qui exécute la session de jeu. Les valeurs ont le format suivant : • Flottes compatibles TLS : <code><unique identifieur>.<region identifiée>.amazongamelift.com</code> • Non-TLS-enabled flottes : <code>ec2-<unique identifieur>.compute.amazonaws.com</code> . Lorsque vous vous connectez à une session de jeu exécutée sur une flotte compatible TLS, vous devez utiliser le nom DNS et non l'adresse IP. Type : <code>std::string</code> Obligatoire : non

ServerParameters

Informations utilisées par un serveur de jeu pour établir une connexion avec le Amazon GameLift Servers service. N'incluez ces paramètres lors de l'appel [InitSDK\(\)](#) que si la version du serveur de jeu doit être déployée sur une flotte Anywhere ou une flotte de conteneurs sans l'Amazon GameLift Serversagent. Pour tous les autres scénarios de déploiement, appelez [InitSDK\(\)](#) sans paramètres.

Propriétés	Description
websocketUrl	Les <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers retours lorsque vous RegisterCompute recherchez une ressource de calcul Amazon GameLift Servers Anywhere. Type : <code>std::string</code>

Propriétés	Description
	Obligatoire : oui
ID du processus	Un identifiant unique enregistré auprès du processus serveur hébergeant votre jeu. Type : <code>std::string</code> Obligatoire : oui
hostId	Le HostID est celui ComputeName utilisé lorsque vous avez enregistré votre ordinateur. Pour plus d'informations, voir, RegisterCompute. Type : <code>std::string</code> Obligatoire : oui
ID de flotte	Identifiant unique de la flotte dans laquelle le calcul est enregistré. Pour plus d'informations, voir, RegisterCompute. Type : <code>std::string</code> Obligatoire : oui
Jeton d'authentification	Le jeton d'authentification généré par Amazon GameLift Servers celui-ci authentifie votre serveur auprès de Amazon GameLift Servers. Pour plus d'informations, voir, GetComputeAuthToken. Type : <code>std::string</code> Obligatoire : oui

StartMatchBackfillRequest

Informations utilisées pour créer une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations Amazon GameLift Servers lors d'un [StartMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Un identifiant de session de jeu unique. L'opération API GetGameSessionId renvoie l'identifiant au format ARN. Type : <code>std::string</code> Obligatoire : oui
MatchmakingConfigurationArn	Un identifiant unique, sous la forme d'un ARN, que le système de jumelage doit utiliser pour cette demande. L'ARN de matchmaking pour la session de jeu d'origine se trouve dans l'objet de session de jeu dans la propriété de données du matchmaker. Pour en savoir plus sur les données du système de jumelage, consultez la section Travailler avec les données du système de jumelage. Type : <code>std::string</code> Obligatoire : oui
Joueurs	Ensemble de données représentant tous les joueurs participant à la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Type : <code>std::vector<Player></code> Obligatoire : oui

Propriétés	Description
TicketId	Un identifiant unique pour un ticket de demande de matchmaking ou de remplacement de match. Si vous ne fournissez aucune valeur, il en Amazon GameLift Servers génère une. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire. Type : <code>std::string</code> Obligatoire : non

Joueur

Ce type de données représente un joueur dans le matchmaking. Lors du lancement d'une demande de matchmaking, un joueur dispose d'un identifiant de joueur, d'attributs et éventuellement de données de latence. Amazon GameLift Servers ajoute des informations sur l'équipe après la fin d'un match.

Propriétés	Description
LatencyInMS	Ensemble de valeurs exprimées en millisecondes qui indiquent le niveau de latence ressenti par un joueur lorsqu'il est connecté à un lieu. Si cette propriété est utilisée, le joueur n'est jumelé qu'aux emplacements répertoriés. Si un matchmaker dispose d'une règle qui évalue la latence, les joueurs doivent indiquer la latence pour être mis en relation. Type : <code>Dictionary<string,int></code> Obligatoire : non

Propriétés	Description
PlayerAttributes	Une collection de paires clé:valeur contenant des informations sur les joueurs à utiliser dans le matchmaking. Les clés d'attribut du joueur doivent correspondre à PlayerAttributes celles utilisées dans un ensemble de règles de matchmaking. Pour plus d'informations sur les attributs des joueurs, consultez AttributeValue. Type : <code>std::map<std::string, AttributeValue></code> Obligatoire : non
PlayerId	Identifiant unique pour un joueur. Type : <code>std::string</code> Obligatoire : non
Equipe	Le nom de l'équipe à laquelle le joueur est affecté lors d'un match. Vous définissez le nom de l'équipe dans le jeu de règles de matchmaking. Type : <code>std::string</code> Obligatoire : non

DescribePlayerSessionsRequest

Objet qui indique les sessions de joueur à récupérer. Le processus du serveur fournit ces informations en [DescribePlayerSessions\(\)](#) appelant Amazon GameLift Servers.

Propriétés	Description
GameSessionId	Un identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'identifiant de session de jeu est <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>. GameSessionID Il s'agit d'une chaîne d'identification personnalisée ou d'un Type : <code>std::string</code> Obligatoire : non
PlayerSessionId	Identifiant unique d'une session de joueur. Utilisez ce paramètre pour demander une session de joueur spécifique. Type : <code>std::string</code> Obligatoire : non
PlayerId	L'identifiant unique d'un joueur. Utilisez ce paramètre pour demander toutes les sessions de joueur pour un joueur spécifique. Consultez Générer un joueur IDs. Type : <code>std::string</code> Obligatoire : non
PlayerSessionStatusFilter	État de la session du joueur sur lequel filtrer les résultats. Les statuts possibles des sessions de joueur sont les suivants : • RÉSERVÉ — La demande de session du joueur a été reçue, mais le joueur ne s'est

Propriétés	Description
	pas connecté au processus du serveur ni n'a été validé. • ACTIF — Le joueur a été validé par le processus du serveur et est connecté. • TERMINÉ — La connexion du joueur a été interrompue. • TIMEDOUT — Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté ou n'a pas été validée dans le délai imparti (60 secondes). Type : <code>std::string</code> Obligatoire : non
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : <code>std::string</code> Obligatoire : non
Limite	Nombre maximal de résultats à renvoyer. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : <code>int</code> Obligatoire : non

StopMatchBackfillRequest

Informations utilisées pour annuler une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Identifiant de session de jeu unique de la demande annulée. Type : <code>char[]</code> Obligatoire : non
MatchmakingConfigurationArn	Un identifiant unique du matchmaker auquel cette demande a été envoyée. Type : <code>char[]</code> Obligatoire : non
TicketId	Identifiant unique du ticket de demande de remblayage à annuler. Type : <code>char[]</code> Obligatoire : non

AttributeValue

Utilisez ces valeurs dans les paires [Joueur](#) clé-valeur d'attribut. Cet objet vous permet de spécifier une valeur d'attribut à l'aide de n'importe quel type de données valide : chaîne, nombre, tableau de chaînes ou mappage de données. Chaque `AttributeValue` objet doit utiliser exactement l'une des propriétés disponibles : `SN`, `SL`, ou `SDM`.

Propriétés	Description
AttrType	Spécifie le type de valeur d'attribut. Les types de valeurs d'attributs possibles sont les suivants :

Propriétés	Description
	• NONE • CHAÎNE • DOUBLE • LISTE_CHAÎNES • STRING_DOUBLE_MAP Obligatoire : non
S	Représente une valeur d'attribut de chaîne. Type : <code>std::string</code> Obligatoire : non
N	Représente une valeur d'attribut numérique. Type : <code>double</code> Obligatoire : non
SL	Représente un tableau de valeurs d'attributs de chaîne. Type : <code>std::vector<std::string></code> Obligatoire : non
SDM	Représente un dictionnaire de clés de chaîne et de valeurs doubles. Type : <code>std::map<std::string, double></code> Obligatoire : non

GetFleetRoleCredentialsRequest

Ce type de données donne au serveur de jeu un accès limité à vos autres AWS ressources. Pour plus d'informations, voir, [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Propriétés	Description
RoleArn	Le nom de ressource Amazon (ARN) du rôle de service qui étend un accès limité à vos AWS ressources. Type : <code>std::string</code> Obligatoire : non
RoleSessionName	Nom de session de rôle que vous pouvez utiliser pour identifier une AWS Security Token Service AssumeRole session de manière unique. Ce nom est exposé dans les journaux d'audit tels que ceux contenus dans CloudTrail. Type : <code>std::string</code> Obligatoire : non

AwsLongOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : <code>long</code> Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>long&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non.

Propriétés	Description
	Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

AwsStringOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : std::string Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : long&& Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError"

Propriétés	Description
	Obligatoire : non

DescribePlayerSessionsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “DescribePlayerSessionsResult” Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “GameLiftError” Obligatoire : non

DescribePlayerSessionsResult

Collection d'objets contenant les propriétés de chaque session de joueur correspondant à la demande.

Propriétés	Description
NextToken	Un jeton qui indique le début de la prochaine page séquentielle de résultats. Utilisez le jeton renvoyé lors d'un appel précédent à cette opération. Pour commencer au début du jeu de résultats, ne spécifiez aucune valeur. Si un ID de session de joueur est spécifié, ce paramètre est ignoré. Type : <code>std::string</code> Obligatoire : oui
PlayerSessions	Type : <code>IList<the section called "PlayerSession"></code> Obligatoire :
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>std::string&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GenericOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GenericOutcomeCallable

Ce type de données est un résultat générique asynchrone. Il comprend les propriétés suivantes :

Propriétés	Description
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

PlayerSession

Ce type de données représente une session de joueur Amazon GameLift Servers transmise au serveur de jeu. Pour de plus amples informations, veuillez consulter [PlayerSession](#).

Propriétés	Description
CreationTime	Type : long Obligatoire : non
FleetId	Type : std::string Obligatoire : non
GameSessionId	Type : std::string Obligatoire : non
IpAddress	Type : std::string Obligatoire : non
PlayerData	Type : std::string Obligatoire : non
PlayerId	Type : std::string Obligatoire : non
PlayerSessionId	Type : std::string Obligatoire : non
Port	Type : int Obligatoire : non
Statut	État de session de joueur pour filtrer les résultats. Lorsqu'un PlayerSessionId ou PlayerId est fourni, cela n' PlayerSessionStatusFilter a aucun effet sur la réponse. Type : Une PlayerSessionStatus énumération. Les valeurs possibles sont notamment les suivantes : • ACTIF

Propriétés	Description
	• TERMINÉ • NOT_SET • RÉSERVÉ • DÉLAI EXPIRÉ Obligatoire : non
TerminationTime	Type : long Obligatoire : non
DnsName	Type : std::string Obligatoire : non

StartMatchBackfillOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “StartMatchBackfillResult” Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : StartMatchBackfillResult&& Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool

Propriétés	Description
	Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

StartMatchBackfillResult

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
TicketId	Un identifiant unique pour un ticket de matchmaking. Si aucun identifiant de ticket n'est spécifié ici, il en Amazon GameLift Servers générera un sous la forme d'un UUID. Utilisez cet identifiant pour suivre l'état du ticket de remplacement des matchs et récupérer les résultats des matchs. Type : <code>std::string</code> Obligatoire : non

GetComputeCertificateOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called "GetComputeCertificateResult" Obligatoire : non

Propriétés	Description
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>Aws::GameLift::Server::Model::GetComputeCertificateResult&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GetComputeCertificateResult

Le chemin d'accès au certificat TLS sur votre ordinateur et le nom d'hôte du calcul.

Propriétés	Description
CertificatePath	Le chemin d'accès au certificat TLS sur votre ressource de calcul. Lorsque vous utilisez une flotte Amazon GameLift Servers gérée, ce chemin contient : <code>certificate.pem</code> : le certificat de l'utilisateur final. La chaîne de certificats complète est la combinaison des <code>certificateChain.pem</code> éléments ajoutés à ce certificat. <code>certificateChain.pem</code> : chaîne de certificats qui contient le certificat racine et les certificats intermédiaires. <code>rootCertificate.pem</code> : le certificat racine.

Propriétés	Description
	<code>privateKey.pem</code> : clé privée pour le certificat d'utilisateur final. Type : <code>std::string</code> Obligatoire : non
ComputeName	Le nom de votre ressource de calcul. Type : <code>std::string</code> Obligatoire : non

GetFleetRoleCredentialsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “GetFleetRoleCredentialsResult” Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui

Propriétés	Description
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GetFleetRoleCredentialsResult

Propriétés	Description
AccessKeyId	ID de clé d'accès permettant d'authentifier et de fournir un accès à vos AWS ressources. Type : <code>string</code> Obligatoire : non
AssumedRoleId	ID de l'utilisateur auquel appartient le rôle de service. Type : <code>string</code> Obligatoire : non
AssumedRoleUserArn	Le nom de ressource Amazon (ARN) de l'utilisateur auquel appartient le rôle de service. Type : <code>string</code> Obligatoire : non
Expiration	Durée avant l'expiration de vos informations d'identification de session. Type : <code>DateTime</code> Obligatoire : non
SecretAccessKey	ID de clé d'accès secrète pour l'authentification.

Propriétés	Description
	Type : <code>string</code> Obligatoire : non
SessionToken	Un jeton pour identifier la session active en cours qui interagit avec vos AWS ressources. Type : <code>string</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

Initialiser SDKOutcome

Note

`InitSDKOutcome` est renvoyé que lorsque vous créez le SDK avec l'indicateur `std`. Si vous construisez avec le drapeau `nostd`, il [the section called "GenericOutcome"](#) est renvoyé à la place.

Propriétés	Description
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code>

Propriétés	Description
	Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GameLiftError

Propriétés	Description
ErrorType	Type d'erreur. Type : Une GameLiftErrorType énumération . Obligatoire : non
ErrorMessage	Nom de l'erreur. Type : std::string Obligatoire : non
ErrorMessage	Message d'erreur. Type : std::string Obligatoire : non

Enums

Les énumérations définies pour le SDK du serveur pour Amazon GameLift Servers (C++) sont définies comme suit :

GameLiftErrorType

Valeur de chaîne indiquant le type d'erreur. Les valeurs valides sont les suivantes :

- MAUVAISE DEMANDE D'EXCEPTION
- GAMESESSION_ID_NOT_SET — L'ID de session de jeu n'a pas été défini.
- EXCEPTION DE SERVICE INTERNE
- LOCAL_CONNECTION_FAILED — La connexion locale a échoué. Amazon GameLift Servers
- NETWORK_NOT_INITIALIZED — Le réseau n'a pas été initialisé.
- SERVICE_CALL_FAILED — Un appel à un AWS service a échoué.
- ÉCHEC DE CONNEXION WEBSOCKET
- WEBSOCKET_CONNECT_FAILURE_FORBIDDEN
- WEBSOCKET_CONNECT_FAILURE_INVALID_URL
- WEBSOCKET_CONNECT_FAILURE_TIMEOUT
- ALREADY_INITIALIZED — Le Amazon GameLift Servers serveur ou le client a déjà été initialisé avec Initialize ().
- FLEET_MISMATCH — La flotte cible ne correspond pas à la flotte d'une GameSession ou d'une PlayerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED — Le client n'a pas été initialisé. Amazon GameLift Servers
- GAMELIFT_SERVER_NOT_INITIALIZED — Le serveur n'a pas été initialisé. Amazon GameLift Servers
- GAME_SESSION_ENDED_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin de la session de jeu.
- GAME_SESSION_NOT_READY — La Amazon GameLift Servers session de jeu sur le serveur n'a pas été activée.
- GAME_SESSION_READY_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que la session de jeu est prête.
- INITIALIZATION_MISMATCH — Une méthode client a été appelée après Server : :Initialize (), ou vice versa.
- NOT_INITIALIZED — Le Amazon GameLift Servers serveur ou le client n'a pas été initialisé avec Initialize ().
- NO_TARGET_ALIASID_SET — Aucun AliasID cible n'a été défini.
- NO_TARGET_FLEET_SET — Aucune flotte cible n'a été définie.
- PROCESS_ENDING_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin du processus.

- `PROCESS_NOT_ACTIVE` — Le processus du serveur n'est pas encore actif, n'est pas lié à un et ne peut ni `GameSession` accepter ni traiter. `PlayerSessions`
- `PROCESS_NOT_READY` — Le processus du serveur n'est pas encore prêt à être activé.
- `PROCESS_READY_FAILED` : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que le processus est prêt.
- `SDK_VERSION_DETECTION_FAILED` — La détection de la version du SDK a échoué.
- `STX_CALL_FAILED` — Un appel au composant principal du XStx serveur a échoué.
- `STX_INITIALIZATION_FAILED` — Le composant principal du XStx serveur n'a pas pu être initialisé.
- `UNEXPECTED_PLAYER_SESSION` — Le serveur a détecté une session de joueur non enregistrée.
- `ÉCHEC DE CONNEXION WEBSOCKET`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_INVALID_URL`
- `WEBSOCKET_CONNECT_FAILURE_TIMEOUT`
- `WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE` — Échec récupérable lors de l'envoi d'un message au service. `GameLift WebSocket`
- `WEBSOCKET_SEND_MESSAGE_FAILURE` — Impossible d'envoyer un message au service. `GameLift WebSocket`
- `MATCH_BACKFILL_REQUEST_VALIDATION` — La validation de la demande a échoué.
- `PLAYER_SESSION_REQUEST_VALIDATION` — La validation de la demande a échoué.

`PlayerSessionCreationPolicy`

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs. Les valeurs valides sont les suivantes :

- `ACCEPT_ALL` — Accepte toutes les sessions de nouveaux joueurs.
- `DENY_ALL` — Refuse toutes les sessions de nouveaux joueurs.
- `NOT_SET` — La session de jeu n'est pas configurée pour accepter ou refuser les sessions de nouveaux joueurs.

[SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Types de données](#)

Rubriques

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Détruire \(\)](#)

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet [the section called “AwsStringOutcome”](#). L'objet renvoyé inclut le numéro de version (exemple `5.0.0`). En cas d'échec, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialise le kit SDK Amazon GameLift Servers. Appelez cette méthode au lancement avant toute autre étape d'initialisation liée à Amazon GameLift Servers. Cette action lit les paramètres du serveur depuis l'environnement hôte pour établir la communication entre le processus du serveur de jeu et le Amazon GameLift Servers service. Il utilise un jeton d'idempotence, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Si la version du serveur de jeu doit être déployée sans l'Amazon GameLift Servers agent sur une flotte ou une flotte de conteneurs Amazon GameLift Servers Anywhere, appelez [InitSDK\(\)](#) et spécifiez un ensemble de paramètres de serveur.

Syntaxe

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

Valeur renvoyée

Renvoie un [the section called “Initialiser SDKOutcome”](#) objet qui indique si le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

exemple

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent  
to enable further communication.  
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =  
    Aws::GameLift::Server::InitSDK();
```

InitSDK()

Initialise le kit SDK Amazon GameLift Servers. Appelez cette méthode au lancement avant toute autre étape d'initialisation liée à Amazon GameLift Servers. Cette action nécessite un ensemble de paramètres de serveur pour établir la communication entre le processus du serveur de jeu et le

Amazon GameLift Servers service. Il utilise un jeton d'idempotence, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Si la version du serveur de jeu doit être déployée sur une EC2 flotte Amazon GameLift Servers gérée, une flotte Amazon GameLift Servers Anywhere ou une flotte de conteneurs avec l'Amazon GameLift Serversagent, appelez [InitSDK\(\)](#) sans paramètres de serveur.

Syntaxe

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

Paramètres

[ServerParameters](#)

Pour initialiser un serveur de jeu sur une flotte Amazon GameLift Servers Anywhere, créez un `ServerParameters` objet avec les informations suivantes :

- URL WebSocket utilisée pour vous connecter à votre serveur de jeu.
- ID du processus utilisé pour héberger votre serveur de jeu.
- L'ID de l'ordinateur hébergeant les processus de votre serveur de jeu.
- L'ID du Amazon GameLift Servers parc contenant votre ordinateur Amazon GameLift Servers Anywhere.
- Le jeton d'autorisation généré par l'Amazon GameLift Serversopération.

Valeur renvoyée

Renvoie un [the section called “Initialiser SDKOutcome”](#) objet qui indique si le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

Note

Si les appels à échouent pour `InitSDK()` les builds de jeu déployés sur des flottes Anywhere, vérifiez le `ServerSdkVersion` paramètre utilisé lors de la création de la ressource de build. Vous devez définir explicitement cette valeur en fonction de la version du SDK du serveur utilisée. La valeur par défaut de ce paramètre est 4.x, ce qui n'est pas compatible. Pour résoudre ce problème, créez une nouvelle version et déployez-la sur une nouvelle flotte.

exemple

Amazon GameLift ServersExemple n'importe où

```
//Define the server parameters
std::string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
std::string processId = "PID1234";
std::string fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
std::string hostId = "HardwareAnywhere";
std::string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
Aws::GameLift::Server::Model::ServerParameters serverParameters =
    Aws::GameLift::Server::Model::ServerParameters(webSocketUrl, authToken, fleetId,
        hostId, processId);

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
Aws::GameLift::Server::InitSDKOutcome initSdkOutcome =
    Aws::GameLift::Server::InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift ServersIndique que le processus du serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après l'avoir invoquée. [InitSDK\(\)](#) Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Paramètres

processParameters

Objet [ProcessParameters](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel implémentées dans le code du serveur de jeu que le Amazon GameLift Servers service invoque pour communiquer avec le processus du serveur.
- Numéro de port sur lequel le processus serveur écoute.

- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre les implémentations de l'appel [ProcessReady\(\)](#) et de la fonction déléguée.

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // Example of a log file written by the
game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths)
    );

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
}
```

```
GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}
```

ProcessReadyAsync()

Informe le service Amazon GameLift Servers que le processus serveur est prêt à héberger des sessions de jeu. Cette méthode doit être appelée une fois que le processus serveur est prêt à héberger une session de jeu. Les paramètres spécifient les noms des fonctions de rappel Amazon GameLift Servers à appeler dans certaines circonstances. Le code du serveur de jeux doit implémenter ces fonctions.

Cet appel est asynchrone. Pour effectuer un appel synchrone, utilisez [ProcessReady\(\)](#). Pour plus d'informations, consultez [Initialiser le processus du serveur](#).

Syntaxe

```
GenericOutcomeCallable ProcessReadyAsync(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Paramètres

processParameters

Objet [ProcessParameters](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel implémentées dans le code du serveur de jeu que le Amazon GameLift Servers service invoque pour communiquer avec le processus du serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // This is an example of a log file
                                                    written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(std::bind(&Server::onStartGameSession, this,
std::placeholders::_1),
    std::bind(&Server::onProcessTerminate, this), std::bind(&Server::OnHealthCheck,
this),
    std::bind(&Server::OnUpdateGameSession, this), listenPort,
    Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);

// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}
```

ProcessEnding()

Amazon GameLift Servers indique que le processus du serveur est en train de se terminer. Appelez cette méthode après toutes les autres tâches de nettoyage (y compris la fermeture de la session de jeu active) et avant de terminer le processus. En fonction du résultat de `ProcessEnding()`, le processus se termine avec succès (0) ou erreur (-1) et génère un événement de flotte. Si le processus se termine par une erreur, l'événement de flotte généré est `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntaxe

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple appelle `ProcessEnding()` et `Destroy()` avant de terminer le processus du serveur avec un code de sortie de réussite ou d'erreur.

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();  
Aws::GameLift::Server::Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess()) {  
    exit(0);  
}  
else {  
    cout << "ProcessEnding() failed. Error: " <<  
        processEndingOutcome.GetError().GetErrorMessage();  
    exit(-1);  
}
```

ActivateGameSession()

Amazon GameLift Servers indique que le processus du serveur a activé une session de jeu et qu'il est maintenant prêt à recevoir les connexions des joueurs. Cette action doit être appelée dans le cadre de la fonction de `onStartGameSession()` rappel, après toute initialisation de session de jeu.

Syntaxe

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =  
    Aws::GameLift::Server::ActivateGameSession();
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple montre l'`ActivateGameSession()` appel dans le cadre de la fonction de `onStartGameSession()` délégation.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)  
{  
    // game-specific tasks when starting a new game session, such as loading map  
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();  
}
```

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur.

Syntaxe

```
GenericOutcome  
UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy  
    newPlayerSessionPolicy);
```

Paramètres

playerCreationSessionPolitique

Type : valeur `PlayerSessionCreationPolicy` [enum](#).

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
Aws::GameLift::GenericOutcome outcome =  
  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

Récupère l'ID de la session de jeu hébergée par le processus serveur actif.

Pour les processus inactifs qui ne sont pas activés lors d'une session de jeu, l'appel renvoie un [the section called "GameLiftError"](#).

Syntaxe

```
AwsStringOutcome GetGameSessionId()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet [the section called "AwsStringOutcome"](#). En cas d'échec, renvoie un message d'erreur.

Pour les processus inactifs qui ne sont pas activés lors d'une session de jeu, l'appel renvoie `Success = True` et `GameSessionId = ""`.

exemple

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

Renvoie l'heure d'arrêt planifiée pour un processus serveur, si une heure de résiliation est disponible. Un processus serveur prend des mesures après avoir reçu un `onProcessTerminate()` rappel

deAmazon GameLift Servers. Amazon GameLift Serversappelle `onProcessTerminate()` pour les raisons suivantes :

- Lorsque le processus du serveur a signalé un mauvais état de santé ou n'a pas réponduAmazon GameLift Servers.
- Lorsque vous mettez fin à l'instance lors d'un événement de réduction de la taille.
- Lorsqu'une instance est interrompue en raison d'une interruption [ponctuelle](#).

Syntaxe

```
AwsDateTimeOutcome GetTerminationTime()
```

Valeur renvoyée

En cas de succès, renvoie l'heure de fin sous forme d'`AwsDateTimeOutcome`objet. La valeur est le délai de résiliation, exprimé en ticks écoulés depuis. `0001 00:00:00` Par exemple, la valeur de la date et de l'heure `2020-09-13 12:26:40 -000Z` est égale à celle `637355968000000000` des ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

Si le processus n'a pas reçu de `ProcessParameters`. `OnProcessTerminate()` rappel, un message d'erreur est renvoyé. Pour plus d'informations sur l'arrêt d'un processus serveur, consultez[Répondre à une notification d'arrêt du processus du serveur](#).

exemple

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift ServersIndique qu'un joueur avec l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Serversvérifie que l'identifiant de session du joueur est valide. Une fois la session du joueur validée, le statut de l'emplacement du joueur Amazon GameLift Servers passe de RÉSERVÉ à ACTIF.

Syntaxe

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple traite une demande de connexion qui inclut la validation et le rejet d'une session de joueur non valide. IDs

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId)
{
    Aws::GameLift::GenericOutcome connectOutcome =
    Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage());
    }
}
```

RemovePlayerSession()

Indique Amazon GameLift Servers qu'un joueur s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers change l'emplacement du joueur pour le rendre disponible.

Syntaxe

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Récupère les données de session du joueur, notamment les paramètres, les métadonnées de session et les données du joueur. Utilisez cette méthode pour obtenir des informations sur les points suivants :

- Une session solo
- Toutes les sessions des joueurs dans une session de jeu
- Toutes les sessions de joueur associées à un identifiant de joueur unique

Syntaxe

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

Paramètres

[DescribePlayerSessionsRequest](#)

[the section called “DescribePlayerSessionsRequest”](#) Objet qui décrit les sessions de joueur à récupérer.

Valeur renvoyée

En cas de réussite, renvoie un objet [the section called “DescribePlayerSessionsOutcome”](#) qui contient un ensemble d'objets de session de joueur correspondant aux paramètres de la demande.

exemple

Cet exemple demande toutes les sessions de joueur activement connectées à une session de jeu spécifiée. En omettant NextToken et en définissant la valeur limite sur 10, Amazon GameLift Servers renvoie les 10 premiers enregistrements de session de joueur correspondant à la demande.

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G
request.SetLimit(10);
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés, Amazon GameLift Servers fournit des données de matchmaking mises à jour à l'aide de la fonction de rappel.

OnUpdateGameSession()

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
    startBackfillRequest);
```

Paramètres

StartMatchBackfillRequest

Un StartMatchBackfillRequest objet qui communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Ces informations sont facultatives ; si aucun identifiant n'est fourni, il en Amazon GameLift Servers générera un.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur se trouve dans les données du matchmaker de la session de jeu.
- ID de la session de jeu à compléter.
- Les données de matchmaking disponibles pour les joueurs actuels de la session de jeu.

Valeur renvoyée

Renvoie un [the section called "StartMatchBackfillOutcome"](#) objet avec l'identifiant du ticket de remplacement correspondant, ou un échec avec un message d'erreur.

exemple

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff"); // optional,
    autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"); //from the game
    session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");           // can use
    GetGameSessionId()
startBackfillRequest.SetPlayers(players);                                 // from the
    game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
```

```
// handle status messages
// perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

Annule une demande de remplacement de match active. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Syntaxe

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Paramètres

[StopMatchBackfillRequest](#)

Un StopMatchBackfillRequest objet identifiant le ticket de matchmaking à annuler :

- L'identifiant du ticket attribué à la demande de remblayage.
- L'entremetteur à qui la demande de remblayage a été envoyée.
- La session de jeu associée à la demande de remplacement.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set backfill stop request parameters

Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;
stopBackfillRequest.SetTicketId("1111aaaa-22bb-33cc-44dd-5555eeee66ff");
stopBackfillRequest.SetGameSessionArn("the game session ARN"); // can use
    GetGameSessionId()
stopBackfillRequest.SetMatchmakingConfigurationArn("arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig");
// from the game session matchmaker data

Aws::GameLift::GenericOutcome stopBackfillOutcome =
```



```
Aws::GameLift::Server::StopMatchBackfill(stopBackfillRequest);
```

GetComputeCertificate()

Récupère le chemin d'accès au certificat TLS utilisé pour chiffrer la connexion réseau entre votre ressource de calcul Amazon GameLift Servers Anywhere et. Amazon GameLift Servers Vous pouvez utiliser le chemin du certificat lorsque vous enregistrez votre appareil informatique dans un parc Amazon GameLift Servers Anywhere. Pour plus d'informations, voir, [RegisterCompute](#).

Syntaxe

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

Valeur renvoyée

Retourne un [the section called “GetComputeCertificateOutcome”](#).

exemple

```
Aws::GameLift::GetComputeCertificateOutcome certificate =  
    Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

Récupère les informations d'identification du rôle IAM qui autorisent l'interaction avec Amazon GameLift Servers d'autres utilisateurs. Services AWS Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

Syntaxe

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest  
    request);
```

Paramètres

[GetFleetRoleCredentialsRequest](#)

Valeur renvoyée

Renvoie un objet [the section called “GetFleetRoleCredentialsOutcome”](#).

exemple

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Cet exemple montre l'utilisation de la `RoleSessionName` valeur facultative pour attribuer un nom à la session d'identification à des fins d'audit. Si vous ne fournissez pas de nom de session de rôle, la valeur par défaut « *[fleet-id]* - *[host-id]* » est utilisée.

```
// form the fleet credentials request
Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest
    getFleetRoleCredentialsRequest;
getFleetRoleCredentialsRequest.SetRoleArn("arn:aws:iam::123456789012:role/service-role/
exampleGameLiftAction");
getFleetRoleCredentialsRequest.SetRoleSessionName("MyFleetRoleSession");

Aws::GameLift::GetFleetRoleCredentialsOutcome credentials =
    Aws::GameLift::Server::GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Détruire ()

Libère de la mémoire le SDK du serveur de Amazon GameLift Servers jeu. Il est recommandé d'appeler cette méthode après `ProcessEnding()` et avant de terminer le processus. Si vous utilisez une flotte Anywhere et que vous n'interrompez pas les processus du serveur après chaque session de jeu, appelez `Destroy()` puis `InitSDK()` réinitialisez avant de signaler Amazon GameLift Servers que le processus est prêt à héberger une session de jeu avec. `ProcessReady()`

Syntaxe

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Paramètres

Il n'y a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();  
Aws::GameLift::Server::Destroy();  
  
// Exit the process with success or failure  
if (processEndingOutcome.IsSuccess()) {  
    exit(0);  
}  
else {  
    cout << "ProcessEnding() failed. Error: " <<  
    processEndingOutcome.GetError().GetErrorMessage();  
    exit(-1);  
}
```

SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers

Utilisez la référence du SDK 5.x du serveur pour intégrer votre jeu multijoueur à des fins d'hébergement. Amazon GameLift Servers Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#). Si vous utilisez le Amazon GameLift Servers plugin pour Unity, consultez également [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#).

SDK 5.x du serveur C# pour Amazon GameLift Servers -- Types de données

Utilisez la référence du SDK 5.x du serveur Amazon GameLift Servers C# pour intégrer votre jeu multijoueur à des fins d'hébergement. Amazon GameLift Servers Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#). Si vous utilisez le Amazon GameLift Servers plugin pour Unity, consultez également [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#).

[SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)

Types de données

- [LogParameters](#)
- [ProcessParameters](#)

- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Joueur](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)
- [AttributeValue](#)
- [AwsStringOutcome](#)
- [GenericOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [AwsDateTimeOutcome](#)
- [GameLiftError](#)
- [Enums](#)

LogParameters

Utilisez ce type de données pour identifier les fichiers générés pendant une session de jeu vers lesquels vous souhaitez que le serveur de jeu les télécharge Amazon GameLift Servers après la fin de la session de jeu. Le serveur de jeu communique `LogParameters` to Amazon GameLift Servers lors d'un [ProcessReady\(\)](#) appel.

Propriétés	Description

LogPaths	La liste des chemins de répertoire vers les fichiers journaux du serveur de jeu que vous Amazon GameLift Servers souhaitez stocker pour un accès futur. Le processus du serveur génère ces fichiers lors de chaque session de jeu. Vous définissez les chemins et les noms des fichiers sur votre serveur de jeu et vous les stockez dans le répertoire racine du jeu. Les chemins du journal doivent être absolus. Par exemple, si votre build de jeu stocke les journaux de session de jeu dans un chemin tel que <code>MyGame\sessionLogs\</code>, le chemin se <code>c:\game\MyGame\sessionLogs</code> trouve sur une instance Windows. Type : <code>List<String></code> Obligatoire : non

ProcessParameters

Ce type de données contient l'ensemble des paramètres envoyés Amazon GameLift Servers lors d'un [ProcessReady\(\)](#) appel.

Propriétés	Description
LogParameters	L'objet contenant une liste de chemins de répertoire vers les fichiers journaux des sessions de jeu. Type : <code>Aws::GameLift::Server:: LogParameters</code> Obligatoire : oui
OnHealthCheck	Nom de la fonction de rappel Amazon GameLift Servers invoquée pour demander un rapport

d'état de santé au processus du serveur. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes. Après avoir appelé cette fonction, Amazon GameLift Servers elle attend 60 secondes pour recevoir une réponse. Si aucune réponse n'est reçue, le processus Amazon GameLift Servers du serveur est considéré comme défaillant.

Type : `void OnHealthCheckDelegate()`

Obligatoire : oui

OnProcessTerminate

Nom de la fonction de rappel Amazon GameLift Servers invoquée pour forcer l'arrêt du processus du serveur. Après l'appel de cette fonction, Amazon GameLift Servers patiente cinq minutes afin que le processus serveur s'arrête et réponde avec un appel [ProcessEnding\(\)](#) avant d'arrêter le processus serveur.

Type : `void OnProcessTerminate Delegate()`

Obligatoire : oui

OnStartGameSession

Nom de la fonction de rappel Amazon GameLift Servers invoquée pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à la demande du client [CreateGameSession](#). La fonction de rappel prend un [GameSession](#) objet.

Type : `void OnStartGameSession Delegate( GameSession )`

Obligatoire : oui

OnUpdateGameSession	Nom de la fonction de rappel Amazon GameLift Servers invoquée pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de remplissage de correspondance a été traitée pour fournir des données de matchmaking mises à jour. Il transmet un GameSession objet, une mise à jour de statut (updateReason) et l'identifiant du ticket de remplacement des matchs. Type : vide OnUpdateGameSessionDelegate (UpdateGameSession) Obligatoire : non
Port	Numéro de port sur lequel le processus serveur écoute les connexions des nouveaux joueurs. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur. Type : Integer Obligatoire : oui

UpdateGameSession

Les informations mises à jour pour un objet de session de jeu incluent la raison pour laquelle la session de jeu a été mise à jour. Si la mise à jour est liée à une action de remplissage par correspondance, ce type de données inclut l'ID du ticket de remplissage.

Propriétés	Description
GameSession	Un objet GameSession. L'GameSession objet contient des propriétés décrivant une session de jeu. Type : <code>GameSession</code> <code>GameSession()</code> Obligatoire : oui
UpdateReason	La raison pour laquelle la session de jeu est mise à jour. Type : <code>UpdateReason</code> <code>UpdateReason()</code> Obligatoire : oui
BackfillTicketId	L'identifiant du ticket de remplacement qui tente de mettre à jour la session de jeu. Type : <code>String</code> Obligatoire : oui

GameSession

Détails d'une session de jeu.

Propriétés	Description
GameSessionId	Identifiant unique pour la session de jeu. L'ARN d'une session de jeu a le format suivant : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Type : <code>String</code> Obligatoire : non

Propriétés	Description
Nom	Une étiquette descriptive de la session de jeu. Type : String Obligatoire : non
FleetId	Identifiant unique de la flotte sur laquelle s'exécute la session de jeu. Type : String Obligatoire : non
MaximumPlayerSessionCount	Le nombre maximum de connexions de joueurs à la session de jeu. Type : Integer Obligatoire : non
Port	Le numéro de port de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : Integer Obligatoire : non
IpAddress	Adresse IP de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : String Obligatoire : non

Propriétés	Description
GameSessionData	Ensemble de propriétés de session de jeu personnalisées, mises en forme en tant que valeur de chaîne unique. Type : <code>String</code> Obligatoire : non
MatchmakerData	Les informations sur le processus de matchmaking utilisé pour créer la session de jeu, en syntaxe JSON, formatées sous forme de chaîne. En plus de la configuration de matchmaking utilisée, elle contient des données sur tous les joueurs affectés au match, y compris les attributs des joueurs et les affectations des équipes. Type : <code>String</code> Obligatoire : non
GameProperties	Ensemble de propriétés personnalisées pour une session de jeu, formatées sous forme de paires clé:valeur. Ces propriétés sont transmises avec une demande de démarrage d'une nouvelle session de jeu. Type : <code>Dictionary<string, string></code> Obligatoire : non

Propriétés	Description
DnsName	Identifiant DNS attribué à l'instance qui exécute la session de jeu. Les valeurs ont le format suivant : • Flottes compatibles TLS : <unique identifieur>.<region identifieur>.amazongamelift.com • Non-TLS-enabled flottes :ec2-<unique identifieur>.compute.amazonaws.com . Lorsque vous vous connectez à une session de jeu exécutée sur une flotte compatible TLS, vous devez utiliser le nom DNS et non l'adresse IP. Type : String Obligatoire : non

ServerParameters

Informations utilisées pour maintenir la connexion entre un serveur Amazon GameLift Servers Anywhere et le Amazon GameLift Servers service. Ces informations sont utilisées lors du lancement de nouveaux processus serveur avec [InitSDK\(\)](#). Pour les serveurs hébergés sur EC2 des instances Amazon GameLift Servers gérées, utilisez un objet vide.

Propriétés	Description
WebSocketUrl	Ils <code>GameLiftServerSdkEndpoint</code> sont retournés lorsque vous <code>RegisterCompute</code> faisiez partie de Amazon GameLift Servers Anywhere. Type : String

Propriétés	Description
	Obligatoire : oui
ProcessId	Un identifiant unique enregistré auprès du processus serveur hébergeant votre jeu. Type : String Obligatoire : oui
HostId	Un identifiant unique pour l'hôte chargé des processus hébergeant votre jeu sur le serveur. Le HostID est celui ComputeName utilisé lors de l'enregistrement de votre ordinateur. Pour plus d'informations, voir RegisterCompute Type : String Obligatoire : oui
FleetId	ID de flotte du parc dans lequel le calcul est enregistré. Pour plus d'informations, voir, RegisterCompute. Type : String Obligatoire : oui
AuthToken	Le jeton d'authentification généré par Amazon GameLift Servers celui-ci authentifie votre serveur auprès de Amazon GameLift Servers. Pour plus d'informations, voir, GetComputeAuthToken. Type : String Obligatoire : oui

StartMatchBackfillRequest

Informations utilisées pour créer une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations Amazon GameLift Servers lors d'un [StartMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	L'identifiant unique de session de jeu. L'opération API GetGameSessionId renvoie l'identifiant au format ARN. Type : String Obligatoire : oui
MatchmakingConfigurationArn	L'identifiant unique, sous la forme d'un ARN, que le système de jumelage doit utiliser pour cette demande. L'ARN de matchmaking pour la session de jeu d'origine se trouve dans l'objet de session de jeu dans la propriété de données du matchmaker. Pour en savoir plus sur les données du système de jumelage, consultez la section Travailler avec les données du système de jumelage. Type : String Obligatoire : oui
Joueurs	Ensemble de données représentant tous les joueurs qui participent actuellement à la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Type : List<Player> Obligatoire : oui

Propriétés	Description
TicketId	L'identifiant unique d'un ticket de demande de matchmaking ou de remplacement de match. Si vous ne fournissez aucune valeur, il en Amazon GameLift Servers génère une. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire. Type : <code>String</code> Obligatoire : non

Joueur

Représente un joueur dans le matchmaking. Lorsqu'une demande de matchmaking démarre, un joueur dispose d'un identifiant de joueur, d'attributs et éventuellement de données de latence. Amazon GameLift Servers ajoute des informations sur l'équipe après la fin d'un match.

Propriétés	Description
LatencyInMS	Ensemble de valeurs exprimées en millisecondes, qui indiquent le niveau de latence ressenti par un joueur lorsqu'il est connecté à un lieu. Si cette propriété est utilisée, le joueur n'est jumelé qu'aux emplacements répertoriés. Si un matchmaker dispose d'une règle qui évalue la latence, les joueurs doivent indiquer la latence pour être mis en relation. Type : <code>Dictionary<string, int></code> Obligatoire : non

Propriétés	Description
PlayerAttributes	Une collection de paires clé:valeur contenant des informations sur les joueurs à utiliser dans le matchmaking. Les clés d'attribut du joueur doivent correspondre à PlayerAttributes celles utilisées dans un ensemble de règles de matchmaking. Pour plus d'informations sur les attributs des joueurs, consultez AttributeValue. Type : Dictionary<string, Attribute Value Obligatoire : non
PlayerId	Identifiant unique pour un joueur. Type : String Obligatoire : non
Equipe	Le nom de l'équipe à laquelle le joueur est affecté lors d'un match. Vous définissez le nom de l'équipe dans le jeu de règles de matchmaking. Type : String Obligatoire : non

DescribePlayerSessionsRequest

Ce type de données est utilisé pour spécifier les sessions de joueur à récupérer. Il peut être utilisé de plusieurs manières : (1) fournir une session PlayerSessionId pour demander une session de joueur spécifique ; (2) fournir une GameSessionId pour demander toutes les sessions de joueur dans la session de jeu spécifiée ; ou (3) fournir une PlayerId pour demander toutes les sessions de joueur

pour le joueur spécifié. Pour les volumes importants de sessions de joueur, utilisez les paramètres de pagination pour récupérer les résultats en tant que pages séquentielles.

Propriétés	Description
GameSessionId	L'identifiant unique de session de jeu. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'ID de session de jeu est le suivant : <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> . La valeur de <code><ID string></code> peut être une chaîne d'ID personnalisée (si spécifiée lors de la création de la session de jeu) ou une chaîne générée automatiquement. Type : String Obligatoire : non
PlayerSessionId	Identifiant unique d'une session de joueur. Type : String Obligatoire : non
PlayerId	L'identifiant unique d'un joueur. Consultez Générer un joueur IDs. Type : String Obligatoire : non
PlayerSessionStatusFilter	État de la session du joueur sur lequel filtrer les résultats. Les états de session de joueur possibles sont les suivants : • RÉSERVÉ — La demande de session du joueur a été reçue, mais le joueur ne

Propriétés	Description
	s'est pas encore connecté au serveur. Le processus and/or a été validé. • ACTIVE - Le joueur a été validé par le processus serveur et est actuellement connecté. • COMPLETED - La connexion du joueur a été abandonnée. • TIMEDOUT — Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté n' and/or a pas été validée dans le délai imparti (60 secondes). Type : String Obligatoire : non
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : String Obligatoire : non
Limite	Nombre maximal de résultats à renvoyer. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : int Obligatoire : non

StopMatchBackfillRequest

Informations utilisées pour annuler une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Identifiant de session de jeu unique de la demande annulée. Type : string Obligatoire : oui
MatchmakingConfigurationArn	L'identifiant unique du matchmaker auquel cette demande a été envoyée. Type : string Obligatoire : oui
TicketId	L'identifiant unique du ticket de demande de remblayage à annuler. Type : string Obligatoire : oui

GetFleetRoleCredentialsRequest

Ce type de données donne au serveur de jeu un accès limité à vos autres AWS ressources. Pour plus d'informations, voir, [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Propriétés	Description
RoleArn	Le nom de ressource Amazon (ARN) du rôle de service qui étend un accès limité à vos AWS ressources.

Propriétés	Description
	Type : <code>string</code> Obligatoire : oui
RoleSessionName	Le nom de la session qui décrit l'utilisation des informations d'identification du rôle. Type : <code>string</code> Obligatoire : non

AttributeValue

Utilisez ces valeurs dans les paires [Joueur](#) clé-valeur d'attribut. Cet objet vous permet de spécifier une valeur d'attribut à l'aide de n'importe quel type de données valide : chaîne, nombre, tableau de chaînes ou mappage de données. Chaque `AttributeValue` objet ne peut utiliser qu'une seule des propriétés disponibles.

Propriétés	Description
Type ATR	Spécifie le type de valeur d'attribut. Type : valeur <code>AttrType</code> enum . Obligatoire : non
S	Représente une valeur d'attribut de chaîne. Type : <code>string</code> Obligatoire : oui
N	Représente une valeur d'attribut numérique. Type : <code>double</code> Obligatoire : oui
SL	Représente un tableau de valeurs d'attributs de chaîne.

Propriétés	Description
	Type : <code>string[]</code> Obligatoire : oui
SDM	Représente un dictionnaire de clés de chaîne et de valeurs doubles. Type : <code>Dictionary<string, double></code> Obligatoire : oui

AwsStringOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : <code>string</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GenericOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

DescribePlayerSessionsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called "DescribePlayerSessionsResult" Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

DescribePlayerSessionsResult

Propriétés	Description
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : <code>string</code> Obligatoire : oui
PlayerSessions	Collection d'objets contenant les propriétés de chaque session de joueur correspondant à la demande. Type : <code>IList<the section called "PlayerSession"></code> Obligatoire :
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

PlayerSession

Propriétés	Description
CreationTime	Type : <code>long</code> Obligatoire : oui

Propriétés	Description
FleetId	Type : string Obligatoire : oui
GameSessionId	Type : string Obligatoire : oui
IpAddress	Type : string Obligatoire : oui
PlayerData	Type : string Obligatoire : oui
PlayerId	Type : string Obligatoire : oui
PlayerSessionId	Type : string Obligatoire : oui
Port	Type : int Obligatoire : oui
Statut	Type : Une PlayerSessionStatus énumération. Obligatoire : oui
TerminationTime	Type : long Obligatoire : oui
DnsName	Type : string Obligatoire : oui

StartMatchBackfillOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “StartMatchBackfillResult” Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “GameLiftError” Obligatoire : non

StartMatchBackfillResult

Propriétés	Description
TicketId	Type : string Obligatoire : oui

GetComputeCertificateOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action.

Propriétés	Description
	Type : the section called “GetComputeCertificateResult” Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “GameLiftError” Obligatoire : non

GetComputeCertificateResult

Le chemin d'accès au certificat TLS sur votre ordinateur et le nom d'hôte du calcul.

Propriétés	Description
CertificatePath	Type : string Obligatoire : oui
ComputeName	Type : string Obligatoire : oui

GetFleetRoleCredentialsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action.

Propriétés	Description
	Type : the section called “GetFleetRoleCredentialsResult” Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “GameLiftError” Obligatoire : non

GetFleetRoleCredentialsResult

Propriétés	Description
AccessKeyId	ID de clé d'accès permettant d'authentifier et de fournir un accès à vos AWS ressources. Type : string Obligatoire : non
AssumedRoleId	ID de l'utilisateur auquel appartient le rôle de service. Type : string Obligatoire : non
AssumedRoleUserArn	Le nom de ressource Amazon (ARN) de l'utilisateur auquel appartient le rôle de service. Type : string Obligatoire : non

Propriétés	Description
Expiration	Durée avant l'expiration de vos informations d'identification de session. Type : <code>DateTime</code> Obligatoire : non
SecretAccessKey	ID de clé d'accès secrète pour l'authentification. Type : <code>string</code> Obligatoire : non
SessionToken	Un jeton pour identifier la session active en cours qui interagit avec vos AWS ressources. Type : <code>string</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

AwsDateTimeOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action.

Propriétés	Description
	Type : <code>DateTime</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

GameLiftError

Propriétés	Description
ErrorType	Type d'erreur. Type : <code>One GameLiftErrorType</code> énumération. Obligatoire : non
ErrorMessage	Nom de l'erreur. Type : <code>string</code> Obligatoire : non
ErrorMessage	Message d'erreur. Type : <code>string</code> Obligatoire : non

Enums

Les énumérations définies pour le SDK du serveur pour Amazon GameLift Servers (C#) sont définies comme suit :

AttrType

- NONE
- CORDE
- DOUBLE
- LISTE_CHAÎNES
- STRING_DOUBLE_MAP

GameLiftErrorType

Valeur de chaîne indiquant le type d'erreur. Les valeurs valides sont les suivantes :

- SERVICE_CALL_FAILED — Un appel à un AWS service a échoué.
- LOCAL_CONNECTION_FAILED — La connexion locale a échoué. Amazon GameLift Servers
- NETWORK_NOT_INITIALIZED — Le réseau n'a pas été initialisé.
- GAMESESSION_ID_NOT_SET — L'ID de session de jeu n'a pas été défini.
- MAUVAISE DEMANDE D'EXCEPTION
- EXCEPTION DE SERVICE INTERNE
- ALREADY_INITIALIZED — Le Amazon GameLift Servers serveur ou le client a déjà été initialisé avec Initialize ().
- FLEET_MISMATCH — La flotte cible ne correspond pas à la flotte d'une GameSession ou d'une PlayerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED — Le client n'a pas été initialisé. Amazon GameLift Servers
- GAMELIFT_SERVER_NOT_INITIALIZED — Le serveur n'a pas été initialisé. Amazon GameLift Servers
- GAME_SESSION_ENDED_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin de la session de jeu.
- GAME_SESSION_NOT_READY — La Amazon GameLift Servers session de jeu sur le serveur n'a pas été activée.
- GAME_SESSION_READY_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que la session de jeu est prête.

- **INITIALIZATION_MISMATCH** — Une méthode client a été appelée après `Server : :Initialize ()`, ou vice versa.
- **NOT_INITIALIZED** — Le Amazon GameLift Servers serveur ou le client n'a pas été initialisé avec `Initialize ()`.
- **NO_TARGET_ALIASID_SET** — Aucun `AliasID` cible n'a été défini.
- **NO_TARGET_FLEET_SET** — Aucune flotte cible n'a été définie.
- **PROCESS_ENDING_FAILED** : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin du processus.
- **PROCESS_NOT_ACTIVE** — Le processus du serveur n'est pas encore actif, n'est pas lié à un et ne peut ni `GameSession` accepter ni traiter. `PlayerSessions`
- **PROCESS_NOT_READY** — Le processus du serveur n'est pas encore prêt à être activé.
- **PROCESS_READY_FAILED** : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que le processus est prêt.
- **SDK_VERSION_DETECTION_FAILED** — La détection de la version du SDK a échoué.
- **STX_CALL_FAILED** — Un appel au composant principal du XStx serveur a échoué.
- **STX_INITIALIZATION_FAILED** — Le composant principal du XStx serveur n'a pas pu être initialisé.
- **UNEXPECTED_PLAYER_SESSION** — Le serveur a détecté une session de joueur non enregistrée.
- **ÉCHEC DE CONNEXION WEBSOCKET**
- **WEBSOCKET_CONNECT_FAILURE_FORBIDDEN**
- **WEBSOCKET_CONNECT_FAILURE_INVALID_URL**
- **WEBSOCKET_CONNECT_FAILURE_TIMEOUT**
- **WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE** — Impossible de récupérer un message au service. `GameLift WebSocket`
- **WEBSOCKET_SEND_MESSAGE_FAILURE** — Impossible d'envoyer un message au service. `GameLift WebSocket`
- **MATCH_BACKFILL_REQUEST_VALIDATION** — La validation de la demande a échoué.
- **PLAYER_SESSION_REQUEST_VALIDATION** — La validation de la demande a échoué.

PlayerSessionCreationPolicy

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs. Les valeurs valides sont les suivantes :

- ACCEPT_ALL — Accepte toutes les sessions de nouveaux joueurs.
- DENY_ALL — Refuse toutes les sessions de nouveaux joueurs.
- NOT_SET — La session de jeu n'est pas configurée pour accepter ou refuser les sessions de nouveaux joueurs.

PlayerSessionStatus

- ACTIF
- TERMINÉ
- NOT_SET
- RÉSERVÉ
- DÉLAI EXPIRÉ

SDK 5.x du serveur C# pour Amazon GameLift Servers -- Types de données

Rubriques

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Détruire \(\)](#)

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
AwsStringOutcome GetSdkVersion();
```

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet [the section called “AwsStringOutcome”](#). La chaîne renvoyée inclut le numéro de version (exemple `5.0.0`). En cas d'échec, renvoie un message d'erreur.

exemple

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitSDK()

Initialise le Amazon GameLift Servers SDK pour un parc géré EC2 . Appelez cette méthode au lancement, avant que toute autre initialisation associée ne se Amazon GameLift Servers produise. Cette méthode lit les paramètres du serveur depuis l'environnement hôte afin de configurer la communication entre le serveur et le Amazon GameLift Servers service. Il utilise un jeton d'idempotence, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Syntaxe

```
GenericOutcome InitSDK();
```

Valeur renvoyée

En cas de succès, renvoie un `InitSdkOutcome` objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

exemple

```
//Call InitSDK to establish a local connection with the GameLift agent to enable further communication.
```



```
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

Initialise le Amazon GameLift Servers SDK pour une flotte Anywhere. Appelez cette méthode au lancement, avant que toute autre initialisation associée ne se Amazon GameLift Servers produise. Cette méthode nécessite des paramètres de serveur explicites pour configurer la communication entre le serveur et le Amazon GameLift Servers service. Il utilise un jeton d'idempotence, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Syntaxe

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

Paramètres

[ServerParameters](#)

Pour initialiser un serveur de jeu sur une flotte Amazon GameLift Servers Anywhere, créez un `ServerParameters` objet avec les informations suivantes :

- URL WebSocket utilisée pour vous connecter à votre serveur de jeu.
- ID du processus utilisé pour héberger votre serveur de jeu.
- L'ID de l'ordinateur hébergeant les processus de votre serveur de jeu.
- L'ID du Amazon GameLift Servers parc contenant votre ordinateur Amazon GameLift Servers Anywhere.
- Le jeton d'autorisation généré par l'Amazon GameLift Servers opération.

Valeur renvoyée

En cas de succès, renvoie un `InitSdkOutcome` objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

Note

Si les appels à échouent pour `InitSDK()` les builds de jeu déployés sur des flottes Anywhere, vérifiez le `ServerSdkVersion` paramètre utilisé lors de la création de la ressource de build. Vous devez définir explicitement cette valeur en fonction de la version

du SDK du serveur utilisée. La valeur par défaut de ce paramètre est 4.x, ce qui n'est pas compatible. Pour résoudre ce problème, créez une nouvelle version et déployez-la sur une nouvelle flotte.

exemple

```
//Define the server parameters
string websocketUrl = "wss://us-west-1.api.amazongamelift.com";
string processId = "PID1234";
string fleetId = "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
string hostId = "HardwareAnywhere";
string authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
ServerParameters serverParameters =
    new ServerParameters(websocketUrl, processId, hostId, fleetId, authToken);

//Call InitSDK to establish a local connection with the GameLift agent to enable
    further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift ServersIndique que le processus du serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après l'avoir invoquée. [InitSDK\(\)](#) Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Paramètres

[ProcessParameters](#)

Un `ProcessParameters` objet contient des informations sur le processus du serveur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre à la fois les implémentations de méthodes et de fonctions déléguées.

```
// Set parameters and call ProcessReady
ProcessParameters processParams = new ProcessParameters(
    this.OnStartGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnUpdateGameSession,
    port,
    new LogParameters(new List<string>()
    // Examples of log and error files written by the game server
    {
        "C:\\game\\logs",
        "C:\\game\\error"
    })
);
GenericOutcome processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

ProcessEnding()

Amazon GameLift Servers indique que le processus du serveur est en train de se terminer. Appelez cette méthode après toutes les autres tâches de nettoyage (y compris la fermeture de la session de jeu active) et avant de terminer le processus. En fonction du résultat de `ProcessEnding()`, le processus se termine avec succès (0) ou erreur (-1) et génère un événement de flotte. Si le processus se termine par une erreur, l'événement de flotte généré est `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntaxe

```
GenericOutcome ProcessEnding()
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple appelle `ProcessEnding()` et `Destroy()` avant de terminer le processus du serveur avec un code de sortie de réussite ou d'erreur.

```
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();
GameLiftServerAPI.Destroy();

if (processEndingOutcome.Success)
{
    Environment.Exit(0);
}
else
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
```

ActivateGameSession()

Amazon GameLift ServersIndique que le processus du serveur a activé une session de jeu et qu'il est maintenant prêt à recevoir les connexions des joueurs. Cette action doit être appelée dans le cadre de la fonction de `onStartGameSession()` rappel, après toute initialisation de session de jeu.

Syntaxe

```
GenericOutcome ActivateGameSession()
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre l'appel de `ActivateGameSession()` dans le cadre de la fonction déléguée `onStartGameSession()`.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    // When ready to receive players
    GenericOutcome activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur.

Syntaxe

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
playerSessionPolicy)
```

Paramètres

playerSessionPolicy

Valeur de chaîne indiquant si la session de jeu accepte de nouveaux joueurs.

Les valeurs valides sont les suivantes :

- ACCEPT_ALL — Accepte toutes les sessions de nouveaux joueurs.
- DENY_ALL — Refuse toutes les sessions de nouveaux joueurs.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
GenericOutcome updatePlayerSessionPolicyOutcome =
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

Récupère l'ID de la session de jeu hébergée par le processus serveur actif.

Pour les processus inactifs qui ne sont pas activés lors d'une session de jeu, l'appel renvoie un [the section called "GameLiftError"](#).

Syntaxe

```
AwsStringOutcome GetGameSessionId()
```

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet [the section called "AwsStringOutcome"](#). En cas d'échec, renvoie un message d'erreur. »

exemple

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

Renvoie l'heure d'arrêt planifiée pour un processus serveur, si une heure de résiliation est disponible. Un processus serveur exécute cette action après avoir reçu un `onProcessTerminate()` rappel de Amazon GameLift Servers. Amazon GameLift Servers appelle `onProcessTerminate()` pour les raisons suivantes :

- Lorsque le processus du serveur a signalé un mauvais état de santé ou n'a pas répondu Amazon GameLift Servers.
- Lorsque vous mettez fin à l'instance lors d'un événement de réduction de la taille.
- Lorsqu'une instance est interrompue en raison d'une interruption [ponctuelle](#).

Syntaxe

```
AwsDateTimeOutcome GetTerminationTime()
```

Valeur renvoyée

En cas de succès, renvoie l'heure de fin sous forme d'[the section called "AwsDateTimeOutcome"](#) objet. La valeur est le délai de résiliation, exprimé en ticks écoulés depuis. 0001 00:00:00 Par exemple, la valeur de la date et de l'heure 2020-09-13 12:26:40 -000Z est égale à celle 637355968000000000 des ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

exemple

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers indique qu'un joueur avec l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide. Une fois la session du joueur validée, le statut de l'emplacement du joueur Amazon GameLift Servers passe de RÉSERVÉ à ACTIF.

Syntaxe

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Paramètres

playerSessionId

Identifiant unique émis GameLift lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre une fonction permettant de gérer une demande de connexion, notamment de valider et de rejeter une session de joueur non valide. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId)
{
    GenericOutcome acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
    else
    {

```

```
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
    }
}
```

RemovePlayerSession()

Indique Amazon GameLift Servers qu'un joueur s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers change l'emplacement du joueur pour le rendre disponible.

Syntaxe

```
GenericOutcome RemovePlayerSession(String playerId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
GenericOutcome removePlayerSessionOutcome =
    GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Récupère les données de session du joueur, notamment les paramètres, les métadonnées de session et les données du joueur. Utilisez cette action pour obtenir des informations pour une seule session de joueur, pour toutes les sessions de joueur d'une session de jeu ou pour toutes les sessions de joueur associées à un seul ID de joueur.

Syntaxe

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
    describePlayerSessionsRequest)
```


Paramètres

[DescribePlayerSessionsRequest](#)

[the section called “DescribePlayerSessionsRequest”](#) Objet qui décrit les sessions de joueur à récupérer.

Valeur renvoyée

En cas de succès, renvoie un [the section called “DescribePlayerSessionsOutcome”](#) objet contenant un ensemble d'objets de session de joueur correspondant aux paramètres de la demande.

exemple

Cet exemple illustre une demande de toutes les sessions de joueur activement connectées à une session de jeu spécifiée. En omettant NextToken et en définissant la valeur limite sur 10, les 10 premiers enregistrements de session de joueurs correspondant à la demande Amazon GameLift Servers seront renvoyés.

```
// Set request parameters
DescribePlayerSessionsRequest describePlayerSessionsRequest = new
    DescribePlayerSessionsRequest()
{
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for the
current game session
    Limit = 10,
    PlayerSessionStatusFilter =
        PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
DescribePlayerSessionsOutcome describePlayerSessionsOutcome =
    GameLiftServerAPI.DescribePlayerSessions(describePlayerSessionsRequest);
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés, Amazon GameLift Servers fournit des données de matchmaking mises à jour à l'aide de la fonction de rappel. `OnUpdateGameSession()`

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
startBackfillRequest);
```

Paramètres

[StartMatchBackfillRequest](#)

Un `StartMatchBackfillRequest` objet contient des informations relatives à la demande de remblayage.

Valeur renvoyée

Renvoie un [the section called "StartMatchBackfillOutcome"](#) objet avec l'identifiant du ticket de remplacement correspondant, ou un échec avec un message d'erreur.

exemple

```
// Build a backfill request  
StartMatchBackfillRequest startBackfillRequest = new StartMatchBackfillRequest()  
{  
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional  
    MatchmakingConfigurationArn = "arn:aws:gamelift:us-  
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",  
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    // gets ID for  
    current game session  
    MatchmakerData matchmakerData =  
        MatchmakerData.FromJson(gameSession.MatchmakerData), // gets matchmaker data for  
    current players  
    // get matchmakerData.Players  
    // remove data for players who are no longer connected  
    Players = ListOfPlayersRemainingInTheGame
```

```
};

// Send backfill request
StartMatchBackfillOutcome startBackfillOutcome =
    GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update matchmaker
    data as needed
}
```

StopMatchBackfill()

Annule une demande de remplacement de match active. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Syntaxe

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Paramètres

[StopMatchBackfillRequest](#)

Un `StopMatchBackfillRequest` objet qui fournit des détails sur le ticket de matchmaking que vous êtes en train d'arrêter.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set backfill stop request parameters
StopMatchBackfillRequest stopBackfillRequest = new StopMatchBackfillRequest(){
    TicketId = "1111aaaa-22bb-33cc-44dd-5555eeee66ff", //optional, if not provided one is
    autogenerated
    MatchmakingConfigurationArn = "arn:aws:gamelift:us-
    west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
```

```
GameSessionId = GameLiftServerAPI.GetGameSessionId().Result    //gets the ID for the
current game session
};
GenericOutcome stopBackfillOutcome =
    GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

GetComputeCertificate()

Récupère le chemin d'accès au certificat TLS utilisé pour chiffrer la connexion réseau entre le serveur de jeu et votre client de jeu. Vous pouvez utiliser le chemin du certificat lorsque vous enregistrez votre appareil informatique dans un parc Amazon GameLift Servers Anywhere. Pour plus d'informations, voir, [RegisterCompute](#).

Syntaxe

```
GetComputeCertificateOutcome GetComputeCertificate();
```

Valeur renvoyée

Renvoie un `GetComputeCertificateResponse` objet contenant les éléments suivants :

- `CertificatePath`: chemin d'accès au certificat TLS sur votre ressource de calcul. Lorsque vous utilisez une flotte Amazon GameLift Servers gérée, ce chemin contient :
 - `certificate.pem`: le certificat d'utilisateur final. La chaîne de certificats complète est la combinaison des `certificateChain.pem` éléments ajoutés à ce certificat.
 - `certificateChain.pem`: chaîne de certificats qui contient le certificat racine et les certificats intermédiaires.
 - `rootCertificate.pem`: le certificat racine.
 - `privateKey.pem`: clé privée pour le certificat d'utilisateur final.
- `ComputeName`: nom de votre ressource de calcul.

exemple

```
GetComputeCertificateOutcome getComputeCertificateOutcome =
    GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

Récupère les informations d'identification du rôle IAM qui autorisent l'interaction avec Amazon GameLift Servers d'autres utilisateurs. Services AWS Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

Syntaxe

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest request);
```

Paramètres

[GetFleetRoleCredentialsRequest](#)

Des informations d'identification de rôle qui étendent un accès limité à vos AWS ressources au serveur de jeu.

Valeur renvoyée

Renvoie un objet [the section called “GetFleetRoleCredentialsOutcome”](#).

exemple

```
// form the fleet credentials request
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new
    GetFleetRoleCredentialsRequest(){
    RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"
};
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =
    GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Détruire ()

Libère de la mémoire le SDK du serveur de Amazon GameLift Servers jeu. Il est recommandé d'appeler cette méthode après `ProcessEnding()` et avant de terminer le processus. Si vous utilisez une flotte Anywhere et que vous n'interrompez pas les processus du serveur après chaque session de jeu, appelez `Destroy()` puis `InitSDK()` réinitialisez avant de signaler Amazon GameLift Servers que le processus est prêt à héberger une session de jeu avec. `ProcessReady()`

Syntaxe

```
GenericOutcome Destroy()
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Operations to end game sessions and the server process
GenericOutcome processEndingOutcome = GameLiftServerAPI.ProcessEnding();

// Shut down and destroy the instance of the GameLift Game Server SDK
GenericOutcome destroyOutcome = GameLiftServerAPI.Destroy();

// Exit the process with success or failure
if (processEndingOutcome.Success)
{
    Environment.Exit(0);
}
else
{
    Console.WriteLine("ProcessEnding() failed. Error: " +
        processEndingOutcome.Error.ToString());
    Environment.Exit(-1);
}
```

SDK du serveur Go pour Amazon GameLift Servers -- Actions

Utilisez la référence du SDK 5.x du serveur pour intégrer votre jeu multijoueur à des fins d'hébergement. Amazon GameLift Servers Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

`GameLiftServerAPI` définit les actions du SDK du serveur Go.

SDK du serveur Go pour Amazon GameLift Servers -- Types de données

Utilisez la référence du SDK du serveur pour intégrer votre jeu multijoueur à des fins d'Amazon GameLift Servers hébergement. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

GameLiftServerAPI.go définit les actions du SDK du serveur Go.

[SDK du serveur Go pour Amazon GameLift Servers -- Actions](#)

Types de données

- [LogParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Joueur](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)

LogParameters

Objet identifiant les fichiers générés au cours d'une session de jeu que vous Amazon GameLift Servers souhaitez télécharger et stocker à la fin de la session de jeu. Le serveur de jeu fournit LogParameters to dans Amazon GameLift Servers le cadre d'un ProcessParameters objet lors d'un [ProcessReady\(\)](#) appel.

Propriétés	Description
LogPaths	Liste des chemins de répertoire vers les fichiers journaux du serveur de jeu que vous Amazon GameLift Servers souhaitez stocker pour un accès futur. Le processus du serveur génère ces fichiers lors de chaque session de jeu. Vous définissez les chemins et les noms des fichiers sur votre serveur de jeu et vous les stockez dans le répertoire racine du jeu. Les chemins du journal doivent être absolus. Par exemple, si votre build de jeu stocke les journaux de session de jeu dans un chemin tel que MyGame\sessionLogs\ , alors le chemin se c : \game\MyGame\sessionLogs trouve sur une instance Windows.

	Type : []string
	Obligatoire : non

ProcessParameters

Objet décrivant la communication entre un processus serveur et Amazon GameLift Servers.

Le processus du serveur fournit ces informations à l'Amazon GameLift Serverside d'un appel à [ProcessReady\(\)](#).

Propriétés	Description
LogParameters	Objet avec des chemins de répertoire vers des fichiers générés au cours d'une session de jeu. Amazon GameLift Servers copie et stocke les fichiers pour un accès futur. Type : LogParameters Obligatoire : non
OnHealthCheck	La fonction de rappel qui Amazon GameLift Servers appelle le processus du serveur pour demander un rapport d'état de santé. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes et attend une réponse pendant 60 secondes. Le processus du serveur revient TRUE s'il est sain, FALSE sinon sain. Si aucune réponse n'est renvoyée, Amazon GameLift Servers enregistre le processus du serveur comme non sain. Type : OnHealthCheck func() bool Obligatoire : non
OnProcessTerminate	La fonction de rappel qui Amazon GameLift Servers invoque pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attendez 5 minutes que le processus serveur s'arrête et répondez par un ProcessEnding() appel avant d'arrêter le processus serveur. Type : OnProcessTerminate func() Obligatoire : oui

OnStartGameSession

La fonction de rappel qui Amazon GameLift Servers appelle pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de remplissage de correspondances a été traitée pour fournir des données de matchmaking mises à jour. Il transmet un [GameSession](#) objet, une mise à jour de statut (updateReason) et l'identifiant du ticket de remplacement des matchs.

Type : OnStartGameSession func (model.GameSession)

Obligatoire : oui

OnUpdateGameSession

La fonction de rappel qui demande de Amazon GameLift Servers transmette les informations de session de jeu mises à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction après avoir traité une demande de remplissage de correspondance pour fournir des données de matchmaking mises à jour.

Type : OnUpdateGameSession func (model.UpdateGameSession)

Obligatoire : non

Port

Numéro de port sur lequel le processus serveur écoute les connexions des nouveaux joueurs. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur.

Type : int

Obligatoire : oui

UpdateGameSession

Les mises à jour apportées à un objet de session de jeu, y compris la raison pour laquelle la session de jeu a été mise à jour et l'identifiant du ticket de remplissage associé si le remplissage est utilisé pour remplir les sessions des joueurs pendant la session de jeu.

Propriétés	Description
GameSession	Un objet GameSession. L'GameSession objet contient des propriétés décrivant une session de jeu. Type : <code>GameSession</code> <code>GameSession()</code> Obligatoire : oui
UpdateReason	La raison pour laquelle la session de jeu est mise à jour. Type : <code>UpdateReason</code> <code>UpdateReason()</code> Obligatoire : oui
BackfillTicketId	L'identifiant du ticket de remplacement qui tente de mettre à jour la session de jeu. Type : <code>String</code> Obligatoire : non

GameSession

Les détails d'une session de jeu.

Propriétés	Description
GameSessionId	Identifiant unique pour la session de jeu. Une session de jeu Amazon Resource Name (ARN) a le format suivant : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Type : <code>String</code> Obligatoire : non
Nom	Une étiquette descriptive de la session de jeu. Type : <code>String</code>

Propriétés	Description
	Obligatoire : non
FleetId	Identifiant unique de la flotte sur laquelle s'exécute la session de jeu. Type : <code>String</code> Obligatoire : non
MaximumPlayerSessionCount	Le nombre maximum de connexions de joueurs à la session de jeu. Type : <code>Integer</code> Obligatoire : non
Port	Le numéro de port de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : <code>Integer</code> Obligatoire : non
IpAddress	Adresse IP de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : <code>String</code> Obligatoire : non
GameSessionData	Ensemble de propriétés de session de jeu personnalisées, mises en forme en tant que valeur de chaîne unique. Type : <code>String</code> Obligatoire : non

Propriétés	Description
MatchmakerData	Les informations sur le processus de matchmaking utilisé pour créer la session de jeu, en syntaxe JSON, formatées sous forme de chaîne. En plus de la configuration de matchmaking utilisée, il contient des données sur tous les joueurs affectés au match, y compris les attributs des joueurs et les affectations des équipes. Type : String Obligatoire : non
GameProperties	Ensemble de propriétés personnalisées pour une session de jeu, formatées sous forme de paires clé:valeur. Ces propriétés sont transmises avec une demande de démarrage d'une nouvelle session de jeu. Type : <code>map[string] string</code> Obligatoire : non
DnsName	Identifiant DNS attribué à l'instance qui exécute la session de jeu. Les valeurs ont le format suivant : - Flottes compatibles TLS : <code><unique identifieur>.<region identifieur>.amazongamelift.com</code> - Non-TLS-enabled flottes : <code>ec2-<unique identifieur>.compute.amazonaws.com</code> Lorsque vous vous connectez à une session de jeu exécutée sur une flotte compatible TLS, vous devez utiliser le nom DNS et non l'adresse IP. Type : String Obligatoire : non

ServerParameters

Informations utilisées pour maintenir la connexion entre un serveur Amazon GameLift Servers Anywhere et le Amazon GameLift Servers service. Ces informations sont utilisées lors du lancement

de nouveaux processus serveur avec [InitSDK\(\)](#). Pour les serveurs hébergés sur EC2 des instances Amazon GameLift Servers gérées, utilisez un objet vide.

Propriétés	Description
WebSocket URL	Les <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers retours lorsque vous RegisterCompute recherchez une ressource de calcul Amazon GameLift Servers Anywhere. Type : string Obligatoire : oui
ProcessID	Un identifiant unique enregistré auprès du processus serveur hébergeant votre jeu. Type : string Obligatoire : oui
HostID	Identifiant unique de la ressource de calcul hébergeant le nouveau processus du serveur. Le HostID est celui <code>ComputeName</code> utilisé lorsque vous avez enregistré votre ordinateur. Pour de plus amples informations, veuillez consulter RegisterCompute. Type : string Obligatoire : oui
FleetID	Identifiant unique de la flotte dans laquelle le calcul est enregistré. Pour de plus amples informations, veuillez consulter RegisterCompute. Type : string Obligatoire : oui
AuthToken	Le jeton d'authentification généré par Amazon GameLift Servers celui-ci authentifie votre serveur auprès de Amazon GameLift Servers. Pour de plus amples informations, veuillez consulter GetComputeAuthToken.

Propriétés	Description
	Type : <code>string</code>
	Obligatoire : oui

StartMatchBackfillRequest

Informations utilisées pour créer une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations Amazon GameLift Servers lors d'un [StartMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	L'identifiant unique de session de jeu. L'opération API GetGameSessionId renvoie l'identifiant au format ARN. Type : <code>String</code> Obligatoire : oui
MatchmakingConfigurationArn	L'identifiant unique (sous la forme d'un ARN) que le système de jumelage doit utiliser pour cette demande. L'ARN du système de matchmaking correspondant à la session de jeu d'origine se trouve dans l'objet de session de jeu dans la propriété des données du système de matchmaking. Pour plus d'informations sur les données du système de jumelage, consultez la section Utilisation des données du système de jumelage . Type : <code>String</code> Obligatoire : oui
Joueurs	Ensemble de données représentant tous les joueurs qui participent actuellement à la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Type : <code>[]model.Player</code> Obligatoire : oui

Propriétés	Description
TicketId	L'identifiant unique d'un ticket de demande de matchmaking ou de remplacement de match. Si vous ne fournissez aucune valeur, il en Amazon GameLift Servers génère une. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire. Type : String Obligatoire : non

Joueur

L'objet qui représente un joueur dans le matchmaking. Lorsqu'une demande de matchmaking démarre, un joueur dispose d'un identifiant de joueur, d'attributs et éventuellement de données de latence. Amazon GameLift Servers ajoute des informations sur l'équipe après la fin d'un match.

Propriétés	Description
LatencyInMS	Ensemble de valeurs exprimées en millisecondes qui indiquent le niveau de latence ressenti par un joueur lorsqu'il est connecté à un lieu. Si cette propriété est utilisée, le joueur n'est jumelé qu'aux emplacements répertoriés. Si un matchmaker dispose d'une règle qui évalue la latence, les joueurs doivent indiquer la latence pour être mis en relation. Type : map[string] int Obligatoire : non
PlayerAttributes	Une collection de paires clé:valeur contenant des informations sur les joueurs à utiliser dans le matchmaking. Les clés d'attribut du joueur doivent correspondre à PlayerAttributes celles utilisées dans un ensemble de règles de matchmaking. Pour plus d'informations sur les attributs des joueurs, consultez Attribute Value. Type : map[string] AttributeValue

Propriétés	Description
	Obligatoire : non
PlayerId	Identifiant unique pour un joueur. Type : String Obligatoire : non
Equipe	Le nom de l'équipe à laquelle le joueur est affecté lors d'un match. Vous définissez le nom de l'équipe dans le jeu de règles de matchmaking. Type : String Obligatoire : non

DescribePlayerSessionsRequest

Objet qui indique les sessions de joueur à récupérer. Le processus du serveur fournit ces informations en [DescribePlayerSessions\(\)](#) appelant Amazon GameLift Servers.

Propriétés	Description
GameSessionID	Un identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'identifiant de session de jeu est <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> . GameSessionID Il s'agit d'une chaîne d'identification personnalisée ou d'une chaîne générée. Type : String Obligatoire : non
PlayerSessionID	Identifiant unique d'une session de joueur. Utilisez ce paramètre pour demander une session de joueur spécifique. Type : String

Propriétés	Description
	Obligatoire : non
PlayerID	L'identifiant unique d'un joueur. Utilisez ce paramètre pour demander toutes les sessions de joueur pour un joueur spécifique. Consultez Générer un joueur IDs. Type : String Obligatoire : non
PlayerSessionStatusFilter	État de la session du joueur sur lequel filtrer les résultats. Les statuts de session de joueur possibles incluent : • RÉSERVÉ — La demande de session du joueur a été reçue, mais le joueur ne s'est pas connecté au processus du serveur ni n'a été validé. • ACTIF — Le joueur a été validé par le processus du serveur et est connecté. • TERMINÉ — La connexion du joueur a été interrompue. • TIMEDOUT — Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté ou n'a pas été validée dans le délai imparti (60 secondes). Type : String Obligatoire : non
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : String Obligatoire : non

Propriétés	Description
Limit	Nombre maximal de résultats à renvoyer. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : <code>int</code> Obligatoire : non

StopMatchBackfillRequest

Informations utilisées pour annuler une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Identifiant de session de jeu unique de la demande annulée. Type : <code>string</code> Obligatoire : non
MatchmakingConfigurationArn	L'identifiant unique du matchmaker auquel cette demande a été envoyée. Type : <code>string</code> Obligatoire : non
TicketId	L'identifiant unique du ticket de demande de remblayage à annuler. Type : <code>string</code> Obligatoire : non

GetFleetRoleCredentialsRequest

Les informations d'identification du rôle qui étendent l'accès limité à vos AWS ressources au serveur de jeu. Pour plus d'informations, voir, [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Propriétés	Description
RoleArn	L'ARN du rôle de service qui étend un accès limité à vos AWS ressources. Type : <code>string</code> Obligatoire : oui
RoleSessionName	Le nom de la session qui décrit l'utilisation des informations d'identification du rôle. Type : <code>string</code> Obligatoire : oui

[SDK du serveur Go pour Amazon GameLift Servers -- Types de données](#)

Actions

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)

- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Détruire \(\)](#)

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
func GetSdkVersion() (string, error)
```

Valeur renvoyée

En cas de succès, renvoie la version actuelle du SDK sous forme de chaîne. La chaîne renvoyée inclut le numéro de version (exemple `5.0.0`). En cas d'échec, renvoie un message d'erreur tel que `common.SdkVersionDetectionFailed`.

exemple

```
version, err := server.GetSdkVersion()
```

InitSDK()

Initialise le kit SDK Amazon GameLift Servers. Appelez cette méthode au lancement avant que toute autre initialisation associée ne se produise. Cette méthode permet de configurer la communication entre le serveur et le Amazon GameLift Servers service.

Syntaxe

```
func InitSDK(params ServerParameters) error
```

Paramètres

[ServerParameters](#)

Pour initialiser un serveur de jeu sur une flotte Amazon GameLift Servers Anywhere, créez un `ServerParameters` objet avec les informations suivantes :

- URL WebSocket utilisée pour vous connecter à votre serveur de jeu.
- ID du processus utilisé pour héberger votre serveur de jeu.
- ID de l'ordinateur hébergeant les processus de votre serveur de jeu.
- L'ID du Amazon GameLift Servers parc contenant votre ordinateur Amazon GameLift Servers Anywhere.
- Le jeton d'autorisation généré par l'Amazon GameLift Servers opération.

Pour initialiser un serveur de jeu sur une EC2 flotte Amazon GameLift Servers gérée, créez un `ServerParameters` objet sans paramètres. Avec cet appel, l'Amazon GameLift Servers agent configure l'environnement informatique et se connecte automatiquement au Amazon GameLift Servers service pour vous.

Valeur renvoyée

En cas de succès, renvoie `nil` une erreur indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

Note

Si les appels à échouent pour `InitSDK()` les builds de jeu déployés sur des flottes Anywhere, vérifiez le `ServerSdkVersion` paramètre utilisé lors de la création de la ressource de build. Vous devez définir explicitement cette valeur en fonction de la version du SDK du serveur utilisée. La valeur par défaut de ce paramètre est 4.x, ce qui n'est pas compatible. Pour résoudre ce problème, créez une nouvelle version et déployez-la sur une nouvelle flotte.

exemple

Amazon GameLift ServersExemple n'importe où

```
//Define the server parameters
serverParameters := ServerParameters {
  WebSocketURL: "wss://us-west-1.api.amazongamelift.com",
  ProcessID: "PID1234",
  HostID: "HardwareAnywhere",
  FleetID: "aarn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa",
```

```
AuthToken: "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

Amazon GameLift Servers EC2 exemple géré

```
//Define the server parameters
serverParameters := ServerParameters {}

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
err := server.InitSDK(serverParameters)
```

ProcessReady()

Amazon GameLift Servers indique que le processus du serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après l'avoir invoquée. [InitSDK\(\)](#) Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
func ProcessReady(param ProcessParameters) error
```

Paramètres

ProcessParameters

Un [ProcessParameters](#) objet communique les informations suivantes concernant le processus du serveur :

- Les noms des méthodes de rappel implémentées dans le code du serveur de jeu invoqué par le Amazon GameLift Servers service pour communiquer avec le processus du serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Type de [LogParameters](#) données contenant le chemin d'accès à tous les fichiers spécifiques à une session de jeu que vous souhaitez capturer et Amazon GameLift Servers stocker.

Valeur renvoyée

Renvoie une erreur accompagnée d'un message d'erreur en cas d'échec de la méthode. Renvoie `nil` si la méthode est réussie.

exemple

Cet exemple illustre les implémentations de l'appel [ProcessReady\(\)](#) et de la fonction déléguée.

```
// Define the process parameters
processParams := ProcessParameters {
    OnStartGameSession: gameProcess.OnStartGameSession,
    OnUpdateGameSession: gameProcess.OnGameSessionUpdate,
    OnProcessTerminate: gameProcess.OnProcessTerminate,
    OnHealthCheck: gameProcess.OnHealthCheck,
    Port: port,
    LogParameters: LogParameters {    // logging and error example
        []string {"C:\\game\\logs", "C:\\game\\error"}
    }
}

err := server.ProcessReady(processParams)
```

ProcessEnding()

Amazon GameLift ServersIndique que le processus du serveur est en train de se terminer. Appelez cette méthode après toutes les autres tâches de nettoyage (y compris la fermeture de la session de jeu active) et avant de terminer le processus. En fonction du résultat de `ProcessEnding()`, le processus se termine avec succès (0) ou erreur (-1) et génère un événement de flotte. Si le processus se termine par une erreur, l'événement de flotte généré est `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntaxe

```
func ProcessEnding() error
```

Valeur renvoyée

Renvoie un code d'erreur 0 ou un code d'erreur défini.

exemple

```
// operations to end game sessions and the server process
defer func() {
    err := server.ProcessEnding()
    server.Destroy()
    if err != nil {
        fmt.Println("ProcessEnding() failed. Error: ", err)
        os.Exit(-1)
    } else {
        os.Exit(0)
    }
}
```

ActivateGameSession()

Amazon GameLift Servers indique que le processus du serveur a activé une session de jeu et qu'il est maintenant prêt à recevoir les connexions des joueurs. Cette action est appelée dans le cadre de la fonction de `onStartGameSession()` rappel, après toute initialisation de session de jeu.

Syntaxe

```
func ActivateGameSession() error
```

Valeur renvoyée

Renvoie une erreur accompagnée d'un message d'erreur en cas d'échec de la méthode.

exemple

Cet exemple montre l'`ActivateGameSession()` appel dans le cadre de la fonction de `onStartGameSession()` délégation.

```
func OnStartGameSession(GameSession gameSession) {
    // game-specific tasks when starting a new game session, such as loading map
    // Activate when ready to receive players
    err := server.ActivateGameSession();
}
```


UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur.

Syntaxe

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

Paramètres

playerSessionCreationPolitique

Valeur de chaîne indiquant si la session de jeu accepte de nouveaux joueurs.

Les valeurs valides sont les suivantes :

- **model.AcceptAll**— Acceptez toutes les sessions pour nouveaux joueurs.
- **model.DenyAll**— Refusez toutes les sessions pour les nouveaux joueurs.

Valeur renvoyée

Renvoie une erreur avec un message d'erreur en cas d'échec.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

Récupère l'ID de la session de jeu hébergée par le processus serveur actif.

Syntaxe

```
func GetGameSessionID() (string, error)
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie l'identifiant de session de jeu et aucune erreur. Pour les processus inactifs qui ne sont pas encore activés lors d'une session de jeu, l'appel renvoie une chaîne vide et une `nil` erreur.

exemple

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

Renvoie l'heure à laquelle il est prévu d'arrêter un processus serveur si une heure de fin est disponible. Un processus serveur exécute cette action après avoir reçu un `onProcessTerminate()` rappel de Amazon GameLift Servers. Amazon GameLift Servers appelle `onProcessTerminate()` pour les raisons suivantes :

- Lorsque le processus du serveur a signalé un mauvais état de santé ou n'a pas répondu Amazon GameLift Servers.
- Lorsque vous mettez fin à l'instance lors d'un événement de réduction de la taille.
- Lorsqu'une instance est interrompue en raison d'une interruption [ponctuelle](#).

Syntaxe

```
func GetTerminationTime() (int64, error)
```

Valeur renvoyée

En cas de succès, renvoie l'horodatage en secondes pendant lequel le processus du serveur est programmé pour s'arrêter et une `nil` erreur s'est terminée. La valeur est le délai de résiliation, exprimé en ticks écoulés depuis. `0001 00:00:00` Par exemple, la valeur de la date et de l'heure `2020-09-13 12:26:40 -000Z` est égale à celle `6373559680000000000` des ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

exemple

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Amazon GameLift Servers indique qu'un joueur avec l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide. Une fois la session du joueur validée, le statut de l'emplacement du joueur Amazon GameLift Servers passe de RESERVED à ACTIVE.

Syntaxe

```
func AcceptPlayerSession(playerSessionID string) error
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple traite une demande de connexion qui inclut la validation et le rejet d'une session de joueur non valide. IDs

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {  
    err := server.AcceptPlayerSession(playerSessionID)  
    if err != nil {  
        connection.Accept()  
    } else {  
        connection.Reject(err.Error())  
    }  
}
```

RemovePlayerSession()

Indique Amazon GameLift Servers qu'un joueur s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers change l'emplacement du joueur pour le rendre disponible.

Syntaxe

```
func RemovePlayerSession(playerSessionID string) error
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

Récupère les données de session du joueur, notamment les paramètres, les métadonnées de session et les données du joueur. Utilisez cette méthode pour obtenir des informations sur les points suivants :

- Une session solo
- Toutes les sessions des joueurs dans une session de jeu
- Toutes les sessions de joueur associées à un identifiant de joueur unique

Syntaxe

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)  
    (result.DescribePlayerSessionsResult, error) {
```

```
return srv.describePlayerSessions(&req)
}
```

Paramètres

[DescribePlayerSessionsRequest](#)

Un `DescribePlayerSessionsRequest` objet décrit les sessions de joueur à récupérer.

Valeur renvoyée

En cas de succès, renvoie un `DescribePlayerSessionsResult` objet contenant un ensemble d'objets de session de joueur correspondant aux paramètres de la demande.

exemple

Cet exemple demande toutes les sessions de joueur activement connectées à une session de jeu spécifiée. En omettant `NextToken` et en définissant la valeur limite sur 10, Amazon GameLift Servers renvoie les 10 premiers enregistrements de session de joueur correspondant à la demande.

```
// create request
describePlayerSessionsRequest := request.NewDescribePlayerSessions()
describePlayerSessionsRequest.GameSessionID, _ = server.GetGameSessionID() // get ID
for the current game session
describePlayerSessionsRequest.Limit = 10                                // return the
first 10 player sessions
describePlayerSessionsRequest.PlayerSessionStatusFilter = "ACTIVE"      // Get all
player sessions actively connected to the game session

describePlayerSessionsResult, err :=
server.DescribePlayerSessions(describePlayerSessionsRequest)
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés, Amazon GameLift Servers fournit des données de matchmaking mises à jour à l'aide de la fonction de rappel.

OnUpdateGameSession()

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
    (result.StartMatchBackfillResult, error)
```

Paramètres

[StartMatchBackfillRequest](#)

Un StartMatchBackfillRequest objet communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Ces informations sont facultatives ; si aucun identifiant n'est fourni, il en Amazon GameLift Servers génère un.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur se trouve dans les données du matchmaker de la session de jeu.
- ID de la session de jeu à compléter.
- Les données de matchmaking disponibles pour les joueurs actuels de la session de jeu.

Valeur renvoyée

Renvoie un StartMatchBackfillResult objet avec l'identifiant du ticket de remplacement correspondant, ou un échec avec un message d'erreur.

exemple

```
// form the request
startBackfillRequest := request.NewStartMatchBackfill()
startBackfillRequest.RequestID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff" // optional
startBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
var matchMaker model.MatchmakerData
if err := matchMaker.UnmarshalJSON([]byte(gameSession.MatchmakerData)); err != nil {
    return
}
```

```
startBackfillRequest.Players = matchMaker.Players
res, err := server.StartMatchBackfill(startBackfillRequest)

// Implement callback function for backfill
func OnUpdateGameSession(myGameSession model.GameSession) {
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

StopMatchBackfill()

Annule une demande de remplacement de match active. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Syntaxe

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

Paramètres

[StopMatchBackfillRequest](#)

Un StopMatchBackfillRequest objet qui identifie le ticket de matchmaking à annuler :

- L'identifiant du ticket attribué à la demande de remblayage.
- L'entremetteur à qui la demande de remblayage a été envoyée.
- La session de jeu associée à la demande de remplacement.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
stopBackfillRequest := request.NewStopMatchBackfill() // Use this function to create
request
stopBackfillRequest.TicketID = "1111aaaa-22bb-33cc-44dd-5555eeee66ff"
stopBackfillRequest.MatchmakingConfigurationArn = "arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig"
```

```
//error  
err := server.StopMatchBackfill(stopBackfillRequest)
```

GetComputeCertificate()

Récupère le chemin d'accès au certificat TLS utilisé pour chiffrer la connexion réseau entre le serveur de jeu et votre client de jeu. Vous pouvez utiliser le chemin du certificat lorsque vous enregistrez votre appareil informatique dans un parc Amazon GameLift Servers Anywhere. Pour de plus amples informations, veuillez consulter [RegisterCompute](#).

Syntaxe

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

Valeur renvoyée

Renvoie un `GetComputeCertificateResult` objet contenant les éléments suivants :

- `CertificatePath`: chemin d'accès au certificat TLS sur votre ressource de calcul. Lorsque vous utilisez une flotte Amazon GameLift Servers gérée, ce chemin contient :
 - `certificate.pem`: le certificat d'utilisateur final. La chaîne de certificats complète est la combinaison des `certificateChain.pem` éléments ajoutés à ce certificat.
 - `certificateChain.pem`: chaîne de certificats qui contient le certificat racine et les certificats intermédiaires.
 - `rootCertificate.pem`: le certificat racine.
 - `privateKey.pem`: clé privée pour le certificat d'utilisateur final.
- `ComputeName`: nom de votre ressource de calcul.

exemple

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

Récupère les informations d'identification du rôle de service que vous créez pour étendre les autorisations à votre interlocuteur Services AWS . Amazon GameLift Servers Ces informations

d'identification permettent à votre serveur de jeu d'utiliser vos AWS ressources. Pour de plus amples informations, veuillez consulter [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Syntaxe

```
func GetFleetRoleCredentials(  
    req request.GetFleetRoleCredentialsRequest,  
) (result.GetFleetRoleCredentialsResult, error) {  
    return srv.getFleetRoleCredentials(&req)  
}
```

Paramètres

[GetFleetRoleCredentialsRequest](#)

Des informations d'identification de rôle qui étendent un accès limité à vos AWS ressources au serveur de jeu.

Valeur renvoyée

Renvoie un `GetFleetRoleCredentialsResult` objet contenant les éléments suivants :

- `AssumedRoleUserArn` - Le nom de ressource Amazon (ARN) de l'utilisateur auquel appartient le rôle de service.
- `AssumedRoleId` - L'ID de l'utilisateur auquel appartient le rôle de service.
- `AccessKeyId` - L'ID de clé d'accès pour authentifier et fournir un accès à vos AWS ressources.
- `SecretAccessKey` - L'identifiant de la clé d'accès secrète pour l'authentification.
- `SessionToken` - Un jeton pour identifier la session active en cours qui interagit avec vos AWS ressources.
- `Expiration` : délai avant l'expiration des informations d'identification de votre session.

exemple

```
// form the customer credentials request  
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()  
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/  
exampleGameLiftAction"
```

```
credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Détruire ()

Libère de la mémoire le SDK du serveur de Amazon GameLift Servers jeu. Il est recommandé d'appeler cette méthode après `ProcessEnding()` et avant de terminer le processus. Si vous utilisez une flotte Anywhere et que vous n'interrompez pas les processus du serveur après chaque session de jeu, appelez `Destroy()` puis `InitSDK()` réinitialisez avant de signaler Amazon GameLift Servers que le processus est prêt à héberger une session de jeu avec. `ProcessReady()`

Syntaxe

```
func Destroy() error {  
    return srv.destroy()  
}
```

Valeur renvoyée

Renvoie une erreur accompagnée d'un message d'erreur en cas d'échec de la méthode.

exemple

```
// operations to end game sessions and the server process  
defer func() {  
    err := server.ProcessEnding()  
    server.Destroy()  
    if err != nil {  
        fmt.Println("ProcessEnding() failed. Error: ", err)  
        os.Exit(-1)  
    } else {  
        os.Exit(0)  
    }  
}
```

SDK 5.x du serveur C++ (Unreal) pour Amazon GameLift Servers -- Actions

Utilisez la référence du SDK du serveur Amazon GameLift Servers Unreal 5.x pour vous aider à préparer votre jeu multijoueur en vue de son utilisation avec. Amazon GameLift Servers Pour plus de détails sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#). Si vous utilisez le Amazon GameLift Servers plugin pour Unreal, consultez également [Amazon GameLift Serversplugin pour Unreal Engine](#).

 Note

Cette rubrique décrit l'API Amazon GameLift Servers C++ que vous pouvez utiliser lorsque vous créez pour Unreal Engine. Plus précisément, cette documentation s'applique au code que vous compilez avec l'option `-DBUILD_FOR_UNREAL=1`.

SDK 5.x du serveur C++ (Unreal) pour Amazon GameLift Servers -- Types de données

Utilisez la référence du SDK du serveur Amazon GameLift Servers Unreal 5.x pour vous aider à préparer votre jeu multijoueur en vue de son utilisation avec Amazon GameLift Servers. Pour plus de détails sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#). Si vous utilisez le Amazon GameLift Servers plugin pour Unreal, consultez également [Amazon GameLift Servers plugin pour Unreal Engine](#).

 Note

Cette rubrique décrit l'API Amazon GameLift Servers C++ que vous pouvez utiliser lorsque vous créez pour Unreal Engine. Plus précisément, cette documentation s'applique au code que vous compilez avec l'option `-DBUILD_FOR_UNREAL=1`.

[SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)

Types de données

- [FProcessParamètres](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [FServerParamètres](#)
- [FStartMatchBackfillRequest](#)
- [FPlayer](#)
- [FGameLiftDescribePlayerSessionsRequest](#)
- [FStopMatchBackfillRequest](#)
- [FAttributeValeur](#)
- [FGameLiftGetFleetRoleCredentialsRequest](#)
- [FGameLiftLongOutcome](#)

- [FGameLiftStringOutcome](#)
- [FGameLiftDescribePlayerSessionsOutcome](#)
- [FGameLiftDescribePlayerSessionsResult](#)
- [FGenericRésultat](#)
- [FGameLiftPlayerSession](#)
- [FGameLiftGetComputeCertificateOutcome](#)
- [FGameLiftGetComputeCertificateResult](#)
- [FGameLiftGetFleetRoleCredentialsOutcome](#)
- [FGetFleetRoleCredentialsResult](#)
- [FGameLiftError](#)
- [Enums](#)

FProcessParamètres

Ce type de données contient l'ensemble des paramètres envoyés à Amazon GameLift Servers dans un [ProcessReady\(\)](#).

Propriétés	Description
LogParameters	Objet avec des chemins de répertoire vers des fichiers générés au cours d'une session de jeu. Amazon GameLift Servers copie et stocke les fichiers pour un accès futur. Type : TArray<FString> Obligatoire : non
OnHealthCheck	La fonction de rappel qui Amazon GameLift Servers appelle le processus du serveur pour demander un rapport d'état de santé. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes et attend une réponse pendant 60 secondes. Le processus du serveur revient TRUE s'il est sain, FALSE sinon sain. Si aucune réponse n'est renvoyée, Amazon GameLift

Servers enregistre le processus du serveur comme non sain.

Cette propriété est une fonction déléguée définie comme suit `DECLARE_DELEGATE_R` et `Val(bool, FOnHealthCheck)` :

Type : `FOnHealthCheck`

Obligatoire : non

OnProcessTerminate

La fonction de rappel qui Amazon GameLift Servers invoque pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attendez 5 minutes que le processus serveur s'arrête et répondez par un [ProcessEnding\(\)](#) appel avant d'arrêter le processus serveur.

Type : `FSimpleDelegate`

Obligatoire : oui

OnStartGameSession

La fonction de rappel qui Amazon GameLift Servers invoque une nouvelle session de jeu pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à une demande du client [CreateGameSession](#). La fonction de rappel transmet un [GameSession](#) objet.

Cette propriété est une fonction déléguée définie comme `DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession);`

Type : `FOnStartGameSession`

Obligatoire : oui

OnUpdateGameSession	La fonction de rappel qui Amazon GameLift Servers appelle pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de remplissage de correspondances a été traitée pour fournir des données de matchmaking mises à jour. Il transmet un GameSession objet, une mise à jour de statut (updateReason) et l'identifiant du ticket de remplacement des matchs. Cette propriété est une fonction déléguée définie comme DECLARE_DELEGATE_OneParam(FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession); Type : FOnUpdateGameSession Obligatoire : non
Port	Numéro de port sur lequel le processus serveur écoute les connexions des nouveaux joueurs. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur. Type : int Obligatoire : oui

UpdateGameSession

Ce type de données est mis à jour vers un objet de session de jeu, qui inclut la raison pour laquelle la session de jeu a été mise à jour et l'identifiant du ticket de remplissage associé si le remplissage est utilisé pour remplir les sessions des joueurs pendant la session de jeu.

Propriétés	Description
GameSession	Un objet GameSession. L'GameSession objet contient des propriétés décrivant une session de jeu. Type : <code>Aws::GameLift::Server::GameSession</code> Obligatoire : non
UpdateReason	La raison pour laquelle la session de jeu est mise à jour. Type : <code>enum class UpdateReason</code> • DONNÉES DE MATCHMAKING MISES À JOUR • BACKFILL_FAILED • BACKFILL_TIMED_OUT • BACKFILL_ANNULÉ Obligatoire : non
BackfillTicketId	L'identifiant du ticket de remplacement qui tente de mettre à jour la session de jeu. Type : <code>char[]</code> Obligatoire : non

GameSession

Ce type de données fournit les détails d'une session de jeu.

Propriétés	Description
GameSessionId	Identifiant unique pour la session de jeu. L'ARN d'une session de jeu a le format suivant : <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Type : <code>char[]</code> Obligatoire : non
Nom	Une étiquette descriptive de la session de jeu. Type : <code>char[]</code> Obligatoire : non
FleetId	Identifiant unique de la flotte sur laquelle s'exécute la session de jeu. Type : <code>char[]</code> Obligatoire : non
MaximumPlayerSessionCount	Le nombre maximum de connexions de joueurs à la session de jeu. Type : <code>int</code> Obligatoire : non
Port	Le numéro de port de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port.

Propriétés	Description
	Type : <code>int</code> Obligatoire : non
IpAddress	Adresse IP de la session de jeu. Pour se connecter à un serveur de Amazon GameLift Servers jeu, une application a besoin à la fois de l'adresse IP et du numéro de port. Type : <code>char[]</code> Obligatoire : non
GameSessionData	Ensemble de propriétés de session de jeu personnalisées, mises en forme en tant que valeur de chaîne unique. Type : <code>char[]</code> Obligatoire : non
MatchmakerData	Informations sur le processus de matchmaking utilisé pour créer la session de jeu, en syntaxe JSON, formatée sous forme de chaîne. En plus de la configuration de matchmaking utilisée, il contient des données sur tous les joueurs affectés au match, y compris les attributs des joueurs et les affectations des équipes. Type : <code>char[]</code> Obligatoire : non

Propriétés	Description
GameProperties	Ensemble de propriétés personnalisées pour une session de jeu, formatées sous forme de paires clé:valeur. Ces propriétés sont transmises avec une demande de démarrage d'une nouvelle session de jeu. Type : <code>GameProperty[]</code> Obligatoire : non
DnsName	Identifiant DNS attribué à l'instance qui exécute la session de jeu. Les valeurs ont le format suivant : - Flottes compatibles TLS : <code><unique identifieur>.<region identifieur>.amazongamelift.com</code> - Non-TLS-enabled flottes : <code>ec2-<unique identifieur>.compute.amazonaws.com</code> Lorsque vous vous connectez à une session de jeu exécutée sur une flotte compatible TLS, vous devez utiliser le nom DNS et non l'adresse IP. Type : <code>char[]</code> Obligatoire : non

FServerParamètres

Informations utilisées pour maintenir la connexion entre un serveur Amazon GameLift Servers Anywhere et le Amazon GameLift Servers service. Ces informations sont utilisées lors du lancement de nouveaux processus serveur avec [InitSDK\(\)](#). Pour les serveurs hébergés sur EC2 des instances Amazon GameLift Servers gérées, utilisez un objet vide.

Propriétés	Description
webSocketUrl	Les <code>GameLiftServerSdkEndpoint</code> Amazon GameLift Servers retours lorsque vous RegisterCompute recherchez une ressource de calcul Amazon GameLift Servers Anywhere. Type : <code>char[]</code> Obligatoire : oui
ID du processus	Un identifiant unique enregistré auprès du processus serveur hébergeant votre jeu. Type : <code>char[]</code> Obligatoire : oui
hostId	Le <code>HostID</code> est celui <code>ComputeName</code> utilisé lorsque vous avez enregistré votre ordinateur. Pour plus d'informations, voir, RegisterCompute. Type : <code>char[]</code> Obligatoire : oui
ID de flotte	Identifiant unique de la flotte dans laquelle le calcul est enregistré. Pour plus d'informations, voir, RegisterCompute. Type : <code>char[]</code> Obligatoire : oui
Jeton d'authentification	Le jeton d'authentification généré par Amazon GameLift Servers celui-ci authentifie votre serveur auprès de Amazon GameLift Servers. Pour plus d'informations, voir, GetComputeAuthToken.

Propriétés	Description
	Type : <code>char[]</code>
	Obligatoire : oui

FStartMatchBackfillRequest

Informations utilisées pour créer une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations Amazon GameLift Servers lors d'un [StartMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Un identifiant de session de jeu unique. L'opération API GetGameSessionId renvoie l'identifiant au format ARN. Type : <code>char[]</code> Obligatoire : oui
MatchmakingConfigurationArn	Un identifiant unique, sous la forme d'un ARN, que le système de jumelage doit utiliser pour cette demande. L'ARN de matchmaking pour la session de jeu d'origine se trouve dans l'objet de session de jeu dans la propriété de données du matchmaker. Pour en savoir plus sur les données du système de jumelage, consultez la section Travailler avec les données du système de jumelage . Type : <code>char[]</code> Obligatoire : oui
Joueurs	Ensemble de données représentant tous les joueurs participant à la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constitue

Propriétés	Description
	nt de bonnes correspondances pour les joueurs actuels. Type : TArray<FPlayer> Obligatoire : oui
TicketId	Un identifiant unique pour un ticket de demande de matchmaking ou de remplacement de match. Si vous ne fournissez aucune valeur, il en Amazon GameLift Servers génère une. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire. Type : char[] Obligatoire : non

FPlayer

Ce type de données représente un joueur dans le matchmaking. Lors du lancement d'une demande de matchmaking, un joueur dispose d'un identifiant de joueur, d'attributs et éventuellement de données de latence. Amazon GameLift Servers ajoute des informations sur l'équipe après la fin d'un match.

Propriétés	Description
LatencyInMS	Ensemble de valeurs exprimées en millisecondes qui indiquent le niveau de latence ressenti par un joueur lorsqu'il est connecté à un lieu. Si cette propriété est utilisée, le joueur n'est jumelé qu'aux emplacements répertoriés. Si un matchmaker dispose d'une règle qui évalue la latence, les joueurs doivent indiquer la latence pour être mis en relation.

Propriétés	Description
	Type : TMap>FString, int32< Obligatoire : non
PlayerAttributes	Une collection de paires clé:valeur contenant des informations sur les joueurs à utiliser dans le matchmaking. Les clés d'attribut du joueur doivent correspondre à PlayerAttributes celles utilisées dans un ensemble de règles de matchmaking. Pour plus d'informations sur les attributs des joueurs, consultez AttributeValue. Type : TMap>FString, FAttributeValue< Obligatoire : non
PlayerId	Identifiant unique pour un joueur. Type : std::string Obligatoire : non
Equipe	Le nom de l'équipe à laquelle le joueur est affecté lors d'un match. Vous définissez le nom de l'équipe dans le jeu de règles de matchmaking. Type : FString Obligatoire : non

FGameLiftDescribePlayerSessionsRequest

Objet qui indique les sessions de joueur à récupérer. Le processus du serveur fournit ces informations en [DescribePlayerSessions\(\)](#) appelant Amazon GameLift Servers.

Propriétés	Description
GameSessionId	Un identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'identifiant de session de jeu est FString. GameSessionID Il s'agit d'une chaîne d'identification personnalisée ou d'un Type : std::string Obligatoire : non
PlayerSessionId	Identifiant unique d'une session de joueur. Utilisez ce paramètre pour demander une session de joueur spécifique. Type : FString Obligatoire : non
PlayerId	L'identifiant unique d'un joueur. Utilisez ce paramètre pour demander toutes les sessions de joueur pour un joueur spécifique. Consultez Générer un joueur IDs. Type : FString Obligatoire : non
PlayerSessionStatusFilter	État de la session du joueur sur lequel filtrer les résultats. Les statuts de session de joueur possibles incluent : • RÉSERVÉ — La demande de session du joueur a été reçue, mais le joueur ne s'est pas connecté au processus du serveur ni n'a été validé.

Propriétés	Description
	• ACTIF — Le joueur a été validé par le processus du serveur et est connecté. • TERMINÉ — La connexion du joueur a été interrompue. • TIMEDOUT — Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté ou n'a pas été validée dans le délai imparti (60 secondes). Type : FString Obligatoire : non
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : FString Obligatoire : non
Limite	Nombre maximal de résultats à renvoyer. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : int Obligatoire : non

FStopMatchBackfillRequest

Informations utilisées pour annuler une demande de remplissage par matchmaking. Le serveur de jeu communique ces informations au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Propriétés	Description
GameSessionArn	Identifiant de session de jeu unique de la demande annulée. Type : FString Obligatoire : oui
MatchmakingConfigurationArn	Un identifiant unique du matchmaker auquel cette demande a été envoyée. Type : FString Obligatoire : oui
TicketId	Identifiant unique du ticket de demande de remblayage à annuler. Type : FString Obligatoire : oui

FAttributeValeur

Utilisez ces valeurs dans les paires [FPlayer](#) clé-valeur d'attribut. Cet objet vous permet de spécifier une valeur d'attribut à l'aide de n'importe quel type de données valide : chaîne, nombre, tableau de chaînes ou mappage de données. Chaque `AttributeValue` objet ne peut utiliser qu'une seule des propriétés disponibles.

Propriétés	Description
Type ATR	Spécifie le type de valeur d'attribut. Type : valeur <code>FAttributeType</code> enum. Obligatoire : non
S	Représente une valeur d'attribut de chaîne.

Propriétés	Description
	Type : FString Obligatoire : non
N	Représente une valeur d'attribut numérique. Type : double Obligatoire : non
SL	Représente un tableau de valeurs d'attributs de chaîne. Type : TArray<FString> Obligatoire : non
SDM	Représente un dictionnaire de clés de chaîne et de valeurs doubles. Type : TMap<FString, double> Obligatoire : non

FGameLiftGetFleetRoleCredentialsRequest

Ce type de données fournit des informations d'identification de rôle qui étendent l'accès limité à vos AWS ressources au serveur de jeu. Pour plus d'informations, voir [Configurer un rôle de service IAM pour Amazon GameLift Servers](#).

Propriétés	Description
RoleArn	Le nom de ressource Amazon (ARN) du rôle de service qui étend un accès limité à vos AWS ressources. Type : FString Obligatoire : non

Propriétés	Description
RoleSessionName	Nom de la session décrivant l'utilisation des informations d'identification du rôle. Type : FString Obligatoire : non

FGameLiftLongOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : long Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : long&& Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “FGameLiftError” Obligatoire : non

FGameLiftStringOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : <code>FString</code> Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>FString&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “FGameLiftError” Obligatoire : non

FGameLiftDescribePlayerSessionsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “FGameLiftDescribePlayerSessionsResult”

Propriétés	Description
	Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : <code>FGameLiftDescribePlayerSessionsResult&&</code> Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : <code>bool</code> Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "FGameLiftError" Obligatoire : non

FGameLiftDescribePlayerSessionsResult

Propriétés	Description
PlayerSessions	Type : <code>TArray<FGameLiftPlayerSession></code> Obligatoire : oui
NextToken	Le jeton indiquant le début de la page de résultats suivante. Pour spécifier le début du jeu de résultats, ne fournissez aucune valeur. Si vous fournissez un identifiant de session de joueur, ce paramètre est ignoré. Type : <code>FString</code>

Propriétés	Description
	Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "FGameLiftError" Obligatoire : non

FGenericRésultat

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "FGameLiftError" Obligatoire : non

FGameLiftPlayerSession

Propriétés	Description
CreationTime	Type : long

Propriétés	Description
	Obligatoire : oui
FleetId	Type : FString Obligatoire : oui
GameSessionId	Type : FString Obligatoire : oui
IpAddress	Type : FString Obligatoire : oui
PlayerData	Type : FString Obligatoire : oui
PlayerId	Type : FString Obligatoire : oui
PlayerSessionId	Type : FString Obligatoire : oui
Port	Type : int Obligatoire : oui
Statut	Type : Une PlayerSessionStatus énumération. Obligatoire : oui
TerminationTime	Type : long Obligatoire : oui
DnsName	Type : FString Obligatoire : oui

FGameLiftGetComputeCertificateOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “FGameLiftGetComputeCertificateResult” Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : FGameLiftGetComputeCertificateResult&& Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “FGameLiftError” Obligatoire : non

FGameLiftGetComputeCertificateResult

Le chemin d'accès au certificat TLS sur votre ordinateur et le nom d'hôte du calcul.

Propriétés	Description
CertificatePath	Type : FString Obligatoire : oui

Propriétés	Description
ComputeName	Type : FString Obligatoire : oui

FGameLiftGetFleetRoleCredentialsOutcome

Ce type de données résulte d'une action et produit un objet doté des propriétés suivantes :

Propriétés	Description
Résultat	Le résultat de l'action. Type : the section called “FGetFleetRoleCredentialsResult” Obligatoire : non
ResultWithOwnership	Le résultat de l'action, converti en valeur rvalue, afin que le code appelant puisse s'approprier l'objet. Type : FGameLiftGetFleetRoleCredentialsResult&& Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called “FGameLiftError” Obligatoire : non

FGetFleetRoleCredentialsResult

Propriétés	Description
AccessKeyId	ID de clé d'accès permettant d'authentifier et de fournir un accès à vos AWS ressources. Type : FString Obligatoire : non
AssumedRoleId	ID de l'utilisateur auquel appartient le rôle de service. Type : FString Obligatoire : non
AssumedRoleUserArn	Le nom de ressource Amazon (ARN) de l'utilisateur auquel appartient le rôle de service. Type : FString Obligatoire : non
Expiration	Durée avant l'expiration de vos informations d'identification de session. Type : FDateTime Obligatoire : non
SecretAccessKey	ID de clé d'accès secrète pour l'authentification. Type : FString Obligatoire : non
SessionToken	Un jeton permettant d'identifier la session active en cours qui interagit avec vos AWS ressources. Type : FString

Propriétés	Description
	Obligatoire : non
Réussite	Que l'action ait été couronnée de succès ou non. Type : bool Obligatoire : oui
Erreur	Erreur survenue en cas d'échec de l'action. Type : the section called "GameLiftError" Obligatoire : non

FGameLiftError

Propriétés	Description
ErrorType	Type d'erreur. Type : Une GameLiftErrorType énumération . Obligatoire : non
ErrorMessage	Nom de l'erreur. Type : std::string Obligatoire : non
ErrorMessage	Message d'erreur. Type : std::string Obligatoire : non

Enums

Les énumérations définies pour le SDK du serveur pour Amazon GameLift Servers (Unreal) sont définies comme suit :

FAttributeType

- NONE
- CHAÎNE
- DOUBLE
- LISTE_CHAÎNES
- STRING_DOUBLE_MAP

GameLiftErrorType

Valeur de chaîne indiquant le type d'erreur. Les valeurs valides sont les suivantes :

- SERVICE_CALL_FAILED — Un appel à un AWS service a échoué.
- LOCAL_CONNECTION_FAILED — La connexion locale a échoué. Amazon GameLift Servers
- NETWORK_NOT_INITIALIZED — Le réseau n'a pas été initialisé.
- GAMESESSION_ID_NOT_SET — L'ID de session de jeu n'a pas été défini.
- MAUVAISE DEMANDE D'EXCEPTION
- EXCEPTION DE SERVICE INTERNE
- ALREADY_INITIALIZED — Le Amazon GameLift Servers serveur ou le client a déjà été initialisé avec Initialize ().
- FLEET_MISMATCH — La flotte cible ne correspond pas à la flotte d'une GameSession ou d'une PlayerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED — Le client n'a pas été initialisé. Amazon GameLift Servers
- GAMELIFT_SERVER_NOT_INITIALIZED — Le serveur n'a pas été initialisé. Amazon GameLift Servers
- GAME_SESSION_ENDED_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin de la session de jeu.
- GAME_SESSION_NOT_READY — La Amazon GameLift Servers session de jeu sur le serveur n'a pas été activée.
- GAME_SESSION_READY_FAILED : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que la session de jeu est prête.

- **INITIALIZATION_MISMATCH** — Une méthode client a été appelée après `Server::Initialize()`, ou vice versa.
- **NOT_INITIALIZED** — Le Amazon GameLift Servers serveur ou le client n'a pas été initialisé avec `Initialize()`.
- **NO_TARGET_ALIASID_SET** — Aucun `AliasID` cible n'a été défini.
- **NO_TARGET_FLEET_SET** — Aucune flotte cible n'a été définie.
- **PROCESS_ENDING_FAILED** : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler la fin du processus.
- **PROCESS_NOT_ACTIVE** — Le processus du serveur n'est pas encore actif, n'est pas lié à un et ne peut ni `GameSession` accepter ni traiter. `PlayerSessions`
- **PROCESS_NOT_READY** — Le processus du serveur n'est pas encore prêt à être activé.
- **PROCESS_READY_FAILED** : le SDK du serveur n'a pas Amazon GameLift Servers pu contacter le service pour signaler que le processus est prêt.
- **SDK_VERSION_DETECTION_FAILED** — La détection de la version du SDK a échoué.
- **STX_CALL_FAILED** — Un appel au composant principal du XStx serveur a échoué.
- **STX_INITIALIZATION_FAILED** — Le composant principal du XStx serveur n'a pas pu être initialisé.
- **UNEXPECTED_PLAYER_SESSION** — Le serveur a détecté une session de joueur non enregistrée.
- **ÉCHEC DE CONNEXION WEBSOCKET**
- **WEBSOCKET_CONNECT_FAILURE_FORBIDDEN**
- **WEBSOCKET_CONNECT_FAILURE_INVALID_URL**
- **WEBSOCKET_CONNECT_FAILURE_TIMEOUT**
- **WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE** — Échec récupérable lors de l'envoi d'un message au service. `GameLift WebSocket`
- **WEBSOCKET_SEND_MESSAGE_FAILURE** — Impossible d'envoyer un message au service. `GameLift WebSocket`
- **MATCH_BACKFILL_REQUEST_VALIDATION** — La validation de la demande a échoué.
- **PLAYER_SESSION_REQUEST_VALIDATION** — La validation de la demande a échoué.

`EPlayerSessionCreationPolicy`

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs. Les valeurs valides sont les suivantes :

- `ACCEPT_ALL` — Accepte toutes les sessions de nouveaux joueurs.
- `DENY_ALL` — Refuse toutes les sessions de nouveaux joueurs.
- `NOT_SET` — La session de jeu n'est pas configurée pour accepter ou refuser les sessions de nouveaux joueurs.

EPlayerSessionStatus

- `ACTIF`
- `TERMINÉ`
- `NOT_SET`
- `RÉSERVÉ`
- `DÉLAI EXPIRÉ`

[SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Types de données](#)

Rubriques

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Détruire \(\)](#)

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
FGameLiftStringOutcome GetSdkVersion();
```

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet [the section called "FGameLiftStringOutcome"](#). L'objet renvoyé inclut le numéro de version (exemple 5.0.0). En cas d'échec, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialise le Amazon GameLift Servers SDK pour un parc géré EC2 . Appelez cette méthode au lancement, avant que toute autre initialisation associée ne se Amazon GameLift Servers produise. Cette méthode lit les paramètres du serveur depuis l'environnement hôte afin de configurer la communication entre le serveur et le Amazon GameLift Servers service. Il utilise un jeton d'idempotencie, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Syntaxe

```
FGameLiftGenericOutcome InitSDK()
```

Valeur renvoyée

En cas de succès, renvoie un InitSdkOutcome objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

exemple

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent  
to enable further communication.
```



```
FGameLiftGenericOutcome InitSdkOutcome = GameLiftSdkModule->InitSDK();
```

InitSDK()

Initialise le Amazon GameLift Servers SDK pour une flotte Anywhere ou une flotte de conteneurs gérés. Appelez cette méthode au lancement, avant que toute autre initialisation associée ne se produise. Cette méthode nécessite des paramètres de serveur explicites pour configurer la communication entre le serveur et le Amazon GameLift Servers service. Il utilise un jeton d'idempotence, de sorte que vous pouvez réessayer cet appel en toute sécurité en cas d'échec.

Syntaxe

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

Paramètres

[FServerParamètres](#)

Pour initialiser un serveur de jeu sur une flotte Amazon GameLift Servers Anywhere, créez un `ServerParameters` objet avec les informations suivantes :

- URL WebSocket utilisée pour vous connecter à votre serveur de jeu.
- ID du processus utilisé pour héberger votre serveur de jeu.
- L'ID de l'ordinateur hébergeant les processus de votre serveur de jeu.
- L'ID du Amazon GameLift Servers parc contenant votre ordinateur Amazon GameLift Servers Anywhere.
- Le jeton d'autorisation généré par l'Amazon GameLift Servers opération.

Valeur renvoyée

En cas de succès, renvoie un `InitSdkOutcome` objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

Note

Si les appels à échouent pour `InitSDK()` les builds de jeu déployés sur des flottes Anywhere, vérifiez le `ServerSdkVersion` paramètre utilisé lors de la création de la ressource de build. Vous devez définir explicitement cette valeur en fonction de la version du SDK du serveur utilisée. La valeur par défaut de ce paramètre est 4.x, ce qui n'est pas

compatible. Pour résoudre ce problème, créez une nouvelle version et déployez-la sur une nouvelle flotte.

exemple

```
//Define the server parameters
FServerParameters serverParameters;
parameters.m_authToken = "1111aaaa-22bb-33cc-44dd-5555eeee66ff";
parameters.m_fleetId = "arn:aws:gamelift:us-west-1:111122223333:fleet/
fleet-9999ffff-88ee-77dd-66cc-5555bbbb44aa";
parameters.m_hostId = "HardwareAnywhere";
parameters.m_processId = "PID1234";
parameters.m_webSocketUrl = "wss://us-west-1.api.amazongamelift.com";

//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK(serverParameters);
```

ProcessReady()

Amazon GameLift Servers indique que le processus du serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après l'avoir invoquée. [InitSDK\(\)](#) Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Paramètres

processParameters

[FProcessParameters](#) Objet communiquant les informations suivantes concernant le processus du serveur :

- Noms des méthodes de rappel implémentées dans le code du serveur de jeu que le Amazon GameLift Servers service invoque pour communiquer avec le processus du serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre les implémentations de l'appel [ProcessReady\(\)](#) et de la fonction déléguée.

```
//Calling ProcessReady tells Amazon GameLift Servers this game server is ready to
receive incoming game sessions!
UE_LOG(GameServerLog, Log, TEXT("Calling Process Ready"));
FGameLiftGenericOutcome processReadyOutcome = GameLiftSdkModule-
>ProcessReady(*params);
```

ProcessEnding()

Amazon GameLift Servers indique que le processus du serveur est en train de se terminer. Appelez cette méthode après toutes les autres tâches de nettoyage (y compris la fermeture de la session de jeu active) et avant de terminer le processus. En fonction du résultat de `ProcessEnding()`, le processus se termine avec succès (0) ou erreur (-1) et génère un événement de flotte. Si le processus se termine par une erreur, l'événement de flotte généré est `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Syntaxe

```
FGameLiftGenericOutcome ProcessEnding()
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
//OnProcessTerminate callback. Amazon GameLift Servers will invoke this callback before
shutting down an instance hosting this game server.
//It gives this game server a chance to save its state, communicate with services,
etc., before being shut down.
//In this case, we simply tell Amazon GameLift Servers we are indeed going to shutdown.
params->OnTerminate.BindLambda( [=]() {
    UE_LOG(GameServerLog, Log, TEXT("Game Server Process is terminating"));
    GameLiftSdkModule->ProcessEnding();
});
```

ActivateGameSession()

Amazon GameLift Servers indique que le processus du serveur a activé une session de jeu et qu'il est maintenant prêt à recevoir les connexions des joueurs. Cette action doit être appelée dans le cadre de la fonction de `onStartGameSession()` rappel, après toute initialisation de session de jeu.

Syntaxe

```
FGameLiftGenericOutcome ActivateGameSession()
```

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple montre l'`ActivateGameSession()` appel dans le cadre de la fonction de `onStartGameSession()` délégation.

```
//When a game session is created, Amazon GameLift Servers sends an activation request
//to the game server and passes along the game session object containing game properties
//and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
//invoke GameLiftServerAPI.ActivateGameSession()
auto onGameSession = [](Aws::GameLift::Server::Model::GameSession gameSession)
{
    FString gameId = FString(gameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"), *gameId);
    GameLiftSdkModule->ActivateGameSession();
};
```

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur.

Syntaxe

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
policy)
```

Paramètres

playerCreationSessionPolitique

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs.

Les valeurs valides sont les suivantes :

- ACCEPT_ALL — Accepte toutes les sessions de nouveaux joueurs.
- DENY_ALL — Refuse toutes les sessions de nouveaux joueurs.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
FGameLiftGenericOutcome outcome = GameLiftSdkModule-  
>UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

Récupère l'ID de la session de jeu hébergée par le processus serveur actif.

Pour les processus inactifs qui ne sont pas activés lors d'une session de jeu, l'appel renvoie un [the section called "FGameLiftError"](#).

Syntaxe

```
FGameLiftStringOutcome GetGameSessionId()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet [the section called "FGameLiftStringOutcome"](#). En cas d'échec, renvoie un message d'erreur. »

Pour les processus inactifs qui ne sont pas activés lors d'une session de jeu, l'appel renvoie `Success = True` et `GameSessionId = ""`.

exemple

```
//When a game session is created, Amazon GameLift Servers sends an activation request
//to the game server and passes along the game session object containing game properties
//and other settings.
//Here is where a game server should take action based on the game session object.
//Once the game server is ready to receive incoming player connections, it should
//invoke GameLiftServerAPI.ActivateGameSession()
auto onGameSession = [=](Aws::GameLift::Server::Model::GameSession gameSession)
{
    FString gameSessionId = FString(gameSession.GetGameSessionId());
    UE_LOG(GameServerLog, Log, TEXT("GameSession Initializing: %s"), *gameSessionId);
    GameLiftSdkModule->ActivateGameSession();
};
```

GetTerminationTime()

Renvoie l'heure d'arrêt planifiée pour un processus serveur, si une heure de résiliation est disponible. Un processus serveur prend des mesures après avoir reçu un `onProcessTerminate()` rappel de Amazon GameLift Servers. Amazon GameLift Servers appelle `onProcessTerminate()` pour les raisons suivantes :

- Lorsque le processus du serveur a signalé un mauvais état de santé ou n'a pas répondu Amazon GameLift Servers.
- Lors de la mise hors service de l'instance lors d'un événement de réduction de la taille.
- Lorsqu'une instance est interrompue en raison d'une interruption [ponctuelle](#).

Syntaxe

```
AwsDateTimeOutcome GetTerminationTime()
```

Valeur renvoyée

En cas de succès, renvoie l'heure de fin sous forme d'`AwsDateTimeOutcome` objet. La valeur est le délai de résiliation, exprimé en ticks écoulés depuis. `0001 00:00:00` Par exemple, la valeur de la date et de l'heure `2020-09-13 12:26:40 -000Z` est égale à celle `6373559680000000000` des ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

Si le processus n'a pas reçu de `ProcessParameters.OnProcessTerminate()` rappel, un message d'erreur est renvoyé. Pour plus d'informations sur l'arrêt d'un processus serveur, consultez [Répondre à une notification d'arrêt du processus du serveur](#).

exemple

```
AwsDateTimeOutcome TermTimeOutcome = GameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Amazon GameLift Servers indique qu'un joueur avec l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide. Une fois la session du joueur validée, le statut de l'emplacement du joueur Amazon GameLift Servers passe de RÉSERVÉ à ACTIF.

Syntaxe

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Paramètres

`playerSessionId`

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple traite une demande de connexion qui inclut la validation et le rejet d'une session de joueur non valide. IDs

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerId, const
    FString& playerId)
{
    #if WITH_GAMELIFT
    UE_LOG(GameServerLog, Log, TEXT("Accepting GameLift PlayerSession: %s . PlayerId:
    %s"), *playerSessionId, *playerId);
```

```
FString gsId = GetCurrentGameSessionId();
if (gsId.IsEmpty()) {
    UE_LOG(GameServerLog, Log, TEXT("No GameLift GameSessionId. Returning early!"));
    return false;
}

if (!GameLiftSdkModule->AcceptPlayerSession(playerSessionId).IsSuccess()) {
    UE_LOG(GameServerLog, Log, TEXT("PlayerSession not Accepted."));
    return false;
}

// Add PlayerSession from internal data structures keeping track of connected players
connectedPlayerSessionIds.Add(playerSessionId);
idToPlayerSessionMap.Add(playerSessionId, PlayerSession{ playerId,
playerSessionId });
return true;
#else
return false;
#endif
}
```

RemovePlayerSession()

Indique Amazon GameLift Servers qu'un joueur s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers change l'emplacement du joueur pour le rendre disponible.

Syntaxe

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
bool GameLiftManager::RemovePlayerSession(const FString& playerSessionId)
{
    #if WITH_GAMELIFT
        UE_LOG(GameServerLog, Log, TEXT("Removing GameLift PlayerSession: %s"),
            *playerSessionId);

        if (!GameLiftSdkModule->RemovePlayerSession(playerSessionId).IsSuccess()) {
            UE_LOG(GameServerLog, Log, TEXT("PlayerSession Removal Failed"));
            return false;
        }

        // Remove PlayerSession from internal data structures that are keeping track of
        connected players
        connectedPlayerSessionIds.Remove(playerSessionId);
        idToPlayerSessionMap.Remove(playerSessionId);

        // end the session if there are no more players connected
        if (connectedPlayerSessionIds.Num() == 0) {
            EndSession();
        }

        return true;
    #else
        return false;
    #endif
}
```

DescribePlayerSessions()

Récupère les données de session du joueur, notamment les paramètres, les métadonnées de session et les données du joueur. Utilisez cette méthode pour obtenir des informations sur les points suivants :

- Une session solo
- Toutes les sessions des joueurs au cours d'une session de jeu
- Toutes les sessions de joueur associées à un identifiant de joueur unique

Syntaxe

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Paramètres

FGameLiftDescribePlayerSessionsRequest

[the section called “FGameLiftDescribePlayerSessionsRequest”](#) Objet qui décrit les sessions de joueur à récupérer.

Valeur renvoyée

En cas de réussite, renvoie un objet [the section called “FGameLiftDescribePlayerSessionsOutcome”](#) qui contient un ensemble d'objets de session de joueur correspondant aux paramètres de la demande.

exemple

Cet exemple demande toutes les sessions de joueur activement connectées à une session de jeu spécifiée. En omettant NextToken en définissant la valeur limite sur 10, Amazon GameLift Servers renvoie les 10 premiers enregistrements de session de joueur correspondant à la demande.

```
void GameLiftManager::DescribePlayerSessions()  
{  
    #if WITH_GAMELIFT  
    FString localPlayerSessions;  
    for (auto& psId : connectedPlayerSessionIds)  
    {  
        PlayerSession ps = idToPlayerSessionMap[psId];  
        localPlayerSessions += FString::Printf(TEXT("%s : %s ; "), *(ps.playerSessionId),  
*(ps.playerId));  
    }  
    UE_LOG(GameServerLog, Log, TEXT("LocalPlayerSessions: %s"), *localPlayerSessions);  
  
    UE_LOG(GameServerLog, Log, TEXT("Describing PlayerSessions in this GameSession"));  
    FGameLiftDescribePlayerSessionsRequest request;  
    request.m_gameSessionId = GetCurrentGameSessionId();  
  
    FGameLiftDescribePlayerSessionsOutcome outcome = GameLiftSdkModule-  
>DescribePlayerSessions(request);
```

```
LogDescribePlayerSessionsOutcome(outcome);  
#endif  
}
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés, Amazon GameLift Servers fournit des données de matchmaking mises à jour à l'aide de la fonction de rappel.

OnUpdateGameSession()

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest  
&startBackfillRequest);
```

Paramètres

[FStartMatchBackfillRequest](#)

Un StartMatchBackfillRequest objet qui communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Ces informations sont facultatives ; si aucun identifiant n'est fourni, il en Amazon GameLift Servers générera un.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur se trouve dans les données du matchmaker de la session de jeu.
- ID de la session de jeu à compléter.
- Les données de matchmaking disponibles pour les joueurs actuels de la session de jeu.

Valeur renvoyée

Renvoie un StartMatchBackfillOutcome objet avec l'identifiant du ticket de remplacement correspondant, ou un échec avec un message d'erreur.

exemple

```

FGameLiftStringOutcome FGameLiftServerSDKModule::StartMatchBackfill(const
    FStartMatchBackfillRequest& request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::StartMatchBackfillRequest sdkRequest;
    sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
    sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

    sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));
    for (auto player : request.m_players) {
        Aws::GameLift::Server::Model::Player sdkPlayer;
        sdkPlayer.SetPlayerId(TCHAR_TO_UTF8(*player.m_playerId));
        sdkPlayer.SetTeam(TCHAR_TO_UTF8(*player.m_team));
        for (auto entry : player.m_latencyInMs) {
            sdkPlayer.WithLatencyMs(TCHAR_TO_UTF8(*entry.Key), entry.Value);
        }

        std::map<std::string, Aws::GameLift::Server::Model::AttributeValue>
        sdkAttributeMap;
        for (auto attributeEntry : player.m_playerAttributes) {
            FAttributeValue value = attributeEntry.Value;
            Aws::GameLift::Server::Model::AttributeValue attribute;
            switch (value.m_type) {
                case FAttributeType::STRING:
                    attribute =
            Aws::GameLift::Server::Model::AttributeValue(TCHAR_TO_UTF8(*value.m_S));
                    break;
                case FAttributeType::DOUBLE:
                    attribute = Aws::GameLift::Server::Model::AttributeValue(value.m_N);
                    break;
                case FAttributeType::STRING_LIST:
                    attribute =
            Aws::GameLift::Server::Model::AttributeValue::ConstructStringList();
                    for (auto sl : value.m_SL) {
                        attribute.AddString(TCHAR_TO_UTF8(*sl));
                    };
                    break;
                case FAttributeType::STRING_DOUBLE_MAP:
                    attribute =
            Aws::GameLift::Server::Model::AttributeValue::ConstructStringDoubleMap();
                    for (auto sdm : value.m_SDM) {
                        attribute.AddStringAndDouble(TCHAR_TO_UTF8(*sdm.Key), sdm.Value);
                    }
            }
        }
    }
    #endif
}

```

```
        };
        break;
    }
    sdkPlayer.WithPlayerAttribute((TCHAR_TO_UTF8(*attributeEntry.Key)), attribute);
}
sdkRequest.AddPlayer(sdkPlayer);
}
auto outcome = Aws::GameLift::Server::StartMatchBackfill(sdkRequest);
if (outcome.IsSuccess()) {
    return FGameLiftStringOutcome(outcome.GetResult().GetTicketId());
}
else {
    return FGameLiftStringOutcome(FGameLiftError(outcome.GetError()));
}
#else
return FGameLiftStringOutcome("");
#endif
}
```

StopMatchBackfill()

Annule une demande de remplacement de match active. Pour plus d'informations, voir la [fonction de FlexMatch remblayage](#).

Syntaxe

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest
&stopBackfillRequest);
```

Paramètres

[FStopMatchBackfillRequest](#)

Un StopMatchBackfillRequest objet identifiant le ticket de matchmaking à annuler :

- L'identifiant du ticket attribué à la demande de remblayage.
- L'entremetteur à qui la demande de remblayage a été envoyée.
- La session de jeu associée à la demande de remplacement.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```

FGameLiftGenericOutcome FGameLiftServerSDKModule::StopMatchBackfill(const
    FStopMatchBackfillRequest& request)
{
    #if WITH_GAMELIFT
        Aws::GameLift::Server::Model::StopMatchBackfillRequest sdkRequest;
        sdkRequest.SetTicketId(TCHAR_TO_UTF8(*request.m_ticketId));
        sdkRequest.SetGameSessionArn(TCHAR_TO_UTF8(*request.m_gameSessionArn));

        sdkRequest.SetMatchmakingConfigurationArn(TCHAR_TO_UTF8(*request.m_matchmakingConfigurationArn));
        auto outcome = Aws::GameLift::Server::StopMatchBackfill(sdkRequest);
        if (outcome.IsSuccess()) {
            return FGameLiftGenericOutcome(nullptr);
        }
        else {
            return FGameLiftGenericOutcome(FGameLiftError(outcome.GetError()));
        }
    #else
        return FGameLiftGenericOutcome(nullptr);
    #endif
}

```

GetComputeCertificate()

Récupère le chemin d'accès au certificat TLS utilisé pour chiffrer la connexion réseau entre votre ressource de calcul Amazon GameLift Servers Anywhere et. Amazon GameLift Servers Vous pouvez utiliser le chemin du certificat lorsque vous enregistrez votre appareil informatique dans un parc Amazon GameLift Servers Anywhere. Pour plus d'informations, voir, [RegisterCompute](#).

Syntaxe

```

FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()

```

Valeur renvoyée

Renvoie un `GetComputeCertificateResponse` objet contenant les éléments suivants :

- **CertificatePath**: chemin d'accès au certificat TLS sur votre ressource de calcul.
- **HostName**: nom d'hôte de votre ressource de calcul.

exemple

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
{
    #if WITH_GAMELIFT
        auto outcome = Aws::GameLift::Server::GetComputeCertificate();
        if (outcome.IsSuccess()) {
            auto& outres = outcome.GetResult();
            FGameLiftGetComputeCertificateResult result;
            result.m_certificate_path = UTF8_T0_TCHAR(outres.GetCertificatePath());
            result.m_computeName = UTF8_T0_TCHAR(outres.GetComputeName());
            return FGameLiftGetComputeCertificateOutcome(result);
        }
        else {
            return FGameLiftGetComputeCertificateOutcome(FGameLiftError(outcome.GetError()));
        }
    #else
        return FGameLiftGetComputeCertificateOutcome(FGameLiftGetComputeCertificateResult());
    #endif
}
```

GetFleetRoleCredentials()

Récupère les informations d'identification du rôle IAM qui autorisent l'interaction avec Amazon GameLift Servers d'autres utilisateurs. Services AWS Pour de plus amples informations, veuillez consulter [Communiquez avec les autres AWS ressources de vos flottes](#).

Syntaxe

```
FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)
```

Paramètres

[FGameLiftGetFleetRoleCredentialsRequest](#)

Valeur renvoyée

Renvoie un objet [the section called “FGameLiftGetFleetRoleCredentialsOutcome”](#).

exemple

```

FGameLiftGetFleetRoleCredentialsOutcome
FGameLiftServerSDKModule::GetFleetRoleCredentials(const
FGameLiftGetFleetRoleCredentialsRequest &request)
{
    #if WITH_GAMELIFT
    Aws::GameLift::Server::Model::GetFleetRoleCredentialsRequest sdkRequest;
    sdkRequest.SetRoleArn(TCHAR_TO_UTF8(*request.m_roleArn));
    sdkRequest.SetRoleSessionName(TCHAR_TO_UTF8(*request.m_roleSessionName));

    auto outcome = Aws::GameLift::Server::GetFleetRoleCredentials(sdkRequest);

    if (outcome.IsSuccess()) {
        auto& outres = outcome.GetResult();
        FGameLiftGetFleetRoleCredentialsResult result;
        result.m_assumedUserRoleArn = UTF8_TO_TCHAR(outres.GetAssumedUserRoleArn());
        result.m_assumedRoleId = UTF8_TO_TCHAR(outres.GetAssumedRoleId());
        result.m_accessKeyId = UTF8_TO_TCHAR(outres.GetAccessKeyId());
        result.m_secretAccessKey = UTF8_TO_TCHAR(outres.GetSecretAccessKey());
        result.m_sessionToken = UTF8_TO_TCHAR(outres.GetSessionToken());
        result.m_expiration = FDateTime::FromUnixTimestamp(outres.GetExpiration());
        return FGameLiftGetFleetRoleCredentialsOutcome(result);
    }
    else {
        return FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftError(outcome.GetError()));
    }
    #else
    return
    FGameLiftGetFleetRoleCredentialsOutcome(FGameLiftGetFleetRoleCredentialsResult());
    #endif
}

```

Détruire ()

Libère de la mémoire le SDK du serveur de Amazon GameLift Servers jeu. Il est recommandé d'appeler cette méthode après `ProcessEnding()` et avant de terminer le processus. Si vous utilisez une flotte Anywhere et que vous n'interrompez pas les processus du serveur après chaque session de jeu, appelez `Destroy()` puis `InitSDK()` réinitialisez avant de signaler Amazon GameLift Servers que le processus est prêt à héberger une session de jeu avec `ProcessReady()`

Syntaxe

```
FGameLiftGenericOutcome Destroy();
```

Paramètres

Il n'y a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// First call ProcessEnding()
FGameLiftGenericOutcome processEndingOutcome = GameLiftSdkModule->ProcessEnding();

// Then call Destroy() to free the SDK from memory
FGameLiftGenericOutcome destroyOutcome = GameLiftSdkModule->Destroy();

// Exit the process with success or failure
if (processEndingOutcome.IsSuccess() && destroyOutcome.IsSuccess())
{
    UE_LOG(GameServerLog, Log, TEXT("Server process ending successfully"));
}
else {
    if (!processEndingOutcome.IsSuccess()) {
        const FGameLiftError& error = processEndingOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("ProcessEnding() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
    if (!destroyOutcome.IsSuccess()) {
        const FGameLiftError& error = destroyOutcome.GetError();
        UE_LOG(GameServerLog, Error, TEXT("Destroy() failed. Error: %s"),
            error.m_errorMessage.IsEmpty() ? TEXT("Unknown error") :
            *error.m_errorMessage);
    }
}
```

SDK de serveur pour Amazon GameLift Servers version 4 et antérieures

Cette référence documente le SDK du serveur pour Amazon GameLift Servers, versions 4.x et antérieures. Le SDK du serveur fournit les fonctionnalités de base que vos serveurs de jeu utilisent pour communiquer avec Amazon GameLift Servers service. Par exemple, votre serveur de jeu reçoit des instructions du service pour démarrer et arrêter des sessions de jeu et fournit régulièrement des mises à jour de l'état des sessions de jeu au service. Intégrez vos serveurs de jeu au SDK du serveur avant de les déployer pour l'hébergement.

Utilisez cette référence au SDK de serveur pour intégrer vos serveurs de jeux multijoueurs personnalisés à des fins d'hébergement avec Amazon GameLift Servers. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

La dernière version majeure du SDK du serveur pour Amazon GameLift Servers est 5.x. Les fonctionnalités d'hébergement suivantes nécessitent des mises à jour vers la version 5.x :

- Amazon GameLift Servers N'importe où
- Amazon GameLift Servers conteneurs gérés
- Amazon GameLift Servers plugin pour Unreal Engine et Unity

Rubriques

- [SDK du serveur C++ pour Amazon GameLift Servers 4.x -- Actions](#)
- [SDK du serveur C# pour Amazon GameLift Servers 4.x -- Actions](#)
- [SDK de serveur \(Unreal\) pour Amazon GameLift Servers -- Mesures](#)

SDK du serveur C++ pour Amazon GameLift Servers 4.x -- Actions

Utilisez la référence du SDK du serveur pour intégrer votre jeu multijoueur à des fins d'Amazon GameLift Servers hébergement. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

 Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour obtenir la dernière version, consultez [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#).

SDK du serveur C++ pour Amazon GameLift Servers 4.x -- Types de données

Utilisez la référence du SDK du serveur pour intégrer votre jeu multijoueur à des fins d'hébergement avec Amazon GameLift Servers. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

 Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour la dernière version, voir [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Types de données](#).

Cette API est définie dans `GameLiftServerAPI.h`, `LogParameters.h` et `ProcessParameters.h`.

[SDK du serveur C++ pour Amazon GameLift Servers 4.x -- Actions](#)

`DescribePlayerSessionsRequest`

Ce type de données est utilisé pour spécifier les sessions de joueur à récupérer. Vous pouvez l'utiliser comme suit :

- Fournissez un `PlayerSessionId` pour demander une session de joueur spécifique.
- Indiquez un `GameSessionId` pour demander toutes les sessions des joueurs dans la session de jeu spécifiée.
- Fournissez un `PlayerId` pour demander toutes les sessions de joueur pour le joueur spécifié.

Pour les volumes importants de sessions de joueur, utilisez les paramètres de pagination pour récupérer les résultats en tant que blocs séquentiels.

Table des matières

GameSessionId

Identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'ID de session de jeu est le suivant : `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. La valeur de `<ID string>` peut être une chaîne d'ID personnalisée ou (si spécifiée lors de la création de la session de jeu) une chaîne générée automatiquement.

Type : chaîne

Obligatoire : non

Limite

Nombre maximum de résultats à renvoyer. Utilisez ce paramètre avec `NextToken` pour obtenir des résultats sous la forme d'un ensemble de pages séquentielles. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : entier

Obligatoire : non

NextToken

Jeton indiquant le début de la prochaine page séquentielle de résultats. Utilisez le jeton qui est renvoyé par un appel précédent à cette action. Pour spécifier le début de l'ensemble de résultats, ne spécifiez aucune valeur. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : chaîne

Obligatoire : non

PlayerId

Identifiant unique pour un joueur. IDs Les joueurs sont définis par le développeur. Consultez [Générer un joueur IDs](#).

Type : chaîne

Obligatoire : non

PlayerSessionId

Identifiant unique d'une session de joueur.

Type : chaîne

Obligatoire : non

PlayerSessionStatusFilter

État de session de joueur pour filtrer les résultats. Les états de session de joueur possibles sont les suivants :

- RESERVED - La demande de session de joueur a été reçue, mais le joueur ne s'est pas encore connecté au processus serveur et/ou n'a pas encore été validé.
- ACTIVE - Le joueur a été validé par le processus serveur et est actuellement connecté.
- COMPLETED - La connexion du joueur a été abandonnée.
- TIMEDOUT - Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté et/ou n'a pas été validé avant l'expiration du délai (60 secondes).

Type : chaîne

Obligatoire : non

LogParameters

Ce type de données est utilisé pour identifier les fichiers générés au cours d'une session de jeu que vous souhaitez Amazon GameLift Servers à télécharger et à stocker une fois la session de jeu terminée. Ces informations sont communiquées au Amazon GameLift Servers service lors d'un [ProcessReady\(\)](#) appel.

Table des matières

logPaths

Chemins de répertoire vers les fichiers journaux du serveur de jeu que vous souhaitez Amazon GameLift Servers à stocker pour un accès futur. Ces fichiers sont générés au cours de chaque session de jeu. Les chemins et noms des fichiers sont définis dans votre serveur de jeux et stockés dans le répertoire racine de génération de jeu. Les chemins du journal doivent être absolus. Par exemple, si la version de génération de votre jeu stocke les journaux de session de jeu suivant un chemin tel que `MyGame\sessionlogs\`, le chemin d'accès aux journaux est `c:\game\MyGame\sessionLogs` (sur une instance Windows) ou `/local/game/MyGame/sessionLogs` (sur une instance Linux).

Type : `std::vector<std::string>`

Obligatoire : non

ProcessParameters

Ce type de données contient le jeu de paramètres envoyé au Amazon GameLift Servers service lors d'un [ProcessReady\(\)](#) appel.

Table des matières

port

Numéro de port sur lequel le processus serveur écoute les nouvelles connexions de joueur. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur.

Type : entier

Obligatoire : oui

logParameters

Objet comportant une liste de chemins de répertoires vers les fichiers journaux de sessions de jeu.

Type : Aws GameLift : ::Server : [LogParameters](#)

Obligatoire : non

onStartGameSession

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à la demande du client [CreateGameSession](#). La fonction de rappel transmet un [GameSession](#) objet (défini dans le Amazon GameLift Servers Référence de l'API de service).

Type : const std::function<void(Aws::GameLift::Model::GameSession)>
onStartGameSession

Obligatoire : oui

onProcessTerminate

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attend cinq minutes que le processus du serveur s'arrête et réponde par un [ProcessEnding\(\)](#) appel. Si aucune réponse n'est reçue, le processus serveur est arrêté.

Type : `std::function<void()> onProcessTerminate`

Obligatoire : non

onHealthCheck

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour demander un rapport d'état de santé au processus du serveur. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes. Après avoir appelé cette fonction Amazon GameLift Servers attend 60 secondes pour obtenir une réponse, et si aucune réponse n'est reçue, enregistre le processus du serveur comme défaillant.

Type : `std::function<bool()> onHealthCheck`

Obligatoire : non

onUpdateGameSession

Nom de la fonction de rappel que Amazon GameLift Servers le service invoque pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de [remplissage de correspondance](#) a été traitée afin de fournir des données de matchmaking mises à jour. Il transmet un [GameSession](#) objet, une mise à jour de statut (`updateReason`) et l'identifiant du ticket de remplacement des matchs.

Type :

`std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)>
onUpdateGameSession`

Obligatoire : non

StartMatchBackfillRequest

Ce type de données est utilisé pour envoyer une requête de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StartMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique. L'action d'API [GetGameSessionId\(\)](#) renvoie l'identifiant au format ARN.

Type : String

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique, sous la forme d'un ARN, pour le matchmaker à utiliser pour cette requête. Pour trouver le matchmaker qui a été utilisé pour créer la session de jeu d'origine, recherchez dans l'objet de session de jeu, dans la propriété de données de matchmaker. En savoir plus sur les données du système de jumelage dans [Word grâce aux données du système de jumelage](#).

Type : String

Obligatoire : oui

Joueurs

Ensemble de données représentant tous les joueurs qui sont actuellement dans la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Consultez Amazon GameLift Servers Guide de référence de l'API pour une description du format d'objet du lecteur. Pour trouver les attributs des joueurs et les affectations des équipes, recherchez dans l'objet de la session de jeu, dans la propriété des données du système de matchmaking. IDs Si une latence est utilisée par le matchmaker, collectez la latence mise à jour pour la région actuelle et incluez-la dans les données de chaque joueur.

Type : std::vectorhttps://docs.aws.amazon.com/gamelift/latest/apireference/API_Player.html
<player>

Obligatoire : oui

TicketId

Identifiant unique pour une correspondance ou un ticket de requête de renvoi de correspondance. Si aucune valeur n'est fournie ici, Amazon GameLift Servers en générera un sous la forme d'un UUID. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire.

Type : chaîne

Obligatoire : non

StopMatchBackfillRequest

Ce type de données est utilisé pour annuler une demande de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique associé à la requête en cours d'annulation.

Type : String

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique du matchmaker auquel cette requête a été envoyée.

Type : String

Obligatoire : oui

TicketId

Identifiant unique d'un ticket de requête de correspondance à annuler.

Type : String

Obligatoire : oui

[SDK du serveur C++ pour Amazon GameLift Servers 4.x -- Types de données](#)

Rubriques

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)

- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [Détruire \(\)](#)

AcceptPlayerSession()

Indique au Amazon GameLift Servers service qu'un joueur possédant l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide, c'est-à-dire que l'identifiant du joueur a réservé une place de joueur dans la session de jeu. Une fois la validation effectuée, Amazon GameLift Servers fait passer l'emplacement de joueur de l'état RESERVED à ACTIVE.

Syntaxe

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

Paramètres

playerSessionId

ID unique émis par le Amazon GameLift Servers service en réponse à un appel à l'action Amazon GameLift Servers [CreatePlayerSession](#) API du AWS SDK. Le client du jeu fait référence à cet identifiant lors de la connexion au processus serveur.

Type : `std::string`

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre une fonction permettant de gérer une demande de connexion, notamment de valider et de rejeter une session de joueur non valide. IDs

```
void ReceiveConnectingPlayerSessionID (Connection& connection, const std::string&
playerSessionId){
    Aws::GameLift::GenericOutcome connectOutcome =
        Aws::GameLift::Server::AcceptPlayerSession(playerSessionId);
    if(connectOutcome.IsSuccess())
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(connectOutcome.GetError().GetMessage());
    }
}
```

ActivateGameSession()

Informe le service Amazon GameLift Servers que le processus serveur a démarré une session de jeu et est maintenant prêt à recevoir les connexions de joueurs. Cette action doit être appelée dans le cadre de la fonction de rappel `onStartGameSession()`, une fois que l'initialisation de toutes les sessions de jeu est terminée.

Syntaxe

```
GenericOutcome ActivateGameSession();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre l'appel de `ActivateGameSession()` dans le cadre de la fonction de rappel `onStartGameSession()`.

```
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession();
}
```

DescribePlayerSessions()

Récupère les données de session de joueur, y compris les paramètres, les métadonnées de session et les données de joueur. Utilisez cette action pour obtenir des informations pour une seule session de joueur, pour toutes les sessions de joueur d'une session de jeu ou pour toutes les sessions de joueur associées à un seul ID de joueur.

Syntaxe

```
DescribePlayerSessionsOutcome DescribePlayerSessions (
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest
    &describePlayerSessionsRequest);
```

Paramètres

describePlayerSessionsDemande

Objet [DescribePlayerSessionsRequest](#) décrivant les sessions de joueur à récupérer.

Obligatoire : oui

Valeur renvoyée

En cas de réussite, renvoie un objet `DescribePlayerSessionsOutcome` qui contient un ensemble d'objets de session de joueur correspondant aux paramètres de la demande. Les objets de session

du joueur ont une structure identique au type de [PlayerSession](#) données de Amazon GameLift Servers l'API du AWS SDK.

exemple

Cet exemple illustre une demande de toutes les sessions de joueur activement connectées à une session de jeu spécifiée. Si vous omettez `NextToken` et définissez la valeur de `Limit` sur 10, Amazon GameLift Servers renvoie les 10 premiers enregistrements de sessions de joueur qui correspondent à la demande.

```
// Set request parameters
Aws::GameLift::Server::Model::DescribePlayerSessionsRequest request;
request.SetPlayerSessionStatusFilter(Aws::GameLift::Server::Model::PlayerSessionStatusMapper::G
request.SetLimit(10);
request.SetGameSessionId("the game session ID");    // can use GetGameSessionId()

// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::DescribePlayerSessions(request);
```

GetGameSessionId()

Extrait un identifiant unique pour la session de jeu actuellement hébergée par le processus serveur, si ce dernier est actif. L'identifiant est retourné dans un format ARN : `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`.

Pour les processus inactifs qui ne sont pas encore activés lors d'une session de jeu, l'appel renvoie `Success GameSessionId = True` et `= ""` (une chaîne vide).

Syntaxe

```
AwsStringOutcome GetGameSessionId();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet `AwsStringOutcome`. En cas d'échec, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

Récupère l'emplacement du fichier d'un certificat TLS codé PEM associé à la flotte et à ses instances. AWS Certificate Manager génère ce certificat lorsque vous créez une nouvelle flotte avec la configuration du certificat définie sur GENERATED. Utilisez ce certificat pour établir une connexion sécurisée avec un client de jeu et pour chiffrer la communication client/serveur.

Syntaxe

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie un `GetInstanceCertificateOutcome` objet contenant l'emplacement du fichier de certificats TLS et de la chaîne de certificats de la flotte, qui sont stockés sur l'instance. Un fichier de certificat racine, extrait de la chaîne de certificats, est également stocké sur l'instance. En cas d'échec, renvoie un message d'erreur.

Pour plus d'informations sur le certificat et les données de la chaîne de certificats, consultez la section [Éléments de GetCertificate réponse](#) dans la référence de l' AWS Certificate Manager API.

exemple

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =  
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

Renvoie numéro de version actuel du kit SDK en cours d'utilisation.

Syntaxe

```
AwsStringOutcome GetSdkVersion();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet `AwsStringOutcome`. La chaîne renvoyée inclut uniquement le numéro de version (par exemple « 3.1.5 »). En cas d'échec, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

GetTerminationTime()

Renvoie l'heure d'arrêt planifiée pour un processus serveur, si une heure de résiliation est disponible. Un processus serveur exécute cette action après avoir reçu un `onProcessTerminate()` rappel du Amazon GameLift Servers service. Amazon GameLift Servers [peut appeler `onProcessTerminate\(\)` pour les raisons suivantes : \(1\) lorsque le processus du serveur a signalé un mauvais état de santé ou n'a pas répondu](#) Amazon GameLift Servers, (2) lors de la résiliation de l'instance lors d'un événement de réduction, ou (3) lorsqu'une instance est interrompue en raison d'une interruption ponctuelle.

Si le processus a reçu un `onProcessTerminate()` rappel, la valeur renvoyée est l'heure de fin estimée. Si le processus n'a pas reçu de `onProcessTerminate()` rappel, un message d'erreur est renvoyé. En savoir plus sur l'[arrêt d'un processus serveur](#).

Syntaxe

```
AwsLongOutcome GetTerminationTime();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie l'heure de fin sous forme d'`AwsLongOutcome` objet. La valeur est le délai de fin, exprimé en ticks écoulés depuis 0001 00:00:00. Par exemple, la valeur de la date et de l'heure 2020-09-13 12:26:40 -000Z est égale à 637355968000000000 ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Initialise le kit SDK Amazon GameLift Servers. Cette méthode doit être appelée au lancement, avant l'exécution de toute autre initialisation liée à Amazon GameLift Servers.

Syntaxe

```
InitSDKOutcome InitSDK();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie un `InitSdkOutcome` objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

exemple

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =  
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Informe le service Amazon GameLift Servers que le processus serveur est en cours d'arrêt. Cette méthode doit être appelée après toutes les autres tâches de nettoyage, y compris l'arrêt de toutes

les sessions de jeu actives. Cette méthode doit quitter avec un code de sortie 0 ; un code de sortie différent de 0 génère un message d'événement indiquant que le processus ne s'est pas fermé correctement.

Une fois la méthode terminée avec un code de 0, vous pouvez terminer le processus avec un code de sortie réussi. Vous pouvez également quitter le processus avec un code d'erreur. Si vous sortez avec un code d'erreur, l'événement de la flotte indiquera que le processus s'est arrêté anormalement (SERVER_PROCESS_TERMINATED_UNHEALTHY).

Syntaxe

```
GenericOutcome ProcessEnding();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();  
if (outcome.Success)  
    exit(0); // exit with success  
// otherwise, exit with error code  
exit(errorCode);
```

ProcessReady()

Informe le service Amazon GameLift Servers que le processus serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après avoir invoqué [InitSDK\(\)](#) et terminé avec succès les tâches de configuration requises avant que le processus serveur puisse héberger une session de jeu. Cette méthode ne doit être appelée qu'une seule fois par processus.

Cet appel est synchrone. Pour effectuer un appel asynchrone, utilisez [ProcessReadyAsync\(\)](#). Pour plus d'informations, consultez [Initialiser le processus du serveur](#).

Syntaxe

```
GenericOutcome ProcessReady(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Paramètres

processParameters

Objet [ProcessParameters](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel, mises en œuvre dans le code du serveur de jeux, que le service Amazon GameLift Servers appelle pour communiquer avec le processus serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre les implémentations de la fonction d'appel et de rappel [ProcessReady\(\)](#).

```
// Set parameters and call ProcessReady  
std::string serverLog("serverOut.log");           // Example of a log file written by the  
game server  
std::vector<std::string> logPaths;  
logPaths.push_back(serverLog);  
  
int listenPort = 9339;  
  
Aws::GameLift::Server::ProcessParameters processReadyParameter =  
    Aws::GameLift::Server::ProcessParameters(  
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),  
        std::bind(&Server::onProcessTerminate, this),  
        std::bind(&Server::OnHealthCheck, this),  
        std::bind(&Server::OnUpdateGameSession, this),
```

```
listenPort,
    Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);

// Implement callback functions
void Server::onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome =
        Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void Server::onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool Server::onHealthCheck()
{
    bool health;
    // complete health evaluation within 60 seconds and set health
    return health;
}
```

ProcessReadyAsync()

Informe le service Amazon GameLift Servers que le processus serveur est prêt à héberger des sessions de jeu. Cette méthode doit être appelée une fois que le processus serveur est prêt à héberger une session de jeu. Les paramètres spécifient les noms des fonctions de rappel qu'Amazon GameLift Servers doit appeler dans certaines circonstances. Le code du serveur de jeux doit implémenter ces fonctions.

Cet appel est asynchrone. Pour effectuer un appel synchrone, utilisez [ProcessReady\(\)](#). Pour plus d'informations, consultez [Initialiser le processus du serveur](#).

Syntaxe

```
GenericOutcomeCallable ProcessReadyAsync(
```

```
const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Paramètres

processParameters

Objet [ProcessParameters](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel, mises en œuvre dans le code du serveur de jeux, que le service Amazon GameLift Servers appelle pour communiquer avec le processus serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set parameters and call ProcessReady
std::string serverLog("serverOut.log");           // This is an example of a log file
written by the game server
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);

int listenPort = 9339;

Aws::GameLift::Server::ProcessParameters processReadyParameter =
    Aws::GameLift::Server::ProcessParameters(
        std::bind(&Server::onStartGameSession, this, std::placeholders::_1),
        std::bind(&Server::onProcessTerminate, this),
        std::bind(&Server::OnHealthCheck, this),
        std::bind(&Server::OnUpdateGameSession, this),
        listenPort,
        Aws::GameLift::Server::LogParameters(logPaths));

Aws::GameLift::GenericOutcomeCallable outcome =
    Aws::GameLift::Server::ProcessReadyAsync(processReadyParameter);
```

```
// Implement callback functions
void onStartGameSession(Aws::GameLift::Model::GameSession myGameSession)
{
    // game-specific tasks when starting a new game session, such as loading map
    GenericOutcome outcome = Aws::GameLift::Server::ActivateGameSession (maxPlayers);
}

void onProcessTerminate()
{
    // game-specific tasks required to gracefully shut down a game session,
    // such as notifying players, preserving game state data, and other cleanup
    GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
}

bool onHealthCheck()
{
    // perform health evaluation and complete within 60 seconds
    return health;
}
```

RemovePlayerSession()

Informe le service Amazon GameLift Servers qu'un joueur avec l'ID de session de joueur spécifié s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers fait passer l'emplacement de joueur à l'état disponible, ce qui lui permet d'être attribué à un nouveau joueur.

Syntaxe

```
GenericOutcome RemovePlayerSession(
    const std::string& playerSessionId);
```

Paramètres

playerSessionId

ID unique émis par le Amazon GameLift Servers service en réponse à un appel à l'action Amazon GameLift Servers [CreatePlayerSession](#) API du AWS SDK. Le client du jeu fait référence à cet identifiant lors de la connexion au processus serveur.

Type : `std::string`

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Voir également l'action du AWS SDK [StartMatchBackfill\(\)](#). Avec cette action, les requêtes de renvoi de correspondance peuvent être initiées par processus de serveur de jeu qui héberge la session de jeu. En savoir plus sur la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés avec succès, le Amazon GameLift Servers service fournit des données de matchmaking mises à jour en invoquant la fonction de rappel.

OnUpdateGameSession()

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
StartMatchBackfillOutcome StartMatchBackfill (  
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest  
    &startBackfillRequest);
```

Paramètres

StartMatchBackfillRequest

Objet [StartMatchBackfillRequest](#) qui communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Cette information est facultative ; si aucun ID n'est fourni, Amazon GameLift Servers en génère automatiquement un.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur peut être acquise à partir des données matchmaker de la session de jeu.
- ID de la session de jeu en cours de renvoi.

- Données de correspondance disponibles pour les joueurs actuels de la session de jeu.

Obligatoire : oui

Valeur renvoyée

Renvoie un `StartMatchBackfillOutcome` objet avec le ticket de remplacement du match ou un échec avec un message d'erreur. L'état du ticket peut être suivi à l'aide de l'action du AWS SDK [DescribeMatchmaking\(\)](#).

exemple

```
// Build a backfill request
std::vector<Player> players;
Aws::GameLift::Server::Model::StartMatchBackfillRequest startBackfillRequest;
startBackfillRequest.SetTicketId("a ticket ID");
    //optional, autogenerated if not provided
startBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration
    ARN"); //from the game session matchmaker data
startBackfillRequest.SetGameSessionArn("the game session ARN");
    // can use GetGameSessionId()
startBackfillRequest.SetPlayers(players);
    //from the game session matchmaker data

// Send backfill request
Aws::GameLift::StartMatchBackfillOutcome backfillOutcome =
    Aws::GameLift::Server::StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void Server::OnUpdateGameSession(Aws::GameLift::Server::Model::GameSession gameSession,
    Aws::GameLift::Server::Model::UpdateReason updateReason, std::string backfillTicketId)
{
    // handle status messages
    // perform game-specific tasks to prep for newly matched players
}
```

StopMatchBackfill()

Annule une requête de renvoi de correspondance active qui a été créée avec [StartMatchBackfill\(\)](#). Voir également l'action du AWS SDK [StopMatchmaking\(\)](#). En savoir plus sur la [fonction de FlexMatch remblayage](#).

Syntaxe

```
GenericOutcome StopMatchBackfill (  
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);
```

Paramètres

StopMatchBackfillRequest

Objet [StopMatchBackfillRequest](#) qui identifie le ticket de correspondance à annuler :

- Identifiant de ticket attribué à la requête de renvoi en cours d'annulation
- matchmaker auquel a été envoyée la requête
- session de jeu associée à la requête de renvoi

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set backfill stop request parameters  
  
Aws::GameLift::Server::Model::StopMatchBackfillRequest stopBackfillRequest;  
stopBackfillRequest.SetTicketId("the ticket ID");  
stopBackfillRequest.SetGameSessionArn("the game session ARN");  
    // can use GetGameSessionId()  
stopBackfillRequest.SetMatchmakingConfigurationArn("the matchmaker configuration ARN");  
    // from the game session matchmaker data  
  
Aws::GameLift::GenericOutcome stopBackfillOutcome =  
    Aws::GameLift::Server::StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Cette méthode est obsolète depuis la version 4.0.1. Au lieu de cela, le processus du serveur doit appeler [ProcessEnding\(\)](#) après la fin d'une session de jeu.

Indique au Amazon GameLift Servers service que le processus du serveur a mis fin à la session de jeu en cours. Cette action est appelée lorsque le processus du serveur restera actif et prêt à héberger

une nouvelle session de jeu. Il ne doit être appelé qu'une fois la procédure de fin de session de jeu terminée, car il indique Amazon GameLift Servers que le processus du serveur est immédiatement disponible pour héberger une nouvelle session de jeu.

Cette action n'est pas appelée si le processus du serveur doit être arrêté après l'arrêt de la session de jeu. Appelez plutôt [ProcessEnding\(\)](#) pour signaler que la session de jeu et le processus du serveur se terminent.

Syntaxe

```
GenericOutcome TerminateGameSession();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur. Voir également l'action du AWS SDK [UpdateGameSession\(\)](#).

Syntaxe

```
GenericOutcome UpdatePlayerSessionCreationPolicy(  
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

Paramètres

newPlayerSessionPolitique

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs.

Type : `Aws : GameLift : :Model : : PlayerSessionCreationPolicy` enum. Les valeurs valides sont les suivantes :

- `ACCEPT_ALL` — Accepte toutes les sessions de nouveaux joueurs.

- **DENY_ALL** — Refuse toutes les sessions de nouveaux joueurs.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Détruire ()

Nettoie la mémoire allouée par `initSDK ()` lors de l'initialisation du serveur de jeu. Utilisez cette méthode après avoir terminé un processus de serveur de jeu pour éviter de gaspiller la mémoire du serveur.

Syntaxe

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Paramètres

Il n'y a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple nettoie la mémoire allouée par `InitSDK` une fois le processus du serveur de jeu terminé.

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {
```

```
Aws::GameLift::Server::Destroy();  
exit(0);  
}
```

SDK du serveur C# pour Amazon GameLift Servers 4.x -- Actions

Utilisez la référence du SDK du serveur pour intégrer votre jeu multijoueur à des fins d'Amazon GameLift Servers hébergement. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour obtenir la dernière version, consultez [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#).

SDK du serveur C# pour Amazon GameLift Servers 4.x -- Types de données

Utilisez la référence du SDK du serveur pour intégrer votre jeu multijoueur à des fins d'hébergement avec Amazon GameLift Servers. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour la dernière version, voir [SDK 5.x du serveur C# pour Amazon GameLift Servers -- Types de données](#).

[SDK du serveur C# pour Amazon GameLift Servers 4.x -- Actions](#)

Rubriques

- [LogParameters](#)
- [DescribePlayerSessionsRequest](#)
- [ProcessParameters](#)
- [StopMatchBackfillRequest](#)

LogParameters

Ce type de données est utilisé pour identifier les fichiers générés au cours d'une session de jeu que vous souhaitez Amazon GameLift Servers à télécharger et à stocker une fois la session de jeu terminée. Ces informations sont communiquées au Amazon GameLift Servers service lors d'un [ProcessReady\(\)](#) appel.

Table des matières

logPaths

Liste des chemins de répertoire vers les fichiers journaux du serveur de jeu que vous souhaitez Amazon GameLift Servers à stocker pour un accès futur. Ces fichiers sont générés par un processus serveur pendant chaque session de jeu ; les noms et les chemins des fichiers sont définis dans votre serveur de jeux et stockés dans le répertoire racine de la version de génération de jeu. Les chemins du journal doivent être absolus. Par exemple, si la version de génération de votre jeu stocke les journaux de session de jeu suivant un chemin tel que MyGame\sessionlogs \, le chemin d'accès aux journaux est c:\game\MyGame\sessionLogs (sur une instance Windows) ou /local/game/MyGame/sessionLogs (sur une instance Linux).

Type : Liste<String>

Obligatoire : non

DescribePlayerSessionsRequest

Ce type de données est utilisé pour spécifier les sessions de joueur à récupérer. Il peut être utilisé de plusieurs manières : (1) fournir une session PlayerSessionId pour demander une session de joueur spécifique ; (2) fournir une GameSessionId pour demander toutes les sessions de joueur dans la session de jeu spécifiée ; ou (3) fournir une PlayerId pour demander toutes les sessions de joueur pour le joueur spécifié. Pour les volumes importants de sessions de joueur, utilisez les paramètres de pagination pour récupérer les résultats en tant que pages séquentielles.

Table des matières

GameSessionId

Identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'ID de session de jeu est le suivant : arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>. La

valeur de <ID string> peut être une chaîne d'ID personnalisée (si spécifiée lors de la création de la session de jeu) ou une chaîne générée automatiquement.

Type : chaîne

Obligatoire : non

Limite

Nombre maximum de résultats à renvoyer. Utilisez ce paramètre avec NextToken pour obtenir des résultats sous la forme d'un ensemble de pages séquentielles. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : entier

Obligatoire : non

NextToken

Jeton indiquant le début de la prochaine page séquentielle de résultats. Utilisez le jeton qui est renvoyé par un appel précédent à cette action. Pour spécifier le début de l'ensemble de résultats, ne spécifiez aucune valeur. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : chaîne

Obligatoire : non

PlayerId

Identifiant unique pour un joueur. IDs Les joueurs sont définis par le développeur. Consultez [Générer un joueur IDs](#).

Type : chaîne

Obligatoire : non

PlayerSessionId

Identifiant unique d'une session de joueur.

Type : chaîne

Obligatoire : non

PlayerSessionStatusFilter

État de session de joueur pour filtrer les résultats. Les états de session de joueur possibles sont les suivants :

- RESERVED - La demande de session de joueur a été reçue, mais le joueur ne s'est pas encore connecté au processus serveur et/ou n'a pas encore été validé.
- ACTIVE - Le joueur a été validé par le processus serveur et est actuellement connecté.
- COMPLETED - La connexion du joueur a été abandonnée.
- TIMEDOUT - Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté et/ou n'a pas été validé avant l'expiration du délai (60 secondes).

Type : chaîne

Obligatoire : non

ProcessParameters

Ce type de données contient le jeu de paramètres envoyé au Amazon GameLift Servers service lors d'un [ProcessReady\(\)](#) appel.

Table des matières

port

Numéro de port sur lequel le processus serveur écoute les nouvelles connexions de joueur. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur.

Type : entier

Obligatoire : oui

logParameters

Objet comportant une liste de chemins de répertoires vers les fichiers journaux de sessions de jeu.

Type : `Aws::GameLift::Server::LogParameters`

Obligatoire : oui

onStartGameSession

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à la demande du client [CreateGameSession](#). La fonction de rappel prend un [GameSession](#) objet (défini dans le Amazon GameLift Servers Référence de l'API de service).

Type : void OnStartGameSessionDelegate(GameSession gameSession)

Obligatoire : oui

onProcessTerminate

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attend cinq minutes que le processus serveur s'arrête et répond par un [ProcessEnding\(\)](#) appel avant d'arrêter le processus serveur.

Type : void OnProcessTerminateDelegate()

Obligatoire : oui

onHealthCheck

Nom de la fonction de rappel que Amazon GameLift Servers le service est invoqué pour demander un rapport d'état de santé au processus du serveur. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes. Après avoir appelé cette fonction Amazon GameLift Servers attend 60 secondes pour obtenir une réponse, et si aucune réponse n'est reçue, enregistre le processus du serveur comme défaillant.

Type : bool OnHealthCheckDelegate()

Obligatoire : oui

onUpdateGameSession

Nom de la fonction de rappel que Amazon GameLift Servers le service invoque pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de [remplissage de correspondance](#) a été traitée afin de fournir des données de matchmaking mises à jour. Il transmet un [GameSession](#) objet, une mise à jour de statut (updateReason) et l'identifiant du ticket de remplacement des matchs.

Type : void OnUpdateGameSessionDelegate (UpdateGameSession
updateGameSession)

Obligatoire : non

StartMatchBackfillRequest

Ce type de données est utilisé pour envoyer une requête de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StartMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique. La méthode SDK [GetGameSessionId\(\)](#) renvoie l'identifiant au format ARN.

Type : String

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique, sous la forme d'un ARN, pour le matchmaker à utiliser pour cette requête. Pour trouver le matchmaker qui a été utilisé pour créer la session de jeu d'origine, recherchez dans l'objet de session de jeu, dans la propriété de données de matchmaker. Pour en savoir plus sur les données du système de jumelage, consultez la section [Travailler avec les données du système de jumelage](#).

Type : String

Obligatoire : oui

Joueurs

Ensemble de données représentant tous les joueurs qui sont actuellement dans la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Consultez Amazon GameLift Servers Guide de référence de l'API pour une description du format d'objet Player. Pour trouver les attributs des joueurs et les affectations des équipes, recherchez dans l'objet de la session de jeu, dans la propriété des données du système de matchmaking. IDs Si une latence est utilisée par le

matchmaker, collectez la latence mise à jour pour la région actuelle et incluez-la dans les données de chaque joueur.

Type : [Player](#)[]

Obligatoire : oui

TicketId

Identifiant unique pour une correspondance ou un ticket de requête de renvoi de correspondance. Si aucune valeur n'est fournie ici, Amazon GameLift Servers en générera un sous la forme d'un UUID. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire.

Type : chaîne

Obligatoire : non

StopMatchBackfillRequest

Ce type de données est utilisé pour annuler une demande de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique associé à la requête en cours d'annulation.

Type : String

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique du matchmaker auquel cette requête a été envoyée.

Type : String

Obligatoire : oui

TicketId

Identifiant unique d'un ticket de requête de correspondance à annuler.

Type : String

Obligatoire : oui

[SDK du serveur C# pour Amazon GameLift Servers 4.x -- Types de données](#)

Rubriques

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Indique au Amazon GameLift Servers service qu'un joueur possédant l'identifiant de session de joueur spécifié s'est connecté au processus du serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide, c'est-à-dire que l'identifiant du joueur a réservé une place de joueur dans la session de jeu. Une fois la validation effectuée, Amazon GameLift Servers fait passer l'emplacement de joueur de l'état RESERVED à ACTIVE.

Syntaxe

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur. Un identifiant de session de joueur est spécifié dans un `PlayerSession` objet, qui est renvoyé en réponse à un appel du client aux actions de l'GameLift API [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#), ou [DescribePlayerSessions](#).

Type : String

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre une fonction permettant de gérer une demande de connexion, notamment de valider et de rejeter une session de joueur non valide. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerSessionId){
    var acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerSessionId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```

ActivateGameSession()

Informe le service Amazon GameLift Servers que le processus serveur a activé une session de jeu et est maintenant prêt à recevoir des connexions de joueur. Cette action doit être appelée dans le cadre

de la fonction de rappel `onStartGameSession()`, une fois que l'initialisation de toutes les sessions de jeu est terminée.

Syntaxe

```
GenericOutcome ActivateGameSession()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre l'appel de `ActivateGameSession()` dans le cadre de la fonction déléguée `onStartGameSession()`.

```
void OnStartGameSession(GameSession gameSession)
{
    // game-specific tasks when starting a new game session, such as loading map

    // When ready to receive players
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();
}
```

DescribePlayerSessions()

Récupère les données de session de joueur, y compris les paramètres, les métadonnées de session et les données de joueur. Utilisez cette action pour obtenir des informations pour une seule session de joueur, pour toutes les sessions de joueur d'une session de jeu ou pour toutes les sessions de joueur associées à un seul ID de joueur.

Syntaxe

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest
describePlayerSessionsRequest)
```

Paramètres

describePlayerSessionsDemande

Objet [DescribePlayerSessionsRequest](#) décrivant les sessions de joueur à récupérer.

Obligatoire : oui

Valeur renvoyée

En cas de réussite, renvoie un objet `DescribePlayerSessionsOutcome` qui contient un ensemble d'objets de session de joueur correspondant aux paramètres de la demande. Les objets de session du joueur ont une structure identique au type de [PlayerSession](#) données de Amazon GameLift Servers l'API du AWS SDK.

exemple

Cet exemple illustre une demande de toutes les sessions de joueur activement connectées à une session de jeu spécifiée. En omettant `NextToken` en définissant la valeur limite sur 10, les 10 premiers enregistrements de sessions de joueurs correspondant à la demande Amazon GameLift Servers seront renvoyés.

```
// Set request parameters
var describePlayerSessionsRequest = new
    Aws.GameLift.Server.Model.DescribePlayerSessionsRequest()
{
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result,    //gets the ID for
    the current game session
    Limit = 10,
    PlayerSessionStatusFilter =
    PlayerSessionStatusMapper.GetNameForPlayerSessionStatus(PlayerSessionStatus.ACTIVE)
};
// Call DescribePlayerSessions
Aws::GameLift::DescribePlayerSessionsOutcome playerSessionsOutcome =
    Aws::GameLift::Server::Model::DescribePlayerSessions(describePlayerSessionRequest);
```

GetGameSessionId()

Extrait l'ID de la session de jeu actuellement hébergée par le processus serveur, si ce dernier est actif.

Pour les processus inactifs qui ne sont pas encore activés lors d'une session de jeu, l'appel renvoie `Success GameSessionId = True` et `= ""` (une chaîne vide).

Syntaxe

```
AwsStringOutcome GetGameSessionId()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet `AwsStringOutcome`. En cas d'échec, renvoie un message d'erreur.

exemple

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

Récupère l'emplacement du fichier d'un certificat TLS codé PEM associé à la flotte et à ses instances. AWS Certificate Manager génère ce certificat lorsque vous créez une nouvelle flotte avec la configuration du certificat définie sur `GENERATED`. Utilisez ce certificat pour établir une connexion sécurisée avec un client de jeu et pour chiffrer la communication client/serveur.

Syntaxe

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie un `GetInstanceCertificateOutcome` objet contenant l'emplacement du fichier de certificats TLS et de la chaîne de certificats de la flotte, qui sont stockés sur l'instance.

Un fichier de certificat racine, extrait de la chaîne de certificats, est également stocké sur l'instance. En cas d'échec, renvoie un message d'erreur.

Pour plus d'informations sur le certificat et les données de la chaîne de certificats, consultez la section [Éléments de GetCertificate réponse](#) dans la référence de l' AWS Certificate Manager API.

exemple

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
AwsStringOutcome GetSdkVersion()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet `AwsStringOutcome`. La chaîne renvoyée inclut uniquement le numéro de version (par exemple « 3.1.5 »). En cas d'échec, renvoie un message d'erreur.

exemple

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

Renvoie l'heure d'arrêt planifiée pour un processus serveur, si une heure de résiliation est disponible. Un processus serveur exécute cette action après avoir reçu un `onProcessTerminate()` rappel du Amazon GameLift Servers service. Amazon GameLift Servers [peut appeler onProcessTerminate\(\)](#) pour les raisons suivantes : (1) en cas de mauvais état (le processus

[serveur a signalé l'état du port ou n'a pas répondu](#)) Amazon GameLift Servers, (2) lors de la résiliation de l'instance lors d'un événement de réduction d'échelle, ou (3) lorsqu'une instance est interrompue en raison d'une interruption ponctuelle de l'instance.

Si le processus a reçu un `onProcessTerminate()` rappel, la valeur renvoyée est l'heure de fin estimée. Si le processus n'a pas reçu de `onProcessTerminate()` rappel, un message d'erreur est renvoyé. En savoir plus sur l'[arrêt d'un processus serveur](#).

Syntaxe

```
AwsDateTimeOutcome GetTerminationTime()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie l'heure de fin sous forme d'`AwsDateTimeOutcome` objet. La valeur est le délai de fin, exprimé en ticks écoulés depuis 0001 00:00:00. Par exemple, la valeur de la date et de l'heure 2020-09-13 12:26:40 -000Z est égale à 637355968000000000 ticks. Si aucune heure de résiliation n'est disponible, renvoie un message d'erreur.

exemple

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

InitSDK()

Initialise le kit SDK Amazon GameLift Servers. Cette méthode doit être appelée au lancement, avant l'exécution de toute autre initialisation liée à Amazon GameLift Servers.

Syntaxe

```
InitSDKOutcome InitSDK()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie un `InitSdkOutcome` objet indiquant que le processus serveur est prêt à être appelé [ProcessReady\(\)](#).

exemple

```
var initSDKOutcome = GameLiftServerAPI.InitSDK();
```

ProcessEnding()

Informe le service Amazon GameLift Servers que le processus serveur est en cours d'arrêt. Cette méthode doit être appelée après toutes les autres tâches de nettoyage, y compris l'arrêt de toutes les sessions de jeu actives. Cette méthode doit quitter avec un code de sortie 0 ; un code de sortie différent de 0 génère un message d'événement indiquant que le processus ne s'est pas fermé correctement.

Une fois que la méthode s'est terminée avec un code de 0, vous pouvez terminer le processus avec un code de sortie réussi. Vous pouvez également quitter le processus avec un code d'erreur. Si vous sortez avec un code d'erreur, l'événement de la flotte indiquera que le processus s'est arrêté anormalement (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Syntaxe

```
GenericOutcome ProcessEnding()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();  
if (processReadyOutcome.Success)  
    Environment.Exit(0);
```

```
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Informe le service Amazon GameLift Servers que le processus serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après avoir invoqué [InitSDK\(\)](#) et terminé avec succès les tâches de configuration requises avant que le processus serveur puisse héberger une session de jeu. Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Paramètres

processParameters

Objet [ProcessParameters](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel, mises en œuvre dans le code du serveur de jeux, que le service Amazon GameLift Servers appelle pour communiquer avec le processus serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin vers les fichiers spécifiques à une session de jeu que vous voulez qu'Amazon GameLift Servers capture et stocke.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre les implémentations de l'appel [ProcessReady\(\)](#) et de la fonction déléguée.

```
// Set parameters and call ProcessReady
```

```
var processParams = new ProcessParameters(  
    this.OnGameSession,  
    this.OnProcessTerminate,  
    this.OnHealthCheck,  
    this.OnGameSessionUpdate,  
    port,  
    new LogParameters(new List<string>()           // Examples of log and error files  
        written by the game server  
        {  
            "C:\\game\\logs",  
            "C:\\game\\error"  
        })  
);  
  
var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);  
  
// Implement callback functions  
void OnGameSession(GameSession gameSession)  
{  
    // game-specific tasks when starting a new game session, such as loading map  
    // When ready to receive players  
    var activateGameSessionOutcome = GameLiftServerAPI.ActivateGameSession();  
}  
  
void OnProcessTerminate()  
{  
    // game-specific tasks required to gracefully shut down a game session,  
    // such as notifying players, preserving game state data, and other cleanup  
    var ProcessEndingOutcome = GameLiftServerAPI.ProcessEnding();  
}  
  
bool OnHealthCheck()  
{  
    bool isHealthy;  
    // complete health evaluation within 60 seconds and set health  
    return isHealthy;  
}
```

RemovePlayerSession()

Informe le service Amazon GameLift Servers qu'un joueur avec l'ID de session de joueur spécifié s'est déconnecté du processus serveur. En réponse, Amazon GameLift Servers fait passer l'emplacement de joueur à l'état disponible, ce qui lui permet d'être attribué à un nouveau joueur.

Syntaxe

```
GenericOutcome RemovePlayerSession(String playerId)
```

Paramètres

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur. Un identifiant de session de joueur est spécifié dans un `PlayerSession` objet, qui est renvoyé en réponse à un appel du client aux actions de l'GameLift API [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#), ou [DescribePlayerSessions](#).

Type : String

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Envoie une demande de recherche de nouveaux joueurs pour des emplacements ouverts dans une session de jeu créée avec FlexMatch. Voir également l'action du AWS SDK [StartMatchBackfill\(\)](#). Avec cette action, les requêtes de renvoi de correspondance peuvent être initiées par processus de serveur de jeu qui héberge la session de jeu. En savoir plus sur la [fonction de FlexMatch remblayage](#).

Cette action est asynchrone. Si de nouveaux joueurs sont mis en relation, le service Amazon GameLift Servers fournit les données du matchmaker mises à jour à l'aide de la fonction de renvoi `OnUpdateGameSession()`.

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
startBackfillRequest);
```

Paramètres

StartMatchBackfillRequest

Objet [StartMatchBackfillRequest](#) qui communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Cette information est facultative ; si aucun ID n'est fourni, Amazon GameLift Servers en génère automatiquement un.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur peut être acquise à partir des données matchmaker de la session de jeu.
- ID de la session de jeu en cours de renvoi.
- Données de correspondance disponibles pour les joueurs actuels de la session de jeu.

Obligatoire : oui

Valeur renvoyée

Renvoie un StartMatchBackfillOutcome objet avec l'identifiant du ticket de remplacement correspondant ou un échec avec un message d'erreur.

exemple

```
// Build a backfill request  
var startBackfillRequest = new AWS.GameLift.Server.Model.StartMatchBackfillRequest()  
{  
    TicketId = "a ticket ID", //optional  
    MatchmakingConfigurationArn = "the matchmaker configuration ARN",  
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result, // gets ID for  
current game session  
    //get player data for all currently connected players  
    MatchmakerData matchmakerData =  
        MatchmakerData.FromJson(gameSession.MatchmakerData); // gets matchmaker  
data for current players  
    // get matchmakerData.Players  
    // remove data for players who are no longer connected  
    Players = ListOfPlayersRemainingInTheGame
```

```
};

// Send backfill request
var startBackfillOutcome = GameLiftServerAPI.StartMatchBackfill(startBackfillRequest);

// Implement callback function for backfill
void OnUpdateGameSession(GameSession myGameSession)
{
    // game-specific tasks to prepare for the newly matched players and update
    matchmaker data as needed
}
```

StopMatchBackfill()

Annule une requête de renvoi de correspondance active qui a été créée avec [StartMatchBackfill\(\)](#). Voir également l'action du AWS SDK [StopMatchmaking\(\)](#). En savoir plus sur la [fonction de FlexMatch remblayage](#).

Syntaxe

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Paramètres

StopMatchBackfillRequest

Objet [StopMatchBackfillRequest](#) qui identifie le ticket de correspondance à annuler :

- Identifiant de ticket attribué à la requête de renvoi en cours d'annulation
- matchmaker auquel a été envoyée la requête
- session de jeu associée à la requête de renvoi

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

```
// Set backfill stop request parameters
```

```
var stopBackfillRequest = new AWS.GameLift.Server.Model.StopMatchBackfillRequest()
{
    TicketId = "a ticket ID", //optional, if not provided one is autogenerated
    MatchmakingConfigurationArn = "the matchmaker configuration ARN", //from the game
    session matchmaker data
    GameSessionId = GameLiftServerAPI.GetGameSessionId().Result //gets the ID for
    the current game session
};

var stopBackfillOutcome =
    GameLiftServerAPI.StopMatchBackfillRequest(stopBackfillRequest);
```

TerminateGameSession()

Cette méthode est obsolète depuis la version 4.0.1. Au lieu de cela, le processus du serveur doit appeler [ProcessEnding\(\)](#) après la fin d'une session de jeu.

Indique au Amazon GameLift Servers service que le processus du serveur a mis fin à la session de jeu en cours. Cette action est appelée lorsque le processus du serveur restera actif et prêt à héberger une nouvelle session de jeu. Il ne doit être appelé qu'une fois la procédure de fin de session de jeu terminée, car il indique Amazon GameLift Servers que le processus du serveur est immédiatement disponible pour héberger une nouvelle session de jeu.

Cette action n'est pas appelée si le processus du serveur doit être arrêté après l'arrêt de la session de jeu. Appelez plutôt [ProcessEnding\(\)](#) pour signaler que la session de jeu et le processus du serveur se terminent.

Syntaxe

```
GenericOutcome TerminateGameSession()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple illustre un processus de serveur à la fin d'une session de jeu.

```
// game-specific tasks required to gracefully shut down a game session,  
// such as notifying players, preserving game state data, and other cleanup  
  
var terminateGameSessionOutcome = GameLiftServerAPI.TerminateGameSession();  
var processReadyOutcome = GameLiftServerAPI.ProcessReady(processParams);
```

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur. (Voir également l'action [UpdateGameSession\(\)](#) dans la référence de l'API de Amazon GameLift Servers service).

Syntaxe

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy  
    playerSessionPolicy)
```

Paramètres

newPlayerSessionPolitique

Valeur de chaîne indiquant si la session de jeu accepte ou non de nouveaux joueurs.

Type : enum [PlayerSessionCreationPolicy](#). Les valeurs valides sont les suivantes :

- ACCEPT_ALL — Accepte toutes les sessions de nouveaux joueurs.
- DENY_ALL — Refuse toutes les sessions de nouveaux joueurs.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Cet exemple définit la stratégie de participation de la session de jeu actuelle de manière à ce que tous les joueurs soient acceptés.

```
var updatePlayerSessionCreationPolicyOutcomex =
```



```
GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

SDK de serveur (Unreal) pour Amazon GameLift Servers -- Mesures

Utilisez le SDK du serveur pour Unreal pour intégrer votre jeu multijoueur afin de l'héberger avec Amazon GameLift Servers. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour la dernière version, voir [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#).

Cette API est définie dans `GameLiftServerSDK.h` et `GameLiftServerSDKModels.h`.

Pour configurer le plug-in Unreal Engine et consulter des exemples de code [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#).

SDK de serveur (Unreal) pour Amazon GameLift Servers -- Types de données

Utilisez la commande Amazon GameLift Servers SDK de serveur pour Unreal Reference pour intégrer votre jeu multijoueur à héberger avec Amazon GameLift Servers. Pour obtenir des conseils sur le processus d'intégration, consultez [Ajouter Amazon GameLift Servers à votre serveur de jeu](#).

Note

Cette référence concerne une version antérieure du SDK du serveur pour Amazon GameLift Servers. Pour la dernière version, voir [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Types de données](#).

Cette API est définie dans `GameLiftServerSDK.h` et `GameLiftServerSDKModels.h`.

Pour configurer le plug-in Unreal Engine et consulter des exemples de code [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#).

[SDK de serveur \(Unreal\) pour Amazon GameLift Servers -- Mesures](#)

Rubriques

- [FDescribePlayerSessionsRequest](#)
- [FProcessParamètres](#)
- [FStartMatchBackfillRequest](#)
- [FStopMatchBackfillRequest](#)

FDescribePlayerSessionsRequest

Ce type de données est utilisé pour spécifier les sessions de joueur à récupérer. Vous pouvez l'utiliser comme suit :

- Fournissez un `PlayerSessionId` pour demander une session de joueur spécifique.
- Indiquez un `GameSessionId` pour demander toutes les sessions des joueurs dans la session de jeu spécifiée.
- Indiquez un `PlayerId` pour demander toutes les sessions de joueur pour le joueur spécifié.

Pour les volumes importants de sessions de joueur, utilisez les paramètres de pagination pour récupérer les résultats en tant que blocs séquentiels.

Table des matières

GameSessionId

Identifiant de session de jeu unique. Utilisez ce paramètre pour demander toutes les sessions de joueur pour la session de jeu spécifiée. Le format de l'ID de session de jeu est le suivant : `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. La valeur de `<ID string>` peut être une chaîne d'ID personnalisée ou (si spécifiée lors de la création de la session de jeu) une chaîne générée automatiquement.

Type : chaîne

Obligatoire : non

Limite

Nombre maximum de résultats à renvoyer. Utilisez ce paramètre avec `NextToken` pour obtenir des résultats sous la forme d'un ensemble de pages séquentielles. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : entier

Obligatoire : non

NextToken

Jeton indiquant le début de la prochaine page séquentielle de résultats. Utilisez le jeton qui est renvoyé par un appel précédent à cette action. Pour spécifier le début de l'ensemble de résultats, ne spécifiez aucune valeur. Si un ID de session de joueur est spécifié, ce paramètre est ignoré.

Type : chaîne

Obligatoire : non

PlayerId

Identifiant unique pour un joueur. IDs Les joueurs sont définis par le développeur. Consultez [Générer un joueur IDs](#).

Type : chaîne

Obligatoire : non

PlayerSessionId

Identifiant unique d'une session de joueur.

Type : chaîne

Obligatoire : non

PlayerSessionStatusFilter

État de session de joueur pour filtrer les résultats. Les états de session de joueur possibles sont les suivants :

- RESERVED - La demande de session de joueur a été reçue, mais le joueur ne s'est pas encore connecté au processus serveur et/ou n'a pas encore été validé.
- ACTIVE - Le joueur a été validé par le processus serveur et est actuellement connecté.
- COMPLETED - La connexion du joueur a été abandonnée.
- TIMEDOUT - Une demande de session de joueur a été reçue, mais le joueur ne s'est pas connecté et/ou n'a pas été validé avant l'expiration du délai (60 secondes).

Type : chaîne

Obligatoire : non

FProcessParamètres

Ce type de données contient le jeu de paramètres envoyé au Amazon GameLift Servers service lors d'un [ProcessReady\(\)](#) appel.

Table des matières

port

Numéro de port sur lequel le processus serveur écoute les nouvelles connexions de joueur. La valeur doit être comprise dans la plage de ports configurée pour toutes les flottes déployant cette version de génération du serveur de jeux. Ce numéro de port est inclus dans les objets de session de jeu et de session de joueur, que les sessions de jeu utilisent pour se connecter à un processus serveur.

Type : entier

Obligatoire : oui

logParameters

Objet comportant une liste de chemins de répertoires vers les fichiers journaux de sessions de jeu.

Type : TArray < FString >

Obligatoire : non

onStartGameSession

Nom de la fonction de rappel que le Amazon GameLift Servers le service est invoqué pour activer une nouvelle session de jeu. Amazon GameLift Servers appelle cette fonction en réponse à la demande du client [CreateGameSession](#). La fonction de rappel prend un [GameSession](#) objet (défini dans le Amazon GameLift Servers Référence de l'API de service).

Type : FOnStartGameSession

Obligatoire : oui

onProcessTerminate

Nom de la fonction de rappel que le Amazon GameLift Servers le service est invoqué pour forcer l'arrêt du processus du serveur. Après avoir appelé cette fonction, Amazon GameLift Servers attend cinq minutes que le processus serveur s'arrête et répond par un [ProcessEnding\(\)](#) appel avant d'arrêter le processus serveur.

Type : FSimple Délégué

Obligatoire : non

onHealthCheck

Nom de la fonction de rappel que le Amazon GameLift Servers le service est invoqué pour demander un rapport d'état de santé au processus du serveur. Amazon GameLift Servers appelle cette fonction toutes les 60 secondes. Après avoir appelé cette fonction Amazon GameLift Servers attend 60 secondes pour obtenir une réponse, et si aucune réponse n'est reçue, enregistre le processus du serveur comme défaillant.

Type : FOnHealthCheck

Obligatoire : non

onUpdateGameSession

Nom de la fonction de rappel que le Amazon GameLift Servers le service invoque pour transmettre un objet de session de jeu mis à jour au processus du serveur. Amazon GameLift Servers appelle cette fonction lorsqu'une demande de [remplissage de correspondance](#) a été traitée afin de fournir des données de matchmaking mises à jour. Il transmet un [GameSession](#) objet, une mise à jour de statut (updateReason) et l'identifiant du ticket de remplacement des matchs.

Type : FOnUpdateGameSession

Obligatoire : non

FStartMatchBackfillRequest

Ce type de données est utilisé pour envoyer une requête de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StartMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique. L'action d'API [GetGameSessionId\(\)](#) renvoie l'identifiant au format ARN.

Type : FString

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique, sous la forme d'un ARN, pour le matchmaker à utiliser pour cette requête. Pour trouver le matchmaker qui a été utilisé pour créer la session de jeu d'origine, recherchez dans l'objet de session de jeu, dans la propriété de données de matchmaker. Pour en savoir plus sur les données du système de jumelage, consultez la section [Travailler avec les données du système de jumelage](#).

Type : FString

Obligatoire : oui

Joueurs

Ensemble de données représentant tous les joueurs qui sont actuellement dans la session de jeu. Le matchmaker utilise ces informations pour rechercher de nouveaux joueurs qui constituent de bonnes correspondances pour les joueurs actuels. Consultez Amazon GameLift Servers Guide de référence de l'API pour une description du format d'objet Player. Pour trouver les attributs des joueurs et les affectations des équipes, recherchez l'objet de la session de jeu, dans la propriété des données du système de matchmaking. IDs Si une latence est utilisée par le matchmaker, collectez la latence mise à jour pour la région actuelle et incluez-la dans les données de chaque joueur.

Type : TArray [< FPlayer >](#)

Obligatoire : oui

TicketId

Identifiant unique pour une correspondance ou un ticket de requête de renvoi de correspondance. Si aucune valeur n'est fournie ici, Amazon GameLift Servers en générera un sous la forme d'un UUID. Utilisez cet identifiant pour suivre l'état du ticket de renvoi de correspondance ou annuler la requête si nécessaire.

Type : FString

Obligatoire : non

FStopMatchBackfillRequest

Ce type de données est utilisé pour annuler une demande de renvoi de correspondance. Les informations sont communiquées au Amazon GameLift Servers service lors d'un [StopMatchBackfill\(\)](#) appel.

Table des matières

GameSessionArn

Identifiant de session de jeu unique associé à la requête en cours d'annulation.

Type : FString

Obligatoire : oui

MatchmakingConfigurationArn

Identifiant unique du matchmaker auquel cette requête a été envoyée.

Type : FString

Obligatoire : oui

TicketId

Identifiant unique d'un ticket de requête de correspondance à annuler.

Type : FString

Obligatoire : oui

[SDK de serveur \(Unreal\) pour Amazon GameLift Servers -- Types de données](#)

Rubriques

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)

- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Informe le Amazon GameLift Servers service qu'un joueur ayant l'identifiant de session de joueur spécifié s'est connecté au serveur et doit être validé. Amazon GameLift Servers vérifie que l'identifiant de session du joueur est valide, c'est-à-dire que l'identifiant du joueur a réservé une place de joueur dans la session de jeu. Une fois validé, Amazon GameLift Servers fait passer le statut de l'emplacement réservé au joueur de RÉSERVÉ à ACTIF.

Syntaxe

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Paramètres

playerSessionId

Identifiant unique émis par le Amazon GameLift Servers service en réponse à un appel au AWS SDK Amazon GameLift Servers Action de l'API [CreatePlayerSession](#). Le client du jeu fait référence à cet identifiant lors de la connexion au processus serveur.

Type : FString

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

ActivateGameSession()

Informe le Amazon GameLift Servers service selon lequel le processus du serveur a activé une session de jeu et est maintenant prêt à recevoir les connexions des joueurs. Cette action doit être appelée dans le cadre de la fonction de rappel `onStartGameSession()`, une fois que l'initialisation de toutes les sessions de jeu est terminée.

Syntaxe

```
FGameLiftGenericOutcome ActivateGameSession()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

DescribePlayerSessions()

Récupère les données de session de joueur, y compris les paramètres, les métadonnées de session et les données de joueur. Utilisez cette action pour obtenir des informations pour une seule session de joueur, pour toutes les sessions de joueur d'une session de jeu ou pour toutes les sessions de joueur associées à un seul ID de joueur.

Syntaxe

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Paramètres

`describePlayerSessionsDemande`

Objet [FDescribePlayerSessionsRequest](#) décrivant les sessions de joueur à récupérer.

Obligatoire : oui

Valeur renvoyée

En cas de réussite, renvoie un objet [FDescribePlayerSessionsRequest](#) qui contient un ensemble d'objets de session de joueur correspondant aux paramètres de la demande. Les objets de session du joueur ont une structure identique à celle du AWS SDK Amazon GameLift Servers Type de [PlayerSession](#) données d'API.

GetGameSessionId()

Extrait l'ID de la session de jeu actuellement hébergée par le processus serveur, si ce dernier est actif.

Syntaxe

```
FGameLiftStringOutcome GetGameSessionId()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie l'ID de session de jeu en tant qu'objet `FGameLiftStringOutcome`. En cas d'échec, renvoie un message d'erreur.

GetInstanceCertificate()

Récupère l'emplacement du fichier d'un certificat TLS codé PEM associé à la flotte et à ses instances. AWS Certificate Manager génère ce certificat lorsque vous créez une nouvelle flotte avec la configuration du certificat définie sur `GENERATED`. Utilisez ce certificat pour établir une connexion sécurisée avec un client de jeu et pour chiffrer la communication client/serveur.

Syntaxe

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de succès, renvoie un `GetInstanceCertificateOutcome` objet contenant l'emplacement du fichier de certificats TLS et de la chaîne de certificats de la flotte, qui sont stockés sur l'instance. Un fichier de certificat racine, extrait de la chaîne de certificats, est également stocké sur l'instance. En cas d'échec, renvoie un message d'erreur.

Pour plus d'informations sur le certificat et les données de la chaîne de certificats, consultez la section [Éléments de GetCertificate réponse](#) dans la référence de l' AWS Certificate Manager API.

GetSdkVersion()

Renvoie le numéro de version actuel du kit SDK intégré dans le processus serveur.

Syntaxe

```
FGameLiftStringOutcome GetSdkVersion();
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

En cas de réussite, renvoie la version actuelle du kit SDK en tant qu'objet `FGameLiftStringOutcome`. La chaîne renvoyée inclut uniquement le numéro de version (par exemple « 3.1.5 »). En cas d'échec, renvoie un message d'erreur.

exemple

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Initialise le Amazon GameLift Servers SDK. Cette méthode doit être appelée au lancement, avant toute autre Amazon GameLift Servers une initialisation associée se produit.

Syntaxe

```
FGameLiftGenericOutcome InitSDK()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

ProcessEnding()

Informe le Amazon GameLift Servers service que le processus du serveur est en train d'arrêter. Cette méthode doit être appelée après toutes les autres tâches de nettoyage, y compris l'arrêt de toutes les sessions de jeu actives. Cette méthode doit quitter avec un code de sortie 0 ; un code de sortie différent de 0 génère un message d'événement indiquant que le processus ne s'est pas fermé correctement.

Syntaxe

```
FGameLiftGenericOutcome ProcessEnding()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

ProcessReady()

Informe le Amazon GameLift Servers service indiquant que le processus du serveur est prêt à héberger des sessions de jeu. Appelez cette méthode après avoir invoqué [InitSDK\(\)](#) et terminé avec succès les tâches de configuration requises avant que le processus serveur puisse héberger une session de jeu. Cette méthode ne doit être appelée qu'une seule fois par processus.

Syntaxe

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

Paramètres

FProcessParamètres

Objet [FProcessParamètres](#) communiquant les informations suivantes relatives au processus serveur :

- Noms des méthodes de rappel, implémentées dans le code du serveur de jeu, que Amazon GameLift Servers le service est invoqué pour communiquer avec le processus du serveur.
- Numéro de port sur lequel le processus serveur écoute.
- Chemin d'accès à tous les fichiers spécifiques à une session de jeu que vous souhaitez Amazon GameLift Servers pour capturer et stocker.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

exemple

Consultez l'exemple de code dans [Utilisation du plug-in Unreal Engine](#).

RemovePlayerSession()

Informe le Amazon GameLift Servers service qu'un joueur possédant l'identifiant de session de joueur spécifié a déconnecté du processus du serveur. En réponse, Amazon GameLift Servers fait passer l'emplacement du joueur à disponible, ce qui permet de l'attribuer à un nouveau joueur.

Syntaxe

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```

Paramètres

playerSessionId

Identifiant unique émis par le Amazon GameLift Servers service en réponse à un appel au AWS SDK Amazon GameLift Servers Action de l'API [CreatePlayerSession](#). Le client du jeu fait référence à cet identifiant lors de la connexion au processus serveur.

Type : FString

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

StartMatchBackfill()

Envoie une demande pour trouver de nouveaux joueurs pour les machines à sous ouvertes lors d'une session de jeu créée avec FlexMatch. Voir également l'action du AWS SDK [StartMatchBackfill\(\)](#). Avec cette action, les requêtes de renvoi de correspondance peuvent être initiées par processus de serveur de jeu qui héberge la session de jeu. En savoir plus sur l'[FlexMatch fonction](#) de remblayage.

Cette action est asynchrone. Si de nouveaux joueurs sont jumelés avec succès, le Amazon GameLift Servers le service fournit des données de matchmaking mises à jour à l'aide de la fonction de rappel. `OnUpdateGameSession()`

Un processus de serveur ne peut comporter qu'une seule requête de renvoi de correspondance à la fois. Pour envoyer une nouvelle requête, appelez d'abord [StopMatchBackfill\(\)](#) pour annuler la requête d'origine.

Syntaxe

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest  
&startBackfillRequest);
```

Paramètres

FStartMatchBackfillRequest

Objet [FStartMatchBackfillRequest](#) qui communique les informations suivantes :

- ID de ticket à attribuer à la requête de renvoi. Ces informations sont facultatives ; si aucun identifiant n'est fourni, Amazon GameLift Servers en générera un automatiquement.
- Matchmaker auquel envoyer la requête. L'ARN de configuration complet est obligatoire. Cette valeur peut être acquise à partir des données matchmaker de la session de jeu.
- ID de la session de jeu en cours de renvoi.
- Données de correspondance disponibles pour les joueurs actuels de la session de jeu.

Obligatoire : oui

Valeur renvoyée

En cas de réussite, renvoie le ticket de renvoi de correspondance sous la forme d'un objet `FGameLiftStringOutcome`. En cas d'échec, renvoie un message d'erreur. L'état du ticket peut être suivi à l'aide de l'action du AWS SDK [DescribeMatchmaking\(\)](#).

StopMatchBackfill()

Annule une requête de renvoi de correspondance active qui a été créée avec [StartMatchBackfill\(\)](#). Voir également l'action du AWS SDK [StopMatchmaking\(\)](#). En savoir plus sur l'[FlexMatch fonction](#) de remblayage.

Syntaxe

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

Paramètres

StopMatchBackfillRequest

Objet [FStopMatchBackfillRequest](#) qui identifie le ticket de correspondance à annuler :

- Identifiant de ticket attribué à la requête de renvoi en cours d'annulation
- matchmaker auquel a été envoyée la requête
- session de jeu associée à la requête de renvoi

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

TerminateGameSession()

Cette méthode est obsolète depuis la version 4.0.1. Au lieu de cela, le processus du serveur doit appeler [ProcessEnding\(\)](#) après la fin d'une session de jeu.

Informe le Amazon GameLift Servers service indiquant que le processus du serveur a mis fin à la session de jeu en cours. Cette action est appelée lorsque le processus du serveur restera actif et prêt à héberger une nouvelle session de jeu. Il ne doit être appelé qu'une fois la procédure de fin de session de jeu terminée, car il indique à Amazon GameLift Servers que le processus du serveur est immédiatement disponible pour héberger une nouvelle session de jeu.

Cette action n'est pas appelée si le processus du serveur doit être arrêté après l'arrêt de la session de jeu. Appelez plutôt [ProcessEnding\(\)](#) pour signaler que la session de jeu et le processus du serveur se terminent.

Syntaxe

```
FGameLiftGenericOutcome TerminateGameSession()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

UpdatePlayerSessionCreationPolicy()

Met à jour la capacité de la session de jeu à accepter de nouvelles sessions de joueur. Une session de jeu peut être définie pour accepter ou refuser toutes les nouvelles sessions joueur. (Voir également l'[UpdateGameSession\(\)](#) action dans le Amazon GameLift Servers Référence de l'API de service).

Syntaxe

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

Paramètres

Politique

Valeur indiquant si la session de jeu accepte ou non de nouveaux joueurs.

Type : enum `EPlayerSessionCreationPolicy`. Les valeurs valides sont les suivantes :

- `ACCEPT_ALL` — Accepte toutes les sessions de nouveaux joueurs.
- `DENY_ALL` — Refuse toutes les sessions de nouveaux joueurs.

Obligatoire : oui

Valeur renvoyée

Renvoie un résultat générique consistant en un succès ou un échec avec un message d'erreur.

Amazon GameLift ServersAmazon GameLift ServersréférenceRealtime

Cette section contient de la documentation de référence pour le Amazon GameLift Servers Amazon GameLift Servers Realtime SDK. Il inclut l'API Realtime client ainsi que des conseils pour configurer votre Amazon GameLift Servers Realtime script.

Rubriques

- [Amazon GameLift ServersRealtime référence de l'API client \(C#\)](#)
- [Référence de script pour Amazon GameLift ServersRealtime](#)

Amazon GameLift ServersRealtime référence de l'API client (C#)

Utilisez l'API Realtime client pour préparer vos clients de jeux multijoueurs à utiliser avec Amazon GameLift Servers Amazon GameLift ServersRealtime. L'API de client contient un ensemble de rappels d'API synchrones et asynchrones qui permettent à un client de jeu de se connecter à un serveur Realtime et d'échanger des messages et des données avec d'autres clients de jeu via le serveur.

Cet API est défini dans les bibliothèques suivantes :

Client.cs

- [Actions synchrones](#)
- [Rappels asynchrones](#)
- [Types de données](#)

Pour configurer l'API Realtime client

1. Téléchargez le [SDK Amazon GameLift ServersRealtime du client](#).
2. Développez les bibliothèques SDK C#. Localisez le fichier de solution `GameLiftRealtimeClientSdkNet45.sln`. Consultez le fichier `README.md` relatif au kit SDK C# Server pour connaître la configuration minimale requise et les options de build supplémentaires. Chargez le fichier de solution dans un IDE. Pour générer les bibliothèques du SDK, restaurez les NuGet packages et créez la solution.
3. Ajoutez les bibliothèques de client Realtime au projet de votre client de jeu.

Amazon GameLift ServersRealtime référence de l'API client (C#) : Actions

Cette référence d'API Realtime client C# peut vous aider à préparer votre jeu multijoueur pour une utilisation sur Amazon GameLift Servers Realtime des Amazon GameLift Servers flottes.

- Actions synchrones
- [Rappels asynchrones](#)
- [Types de données](#)

Cliente ()

Initialise un nouveau client pour communiquer avec le serveur Realtime et identifie le type de connexion à utiliser.

Syntaxe

```
public Client(ClientConfiguration configuration)
```

Paramètres

clientConfiguration

Détails de configuration spécifiant le type de client/server connexion. Vous pouvez choisir d'appeler `Client()` sans ce paramètre ; cependant, cette approche se traduit par une connexion non sécurisée par défaut.

Type : [ClientConfiguration](#)

Obligatoire : non

Valeur renvoyée

Renvoie une instance du client Realtime à utiliser avec la communication avec le serveur Realtime.

Connect()

Demande une connexion à un processus serveur qui héberge une session de jeu.

Syntaxe

```
public ConnectionStatus Connect(string endpoint, int remoteTcpPort, int listenPort,
    ConnectionToken token)
```

Paramètres

point de terminaison

Adresse IP ou nom DNS de la session de jeu à laquelle se connecter. Le point de terminaison est spécifié dans un `GameSession` objet, qui est renvoyé en réponse à un appel du client aux actions de l'Amazon GameLift ServersAPI du AWS SDK [StartGameSessionPlacementCreateGameSession](#), ou [DescribeGameSessions](#).

 Note

Si le serveur Realtime s'exécute sur une flotte avec un certificat TLS, vous devez utiliser le nom DNS.

Type : String

Obligatoire : oui

remoteTcpPort

Numéro de port pour la connexion TCP attribuée à la session de jeu. Ces informations sont spécifiées dans un `GameSession` objet, qui est renvoyé en réponse à une [StartGameSessionPlacementCreateGameSessionDescribeGameSession](#) demande ou.

Type : entier

Valeurs valides : 1900 à 2000.

Obligatoire : oui

listenPort

Numéro de port que le client de jeu surveille pour des messages envoyés à l'aide du canal UDP.

Type : entier

Valeurs valides : 33400 à 33500.

Obligatoire : oui

jeton

Informations facultatives qui identifient la demande client de jeu au processus serveur.

Type : [ConnectionToken](#)

Obligatoire : oui

Valeur renvoyée

Renvoie une valeur [ConnectionStatus](#) enum indiquant l'état de connexion du client.

Disconnect()

Lorsque vous êtes connecté à une session de jeu, déconnecte le client de jeu à partir de la session de jeu.

Syntaxe

```
public void Disconnect()
```

Paramètres

Cette action n'a aucun paramètre.

Valeur renvoyée

Cette méthode ne renvoie rien.

NewMessage()

Crée un nouvel objet de message avec un code d'opération spécifié. Une fois qu'un message est renvoyé, complétez le contenu du message en spécifiant une cible, en mettant à jour le mode de remise, et en ajoutant une charge utile de données en fonction de vos besoins. Une fois terminé, envoyez le message à l'aide de `SendMessage()`.

Syntaxe

```
public RTMessage NewMessage(int opCode)
```

Paramètres

opCode

Code d'opération défini par le développeur qui identifie un événement ou une action de jeu, par exemple un mouvement de joueur ou une notification.

Type : entier

Obligatoire : oui

Valeur renvoyée

Renvoie un objet [RTMessage](#) contenant l'opération spécifiée par défaut et la méthode de livraison de code. Le paramètre d'intention d'acheminement est défini sur FAST par défaut.

SendMessage()

Envoie un message à un joueur ou un groupe à l'aide de la méthode de livraison spécifiée.

Syntaxe

```
public void SendMessage(RTMessage message)
```

Paramètres

message

Message d'objet qui spécifie le destinataire cible, la méthode de livraison et le contenu du message.

Type : [RTMessage](#)

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien.

JoinGroup()

Ajoute le joueur à un groupe spécifié. Les groupes peuvent contenir n'importe quel joueur connecté au jeu. Une fois qu'il a rejoint le serveur, le joueur reçoit tous les futurs messages envoyés au groupe et peut envoyer des messages à l'ensemble du groupe.

Syntaxe

```
public void JoinGroup(int targetGroup)
```

Paramètres

targetGroup

ID unique qui identifie le groupe auquel ajouter le joueur. IDsLes groupes sont définis par le développeur.

Type : entier

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien. Étant donné que cette demande est envoyée à l'aide de la méthode de livraison fiable (TCP), un échec de demande déclenche le rappel [OnError\(\)](#).

LeaveGroup()

Retire le joueur d'un groupe spécifié. Une fois qu'il n'est plus dans le groupe, le joueur ne reçoit pas les messages envoyés au groupe et ne peut envoyer de messages à l'ensemble du groupe.

Syntaxe

```
public void LeaveGroup(int targetGroup)
```

Paramètres

targetGroup

ID unique qui identifie le groupe duquel retirer le joueur. IDs Les groupes sont définis par le développeur.

Type : entier

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien. Étant donné que cette demande est envoyée à l'aide de la méthode de livraison fiable (TCP), un échec de demande déclenche le rappel [OnError\(\)](#).

RequestGroupMembership()

Demande qu'une liste de joueurs dans le groupe spécifié soit envoyée au client de jeu. Tout joueur peut demander ces informations, qu'il s'agisse ou non d'un membre du groupe. En réponse à cette demande, la liste d'adhésion est envoyée au client via un rappel [OnGroupMembershipUpdated\(\)](#).

Syntaxe

```
public void RequestGroupMembership(int targetGroup)
```

Paramètres

targetGroup

ID unique qui identifie le groupe pour lequel obtenir des informations sur l'adhésion. IDs Les groupes sont définis par le développeur.

Type : entier

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien.

Amazon GameLift ServersRéférence de l'API client (C#) : rappels asynchrones

Utilisez cette référence d'API Realtime client C# pour vous aider à préparer votre jeu multijoueur en vue de son utilisation sur Amazon GameLift Servers Realtime des Amazon GameLift Servers flottes.

- [Actions synchrones](#)
- Rappels asynchrones
- [Types de données](#)

Un client de jeu a besoin de mettre en œuvre ces méthodes de rappel pour répondre aux événements. Le serveur Realtime invoque ces rappels pour envoyer des informations liées au jeu pour le client de jeu. Les rappels pour les mêmes événements peuvent également être mis en œuvre avec une logique de jeu personnalisée dans le script de serveur Realtime. Consultez [Rappels de script pour Amazon GameLift ServersRealtime](#).

Les méthodes de rappel sont définies dans `ClientEvents.cs`.

OnOpen()

Appelée lorsque le processus serveur accepte la demande de connexion d'un client de jeu et ouvre une connexion.

Syntaxe

```
public void OnOpen()
```

Paramètres

Cette méthode ne prend aucun paramètre.

Valeur renvoyée

Cette méthode ne renvoie rien.

OnClose()

Appelée lorsque le processus serveur met fin à la connexion avec le client de jeu, comme lorsqu'une session de jeu se termine.

Syntaxe

```
public void OnClose()
```

Paramètres

Cette méthode ne prend aucun paramètre.

Valeur renvoyée

Cette méthode ne renvoie rien.

OnError()

Appelée lorsqu'un échec se produit pour une demande d'API du client en temps réel. Ce rappel peut être personnalisé afin de prendre en charge une grande variété d'erreurs de connexion.

Syntaxe

```
private void OnError(byte[] args)
```

Paramètres

Cette méthode ne prend aucun paramètre.

Valeur renvoyée

Cette méthode ne renvoie rien.

OnDataReceived()

Appelée lorsque le client de jeu reçoit un message à partir du serveur Realtime. Il s'agit de la méthode principale de réception de messages et de notifications par un client de jeu.

Syntaxe

```
public void OnDataReceived(DataReceivedEventArgs dataReceivedEventArgs)
```

Paramètres

dataReceivedEventArgs

Informations liées à l'activité de messages.

Type : [DataReceivedEventArgs](#)

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien.

OnGroupMembershipUpdated()

Appelée lorsque l'adhésion à un groupe auquel le joueur appartient a été mise à jour. Ce rappel est également invoqué lorsqu'un client appelle RequestGroupMembership.

Syntaxe

```
public void OnGroupMembershipUpdated(GroupMembershipEventArgs groupMembershipEventArgs)
```

Paramètres

groupMembershipEventArgs

Informations liées à l'activité d'adhésion d'un groupe.

Type : [GroupMembershipEventArgs](#)

Obligatoire : oui

Valeur renvoyée

Cette méthode ne renvoie rien.

Amazon GameLift ServersRealtimeréférence de l'API client (C#) : types de données

Cette référence d'API Realtime client C# peut vous aider à préparer votre jeu multijoueur pour une utilisation sur Amazon GameLift Servers Realtime des Amazon GameLift Servers flottes.

- [Actions synchrones](#)
- [Rappels asynchrones](#)
- Les types de données

ClientConfiguration

Informations sur la façon dont le client de jeu se connecte à un serveur Realtime.

Table des matières

ConnectionType

Type de client/server connexion à utiliser, sécurisée ou non sécurisée. Si vous ne spécifiez pas de type de connexion, la valeur par défaut est non sécurisée.

Note

Lorsque vous vous connectez à un serveur en temps réel sur un parc sécurisé avec un certificat TLS, vous devez utiliser la valeur `RT_OVER_WSS_DTLS_.TLS12`.

Type : valeur `ConnectionType` [enum](#) .

Obligatoire : non

ConnectionToken

Informations sur le and/or joueur client du jeu qui demande une connexion à un Realtime serveur.

Table des matières

playerSessionId

Identifiant unique émis Amazon GameLift Servers lors de la création d'une nouvelle session de joueur. Un identifiant de session de joueur est spécifié dans un `PlayerSession` objet, qui est renvoyé en réponse à un appel du client aux actions de l'GameLift API [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#), ou [DescribePlayerSessions](#).

Type : String

Obligatoire : oui

payload

Informations définies par le développeur à communiquer au serveur Realtime au moment de la connexion. Cela inclut toutes les données arbitraires qui peuvent être utilisées pour un

mécanisme de connexion personnalisé. Pour obtenir des exemples, une charge utile peut fournir des informations d'authentification à traiter par le script de serveur Realtime avant d'autoriser la connexion d'un client.

Type : tableau d'octets

Obligatoire : non

RTMessage

Contenu et informations de livraison pour un message. Un message doit spécifier soit un joueur ou un groupe cible.

Table des matières

opCode

Code d'opération défini par le développeur qui identifie un événement ou une action de jeu, par exemple un mouvement de joueur ou une notification. Un code Op d'un message fournit le contexte pour les données utiles qui sont fournies. Le code d'opération est déjà défini pour les messages créés à l'aide de `NewMessage()`, mais il peut être modifié à tout moment.

Type : entier

Obligatoire : oui

targetPlayer

ID unique qui identifie le destinataire voulu du message envoyé. La cible peut être le serveur lui-même (en utilisant l'ID de serveur) ou un autre joueur (à l'aide d'un ID de joueur).

Type : entier

Obligatoire : non

targetGroup

ID unique qui identifie le groupe destinataire voulu du message envoyé. Les groupes sont définis par le développeur.

Type : entier

Obligatoire : non

deliveryIntent

Indique s'il convient d'envoyer le message à l'aide de la connexion TCP fiable ou à l'aide du canal UDP rapide. Messages créés à l'aide de [NewMessage\(\)](#).

Type : DeliveryIntent enum

Valeurs valides : RAPIDE | FIABLE

Obligatoire : oui

payload

Contenu du message. Ces informations sont structurées de telle manière à être traitées par le client de jeu en fonction du code d'opération connexe. Elles peuvent contenir des données d'état du jeu ou d'autres informations qui doivent être communiquées entre les clients de jeu ou entre un client de jeu et le serveur en temps réel.

Type : tableau d'octets

Obligatoire : non

DataReceivedEventArgs

Les données fournies avec un rappel [OnDataReceived\(\)](#).

Table des matières

sender

ID unique qui identifie l'entité (ID de joueur ou ID de serveur) ayant transmis le message.

Type : entier

Obligatoire : oui

opCode

Code d'opération défini par le développeur qui identifie un événement ou une action de jeu, par exemple un mouvement de joueur ou une notification. Un code Op d'un message fournit le contexte pour les données utiles qui sont fournies.

Type : entier

Obligatoire : oui

data

Contenu du message. Ces informations sont structurées de telle manière à être traitées par le client de jeu en fonction du code d'opération connexe. Elles peuvent contenir des données d'état du jeu ou d'autres informations qui doivent être communiquées entre les clients de jeu ou entre un client de jeu et le serveur en temps réel.

Type : tableau d'octets

Obligatoire : non

GroupMembershipEventArgs

Les données fournies avec un rappel [OnGroupMembershipUpdated\(\)](#).

Table des matières

sender

ID unique qui identifie le joueur qui a demandé une mise à jour d'adhésion au groupe.

Type : entier

Obligatoire : oui

opCode

Code d'opération défini par le développeur qui identifie un événement ou une action de jeu.

Type : entier

Obligatoire : oui

groupId

ID unique qui identifie le groupe destinataire voulu du message envoyé. IDs Les groupes sont définis par le développeur.

Type : entier

Obligatoire : oui

playerId

Liste des IDs joueurs actuellement membres du groupe spécifié.

Type : tableau integer

Obligatoire : oui

Enums

Les énumérations définies pour le SDK client pour Amazon GameLift Servers Realtime sont définies comme suit :

ConnectionStatus

- **CONNECTÉ** — Le client du jeu est connecté au Realtime serveur via une connexion TCP uniquement. Tous les messages sont envoyés via TCP ; indépendamment de l'intention de livraison.
- **CONNECTED_SEND_FAST** — Le client du jeu est connecté au Realtime serveur via une connexion TCP et UDP. Cependant, la capacité de recevoir des messages via UDP n'est pas encore vérifiée. Par conséquent, tous les messages envoyés au client de jeu utilisent TCP.
- **CONNECTED_SEND_AND_RECEIVE_FAST** — Le client du jeu est connecté au serveur via une connexion TCP et UDP. Realtime Le client de jeu peut envoyer et recevoir des messages à l'aide de TCP ou UDP.
- **CONNECTING** Le client de jeu a envoyé une demande de connexion et le serveur Realtime est en cours de traitement.
- **DISCONNECTED_CLIENT_CALL** — Le client du jeu a été déconnecté du Realtime serveur en réponse à une [Disconnect\(\)](#) demande du client du jeu.
- **DÉCONNECTÉ** — Le client du jeu a été déconnecté du Realtime serveur pour une raison autre qu'un appel de déconnexion du client.

ConnectionType

- **TLS12_RT_OVER_WSS_DTLS_** — Type de connexion sécurisée.

À utiliser avec des serveurs en temps réel exécutés sur une Amazon GameLift Servers flotte avec un certificat TLS généré. Lorsque vous utilisez une connexion sécurisée, le trafic TCP est chiffré à l'aide de TLS 1.2 et le trafic UDP est chiffré à l'aide de DTLS 1.2.

- **RT_OVER_WS_UDP_UNSECURED** — Type de connexion non sécurisé.
- **RT_OVER_WEBSOCKET** — Type de connexion non sécurisé. Cette valeur n'est plus préférée.

DeliveryIntent

- FAST — Fourni via un canal UDP.
- FIABLE — Fourni via une connexion TCP.

Référence de script pour Amazon GameLift ServersRealtime

Utilisez ces ressources pour développer une logique personnalisée dans vos scripts Realtime.

Rubriques

- [Rappels de script pour Amazon GameLift ServersRealtime](#)
- [Amazon GameLift ServersRealtimeinterface](#)

Rappels de script pour Amazon GameLift ServersRealtime

Vous pouvez fournir une logique personnalisée pour répondre à des événements en mettant en œuvre ces rappels dans votre script Realtime.

Initialiser

Initialise le serveur Realtime et reçoit une interface de serveur en temps réel.

Syntaxe

```
init(rtsession)
```

onMessage

Appelé lorsqu'un message reçu est envoyé au serveur.

Syntaxe

```
onMessage(gameMessage)
```

onHealthCheck

Appelé pour définir le statut d'intégrité de la session de jeu. Par défaut, le statut d'intégrité est sain (ou `true`). Ce rappel peut être implémenté pour effectuer des vérifications de l'état personnalisées et renvoyer un statut.

Syntaxe

```
onHealthCheck()
```

onStartGameSession

Appelé lorsqu'une nouvelle session de jeu démarre, avec un objet de session de jeu transmis.

Syntaxe

```
onStartGameSession(session)
```

onProcessTerminate

Appelé lorsque le processus serveur est en cours de résiliation par le service Amazon GameLift Servers. Il peut agir comme déclencheur pour quitter de façon nette la session de jeu. Il n'y a pas besoin d'appeler `processEnding()`.

Syntaxe

```
onProcessTerminate()
```

onPlayerConnect

Appelé lorsqu'un joueur demande une connexion et a réussi la validation initiale.

Syntaxe

```
onPlayerConnect(connectMessage)
```

onPlayerAccepted

Appelé lorsqu'une connexion de joueur est acceptée.

Syntaxe

```
onPlayerAccepted(player)
```

onPlayerDisconnect

Appelé lorsqu'un joueur se déconnecte de la session de jeu, que ce soit en envoyant une demande de déconnexion ou par d'autres moyens.

Syntaxe

```
onPlayerDisconnect(peerId)
```

onProcessStarted

Appelé lorsqu'un processus serveur est démarré. Ce rappel autorise le script à effectuer toutes les tâches personnalisées nécessaires pour préparer l'hébergement d'une session de jeu.

Syntaxe

```
onProcessStarted(args)
```

onSendToJoueur

Appelé lorsqu'un message est reçu sur le serveur d'un joueur pour être livré à un autre joueur. Ce processus s'exécute avant que le message soit livré.

Syntaxe

```
onSendToPlayer(gameMessage)
```

onSendToGroupe

Appelé lorsqu'un message est reçu sur le serveur d'un joueur pour être livré à un groupe. Ce processus s'exécute avant que le message soit livré.

Syntaxe

```
onSendToGroup(gameMessage)
```

onPlayerJoinGroupe

Appelé lorsqu'un joueur envoie une demande pour rejoindre un groupe.

Syntaxe

```
onPlayerJoinGroup(groupId, peerId)
```

onPlayerLeaveGroupe

Appelé lorsqu'un joueur envoie une demande pour quitter un groupe.

Syntaxe

```
onPlayerLeaveGroup(groupId, peerId)
```

Amazon GameLift ServersRealtimeinterface

Lorsqu'un Amazon GameLift Servers Realtime script s'initialise, une interface est renvoyée au Realtime serveur. Cette rubrique décrit les propriétés et les méthodes disponibles via cette interface. Obtenez plus d'informations sur l'écriture de scripts Realtime et consultez un exemple de script détaillé dans [Création d'un Realtime script](#).

L'interface Realtime permet d'accéder aux objets suivants :

- séance
- joueur (player)
- message de jeu (gameMessage)
- configuration

Objet session Realtime

Utilisez ces méthodes pour accéder aux informations liées au serveur et effectuer des actions liées au serveur.

getPlayers()

Récupère une liste de pairs IDs pour les joueurs actuellement connectés à la session de jeu. Renvoie un tableau d'objets joueur.

Syntaxe

```
rtSession.getPlayers()
```

broadcastGroupMembershipMettre à jour ()

Déclenche la livraison d'une mise à jour de la liste des membres d'un groupe de joueurs. Spécifiez l'adhésion à diffuser (groupIdTodiffusion) et le groupe devant recevoir la mise à jour (targetGroupId).

Le groupe IDs doit être un entier positif ou « -1 » pour indiquer tous les groupes. Voir un [Amazon GameLift ServersRealtimeexemple de script](#) exemple de groupe IDs défini par l'utilisateur.

Syntaxe

```
rtSession.broadcastGroupMembershipUpdate(groupIdToBroadcast, targetGroupId)
```

getServerId()

Récupère l'identifiant de pair unique du serveur, qui est utilisé pour router les messages vers le serveur.

Syntaxe

```
rtSession.getServerId()
```

getAllPlayersGroupId()

Récupère l'ID de groupe pour le groupe par défaut qui contient tous les joueurs actuellement connectés à la session de jeu.

Syntaxe

```
rtSession.getAllPlayersGroupId()
```

processEnding()

Déclenche la résiliation du serveur de jeu par le serveur Realtime. Cette fonction doit être appelée depuis le script Realtime pour quitter de façon nette une session de jeu.

Syntaxe

```
rtSession.processEnding()
```

getGameSessionIdentifiant ()

Récupère l'ID unique de la session de jeu en cours d'exécution.

Syntaxe

```
rtSession.getGameSessionId()
```

getLogger()

Récupère l'interface pour la journalisation. Utilisez cette commande pour consigner les instructions qui seront capturées dans vos journaux de session de jeu. L'enregistreur prend en charge les instructions « info », « warn » et « error ». Par exemple : `logger.info("<string>")`.

Syntaxe

```
rtSession.getLogger()
```

sendMessage()

Envoie un message, créé à l'aide de `newTextGameMessage` ou `newBinaryGameMessage`, à partir du serveur Realtime à un joueur destinataire à l'aide du canal UDP. Identifiez le destinataire à l'aide de l'ID de pair du joueur.

Syntaxe

```
rtSession.sendMessage(gameMessage, targetPlayer)
```

sendGroupMessage()

Envoie un message, créé à l'aide de `newTextGameMessage` ou `newBinaryGameMessage`, à partir du serveur Realtime à tous les joueurs d'un groupe de joueurs à l'aide du canal UDP. Le groupe IDs doit être un entier positif ou « -1 » pour indiquer tous les groupes. Voir un [Amazon GameLift ServersRealtimeexemple de script](#) exemple de groupe IDs défini par l'utilisateur.

Syntaxe

```
rtSession.sendGroupMessage(gameMessage, targetGroup)
```

sendReliableMessage()

Envoie un message, créé à l'aide de `newTextGameMessage` ou `newBinaryGameMessage`, à partir du serveur Realtime à un joueur destinataire à l'aide du canal TCP. Identifiez le destinataire à l'aide de l'ID de pair du joueur.

Syntaxe

```
rtSession.sendReliableMessage(gameMessage, targetPlayer)
```

sendReliableGroupUn message ()

Envoie un message, créé à l'aide de `newTextGameMessage` ou `newBinaryGameMessage`, à partir du serveur Realtime à tous les joueurs d'un groupe de joueurs à l'aide du canal TCP. Groupe IDs qui doit être un entier positif ou « -1 » pour indiquer tous les groupes. Voir un [Amazon GameLift ServersRealtimeexemple de script](#) exemple de groupe IDs défini par l'utilisateur.

Syntaxe

```
rtSession.sendReliableGroupMessage(gameMessage, targetGroup)
```

newTextGameUn message ()

Crée un nouveau message contenant du texte, à envoyer depuis le serveur aux destinataires des joueurs à l'aide des `SendMessage` fonctions. Le format des messages est similaire à celui utilisé dans le SDK client pour Realtime (voir [RTMessage](#)). Renvoie un objet `gameMessage`.

Syntaxe

```
rtSession.newTextGameMessage(opcode, sender, payload)
```

newBinaryGameUn message ()

Crée un nouveau message contenant des données binaires, à envoyer depuis le serveur aux destinataires des joueurs à l'aide des `SendMessage` fonctions. Le format des messages est similaire à celui utilisé dans le SDK client pour Realtime (voir [RTMessage](#)). Renvoie un objet `gameMessage`.

Syntaxe

```
rtSession.newBinaryGameMessage(opcode, sender, binaryPayload)
```

Objet joueur

Accédez aux informations liées au joueur.

`player.peerId`

ID unique affecté à un client de jeu lorsqu'il se connecte au serveur Realtime et rejoint la session de jeu.

joueur. `playerSessionId`

ID de session de joueur qui a été référencé par le client de jeu lorsqu'il s'est connecté au serveur Realtime et a rejoint la session de jeu.

Objet message de jeu

Utilisez ces méthodes pour accéder aux messages qui sont reçus par le serveur Realtime. Les messages reçus des clients de jeu ont la structure [RTMessage](#).

`getPayloadAsTexte ()`

Récupère la charge utile du message du jeu sous forme de texte.

Syntaxe

```
gameMessage.getPayloadAsText()
```

`gameMessage.opcode`

Code d'opération contenu dans un message.

`gameMessage.payload`

Charge utile contenue dans un message. Peut être de type texte ou binaire.

`gameMessage.sender`

ID de pair du client de jeu qui a envoyé un message.

`gameMessage.reliable`

Booléen indiquant si le message a été envoyé via TCP (true) ou UDP (false).

Objet de configuration

L'objet de configuration peut être utilisé pour remplacer les configurations par défaut.

Configuration. Max Players

Le nombre maximum de connexions client/serveur pouvant être acceptées par RealTimeServers.

La valeur par défaut est 32.

configuration. pingIntervalTime

Intervalle de temps en millisecondes pendant lequel le serveur tentera d'envoyer un ping à tous les clients connectés pour vérifier que les connexions sont saines.

La valeur par défaut est de 3000 ms.

Événements de placement lors des sessions de jeu

Amazon GameLift Servers émet des événements pour chaque demande de placement de session de jeu au fur et à mesure de son traitement. Vous pouvez publier ces événements sur une rubrique Amazon SNS, comme décrit dans [Configurer la notification des événements pour le placement des sessions de jeu](#). Ces événements sont également transmis à Amazon CloudWatch Events en temps quasi réel et dans la mesure du possible.

Cette rubrique décrit la structure des événements de placement de session de jeu et fournit un exemple pour chaque type d'événement. Pour plus d'informations sur le statut des demandes de placement de session de jeu, consultez [GameSessionPlacement](#) le Amazon GameLift Servers Référence d'API.

Syntaxe des événements de placement

Les événements sont représentés sous la forme d'objets JSON. La structure des événements est conforme au modèle des CloudWatch événements, avec des champs de haut niveau similaires et des détails spécifiques au service.

Les champs de niveau supérieur sont les suivants (voir le [modèle d'événement](#) pour plus de détails) :

version

Ce champ est toujours défini sur 0 (zéro).

id

Identifiant de suivi unique pour l'événement.

detail-type

La valeur est toujours `GameLift Queue Placement Event`.

source

La valeur est toujours `saws.gamelift`.

compte

Le AWS compte utilisé pour gérer Amazon GameLift Servers.

time

Horodatage de l'événement.

region

AWS Région dans laquelle la demande de placement est traitée. Il s'agit de la région dans laquelle se trouve la file d'attente des sessions de jeu en cours d'utilisation.

resources

Valeur ARN de la file d'attente de session de jeu qui traite la demande de placement.

PlacementFulfilled

La demande de placement a été traitée avec succès. Une nouvelle session de jeu a été démarrée et de nouvelles sessions de joueur ont été créées pour chaque joueur répertorié dans la demande de placement de session de jeu. Les informations de connexion du joueur sont disponibles.

Syntaxe détaillée :

Identifiant du lieu

Identifiant unique attribué à la demande de placement de session de jeu.

port

Le numéro de port de la nouvelle session de jeu.

gameSessionArn

L'identifiant ARN de la nouvelle session de jeu.

ipAddress

Adresse IP de la session de jeu.

Nom DNS

Identifiant DNS attribué à l'instance qui exécute la nouvelle session de jeu. Le format de valeur est différent selon que l'instance qui exécute la session de jeu est compatible TLS ou non. Lorsqu'ils

se connectent à une session de jeu sur une flotte compatible TLS, les joueurs doivent utiliser le nom DNS et non l'adresse IP.

Flottes compatibles TLS : `<unique identifieur>.<region identifieur>.amazongamelift.com`

Non-TLS-enabled flottes : `ec2-<unique identifieur>.compute.amazonaws.com`.

`startTime`

Horodatage indiquant à quel moment cette demande a été placée dans la file d'attente.

`endTime`

Horodatage indiquant la date à laquelle cette demande a été satisfaite.

`gameSessionRegion`

AWS Région de la flotte qui héberge la session de jeu. Cela correspond au jeton de région dans `leGameSessionArn`.

`placedPlayerSessions`

L'ensemble des sessions de joueur créées pour chaque joueur dans la demande de placement de session de jeu.

exemple

```
{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
  ],
  "detail": {
    "type": "PlacementFulfilled",
    "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
    "port": "6262",
```

```

    "gameSessionArn": "arn:aws:gamelift:us-west-2::gamesession/
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa/4444dddd-55ee-66ff-77aa-8888bbbb99cc",
    "ipAddress": "98.987.98.987",
    "dnsName": "ec2-12-345-67-890.us-west-2.compute.amazonaws.com",
    "startTime": "2021-03-01T15:50:49.741Z",
    "endTime": "2021-03-01T15:50:52.084Z",
    "gameSessionRegion": "us-west-2",
    "placedPlayerSessions": [
      {
        "playerId": "player-1"
        "playerSessionId": "psess-1232131232324124123123"
      }
    ]
  }
}

```

PlacementCancelled

La demande de placement a été annulée par un appel au GameLift service [StopGameSessionPlacement](#).

Détail :

Identifiant du lieu

Identifiant unique attribué à la demande de placement de session de jeu.

startTime

Horodatage indiquant à quel moment cette demande a été placée dans la file d'attente.

endTime

Horodatage indiquant la date à laquelle cette demande a été annulée.

exemple

```

{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",

```

```

"account": "123456789012",
"time": "2021-03-01T15:50:52Z",
"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementCancelled",
  "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}

```

PlacementTimedOut

Le placement de la session de jeu n'a pas été effectué correctement avant l'expiration du délai de la file d'attente. La demande de placement peut être soumise à nouveau si nécessaire.

Détail :

Identifiant du lieu

Identifiant unique attribué à la demande de placement de session de jeu.

startTime

Horodatage indiquant à quel moment cette demande a été placée dans la file d'attente.

endTime

Horodatage indiquant la date à laquelle cette demande a été annulée.

exemple

```

{
  "version": "0",
  "id": "1111aaaa-bb22-cc33-dd44-5555eeee66ff",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2021-03-01T15:50:52Z",

```

```
"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:123456789012:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementTimedOut",
  "placementId": "9999ffff-88ee-77dd-66cc-5555bb44aa",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}
```

PlacementFailed

Amazon GameLift Servers n'a pas pu répondre à la demande de session de jeu. Cela est généralement dû à une erreur interne inattendue. La demande de placement peut être soumise à nouveau si nécessaire.

Détail :

Identifiant du lieu

Identifiant unique attribué à la demande de placement de session de jeu.

startTime

Horodatage indiquant à quel moment cette demande a été placée dans la file d'attente.

endTime

Horodatage indiquant à quel moment cette demande a échoué.

exemple

```
{
  "version": "0",
  "id": "39c978f3-ba46-3f7c-e787-55bfcca1bd31",
  "detail-type": "GameLift Queue Placement Event",
  "source": "aws.gamelift",
  "account": "252386620677",
  "time": "2021-03-01T15:50:52Z",
```

```

"region": "us-east-1",
"resources": [
  "arn:aws:gamelift:us-west-2:252386620677:gamesessionqueue/MegaFrogRace-NA"
],
"detail": {

  "type": "PlacementFailed",
  "placementId": "e4a1119a-39af-45cf-a990-ef150fe0d453",
  "startTime": "2021-03-01T15:50:49.741Z",
  "endTime": "2021-03-01T15:50:52.084Z"
}
}

```

Amazon GameLift Servers Versions de l'AMI

Le tableau suivant identifie les dernières images de machine Amazon (AMIs) Amazon GameLift Servers utilisées pour l'EC2 hébergement géré. Comme décrit dans [Configuration et analyse des vulnérabilités dans Amazon GameLift Servers](#), vous devez régulièrement créer de nouvelles EC2 flottes Amazon GameLift Servers gérées pour déployer les dernières mises à jour des versions de l'AMI.

AMIs à utiliser avec le SDK de serveur pour 5+ Amazon GameLift Servers

Amazon GameLift Servers utilise ce qui suit AMIs pour héberger des serveurs de jeu intégrés au SDK du serveur pour la Amazon GameLift Servers version 5.

Amazon GameLift Servers image	Architecture	Patch le plus récent	AWS Image de base
Amazon Linux 2023 BASE_AMI_LINUX_2023	x86	12/06/2025	Amazon Linux 2023 AMI 2023.7.20 250609.0 x86_64 HVM kernel-6.1 (notes de mise à jour)
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_5	x86	12/06/2025	Amazon Linux 2 Kernel 5.10 AMI 2.0.20250603.0 x86_64 HVM gp2 (notes de mise à jour)
Windows 2016	x86	12/06/2025	Windows_Server-2016-English-Full-Base-2025.06.12 (notes de mise à jour)

Amazon GameLift Serversimage	Architecture	Patch le plus récent	AWS Image de base
BASE_AMI_WINDOWS_2016_SDK_5			
Amazon Linux 2023 BASE_AMI_LINUX_2023_ARM	ARM64	12/06/2025	Amazon Linux 2023 AMI 2023.7.20 250609.0 arm64 HVM kernel-6.1 (notes de mise à jour)
Amazon Linux 2 BASE_AMI_LINUX_2_ARM	ARM64	12/06/2025	Amazon Linux 2 LTS Arm64 Kernel 5.10 AMI 2.0.20250603.0 arm64 HVM gp2 (notes de mise à jour)

AMIs à utiliser avec le SDK du serveur pour 4 Amazon GameLift Servers

Amazon GameLift Servers utilise ce qui suit AMIs pour héberger des serveurs de jeu intégrés au SDK du serveur pour Amazon GameLift Servers la version 4 ou antérieure.

Amazon GameLift Serversimage	Architecture	Patch le plus récent	AWS Image de base
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_4	x86	12/06/2025	Amazon Linux 2 AMI 2.0.20250605.0 x86_64 HVM gp2 (notes de mise à jour)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_4	x86	12/06/2025	Windows_Server-2016-English-Full-Base-2025.06.12 (notes de mise à jour)

Pour plus d'informations, consultez les ressources suivantes :

- [Notes de mise à jour d'Amazon Linux 2023](#)
- [Notes de mise à jour d'Amazon Linux 2](#)

- [AWS Historique des versions de Windows AMI](#)
- [Description des modifications apportées au contenu des services de mise à jour logicielle et des services de mise à jour Windows Server pour 2024](#)

Amazon GameLift Servers points de terminaison et quotas

- [Pour les points de terminaison de service et les points de terminaison de balise que vous pouvez utiliser pour vous connecter par programmation au service, voir Points de terminaison de service. Amazon GameLift Servers](#)
- Pour les Amazon GameLift Servers quotas d'utilisation des ressources de service ou des opérations par AWS compte, consultez la section [Quotas Amazon GameLift Servers de service](#). Pour plus d'informations sur les quotas et sur la manière de demander une augmentation, consultez la section [Quotas AWS de service](#). Vous pouvez demander des augmentations de quota à l'aide de la Amazon GameLift Servers console.

Limites de l'API Amazon GameLift Servers

Les tableaux de cette section répertorient les limites de débit par défaut pour les opérations d'API Amazon GameLift Servers. Ces limites contribuent à maintenir la stabilité du service et à garantir une utilisation équitable. Chaque opération d'API est soumise à des limites spécifiques en fonction de son impact sur les ressources et de ses modèles d'utilisation.

Par défaut, la limite de débit pour chaque compte client est fixée à 500 pour les demandes en rafale et à 500 pour le trafic normal. En outre, chaque API possède sa propre paire de limites par défaut, comme indiqué ci-dessous.

Référence des limites de l'API

Le tableau suivant répertorie les limites de débit par défaut pour les opérations d'API Amazon GameLift Servers. APIs marquée par une limitation du niveau des ressources peut être soumise à des limites supplémentaires afin d'éviter les problèmes liés aux touches de raccourci de la base de données.

Référence des limites de l'API

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
AcceptMatch	100	100	Non	La limite est fixée par compte.
ClaimGameServer	40	20	Non	La limite est fixée par compte.
CreateAlias	4	1	Non	La limite est fixée par compte.
CreateBuild	10	1	Non	La limite est fixée par compte.
CreateContainerFleet	4	1	Non	La limite est fixée par compte.
CreateContainerGroupDefinition	10	1	Non	La limite est fixée par compte.
CreateFleet	4	1	Non	La limite est fixée par compte.
CreateFleetLocations	4	1	Non	La limite est fixée par compte.
CreateGameServerGroup	5	0.1	Non	La limite est fixée par compte.
CreateGameSession	50	50	Non	La limite est fixée par compte.
CreateGameSessionQueue	4	1	Non	La limite est fixée par compte.
CreateLocation	4	1	Non	La limite est fixée par compte.
CreateMatchmakingConfiguration	4	1	Non	La limite est fixée par compte.
CreateMatchmakingRuleSet	4	1	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
CreatePlayerSession	50	50	Non	La limite est fixée par compte.
CreatePlayerSessions	150	150	Non	La limite est fixée par compte.
CreateScript	10	1	Non	La limite est fixée par compte.
CreateVpcPeeringAuthorization	1	0.1	Non	La limite est fixée par compte.
CreateVpcPeeringConnection	1	0.1	Non	La limite est fixée par compte.
DeleteAlias	4	1	Non	La limite est fixée par compte.
DeleteBuild	10	1	Non	La limite est fixée par compte.
DeleteContainerFleet	4	1	Non	La limite est fixée par compte.
DeleteContainerGroupDefinition	10	1	Non	La limite est fixée par compte.
DeleteFleet	4	1	Non	La limite est fixée par compte.
DeleteFleetLocations	4	1	Non	La limite est fixée par compte.
DeleteGameServerGroup	5	0.1	Non	La limite est fixée par compte.
DeleteGameSessionQueue	4	1	Non	La limite est fixée par compte.
DeleteLocation	4	1	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
DeleteMatchmakingConfiguration	4	1	Non	La limite est fixée par compte.
DeleteMatchmakingRuleSet	4	1	Non	La limite est fixée par compte.
DeleteScalingPolicy	4	1	Non	La limite est fixée par compte.
DeleteScript	10	1	Non	La limite est fixée par compte.
DeleteVpcPeeringAuthorization	1	0.1	Non	La limite est fixée par compte.
DeleteVpcPeeringConnection	1	0.1	Non	La limite est fixée par compte.
DeregisterGameServer	40	20	Non	La limite est fixée par compte.
DeregisterCompute	10	2	Non	La limite est fixée par compte.
DescribeAlias	10	2	Non	La limite est fixée par compte.
DescribeBuild	20	5	Non	La limite est fixée par compte.
DescribeCompute	10	2	Non	La limite est fixée par compte.
DescribeContainerFleet	20	5	Non	La limite est fixée par compte.
DescribeContainerGroupDefinition	20	5	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
Décrivez EC2 InstanceLimits	40	15	Non	La limite est fixée par compte.
DescribeFleetAttributes	10	2	Non	La limite est fixée par compte.
DescribeFleetCapacity	4	1	Non	La limite est fixée par compte.
DescribeFleetDeployment	20	5	Non	La limite est fixée par compte.
DescribeFleetEvents	10	5	Non	La limite est fixée par compte.
DescribeFleetLocationAttributes	10	2	Non	La limite est fixée par compte.
DescribeFleetLocationCapacity	10	5	Non	La limite est fixée par compte.
DescribeFleetLocationUtilization	10	2	Non	La limite est fixée par compte.
DescribeFleetPortSettings	10	2	Non	La limite est fixée par compte.
DescribeFleetUtilization	10	2	Non	La limite est fixée par compte.
DescribeGameServer	40	20	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
DescribeGameServerGroup	10	2	Non	La limite est fixée par compte.
DescribeGameServerInstances	10	2	Non	La limite est fixée par compte.
DescribeGameSessionDetails	10	5	Non	La limite est fixée par compte. Envisagez de l'utiliser à la DescribeGameSessions place, car il a une limite par défaut plus élevée et prend en charge les limites de niveau de ressources.
DescribeGameSessionPlacement	75	75	Oui	La limite est fixée par compte. Sous réserve d'une limitation du niveau des ressources.
DescribeGameSessionQueues	10	2	Non	La limite est fixée par compte.
DescribeGameSessions	20	20	Oui	La limite est fixée par compte. Sous réserve d'une limitation du niveau des ressources.
DescribeInstances	10	2	Non	La limite est fixée par compte.
DescribeMatchmaking	75	75	Oui	La limite est fixée par compte. Sous réserve d'une limitation du niveau des ressources.
DescribeMatchmakingConfigurations	15	7	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
DescribeMatchmakingRuleSets	20	10	Non	La limite est fixée par compte.
DescribePlayerSessions	40	40	Oui	La limite est fixée par compte. Sous réserve d'une limitation du niveau des ressources.
DescribeRuntimeConfiguration	4	1	Non	La limite est fixée par compte.
DescribeScalingPolicies	20	20	Non	La limite est fixée par compte.
DescribeScript	20	5	Non	La limite est fixée par compte.
DescribeVpcPeeringAuthorizations	1	0.1	Non	La limite est fixée par compte.
DescribeVpcPeeringConnections	1	0.1	Non	La limite est fixée par compte.
GetComputeAccess	4	1	Non	La limite est fixée par compte.
GetComputeAuthToken	10	5	Non	La limite est fixée par compte.
GetGameSessionLogUrl	4	1	Non	La limite est fixée par compte.
GetInstanceAccess	4	1	Non	La limite est fixée par compte.
ListAliases	10	2	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
ListBuilds	20	5	Non	La limite est fixée par compte.
ListContainerFleets	20	5	Non	La limite est fixée par compte.
ListCompute	10	2	Non	La limite est fixée par compte.
ListContainerGroup Definitions	20	5	Non	La limite est fixée par compte.
ListContainerGroup DefinitionVersions	20	5	Non	La limite est fixée par compte.
ListFleets	10	2	Non	La limite est fixée par compte.
ListFleetDeployments	20	5	Non	La limite est fixée par compte.
ListGameServerGroups	10	2	Non	La limite est fixée par compte.
ListGameServers	10	2	Non	La limite est fixée par compte.
ListLocations	10	2	Non	La limite est fixée par compte.
ListScripts	20	5	Non	La limite est fixée par compte.
ListTagsForResource	20	5	Non	La limite est fixée par compte.
PutScalingPolicy	4	1	Non	La limite est fixée par compte.
RegisterCompute	10	5	Non	La limite est fixée par compte.
RegisterGameServer	40	20	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
RequestUploadCredentials	4	1	Non	La limite est fixée par compte.
ResolveAlias	4	1	Non	La limite est fixée par compte.
ResumeGameServerGroup	1	0.1	Non	La limite est fixée par compte.
SearchGameSessions	10	5	Non	La limite est fixée par compte.
StartFleetActions	1	0.1	Non	La limite est fixée par compte.
StartGameSessionPlacement	75	75	Non	La limite est fixée par compte.
StartMatchBackfill	70	70	Non	La limite est fixée par compte.
StartMatchmaking	100	100	Non	La limite est fixée par compte.
StopFleetActions	1	0.1	Non	La limite est fixée par compte.
StopGameSessionPlacement	20	20	Non	La limite est fixée par compte.
StopMatchmaking	100	100	Non	La limite est fixée par compte.
SuspendGameServerGroup	1	0.1	Non	La limite est fixée par compte.
TagResource	20	5	Non	La limite est fixée par compte.
TerminateGameSession	10	2	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
UntagResource	20	5	Non	La limite est fixée par compte.
UpdateAlias	4	1	Non	La limite est fixée par compte.
UpdateBuild	10	1	Non	La limite est fixée par compte.
UpdateContainerGroupDefinition	10	1	Non	La limite est fixée par compte.
UpdateContainerFleet	4	1	Non	La limite est fixée par compte.
UpdateFleetAttributes	4	1	Non	La limite est fixée par compte.
UpdateFleetCapacity	4	1	Non	La limite est fixée par compte.
UpdateFleetPortSettings	4	1	Non	La limite est fixée par compte.
UpdateGameServer	40	20	Non	La limite est fixée par compte.
UpdateGameServerGroup	5	0.1	Non	La limite est fixée par compte.
UpdateGameSession	10	2	Non	La limite est fixée par compte.
UpdateGameSessionQueue	4	1	Non	La limite est fixée par compte.
UpdateMatchmakingConfiguration	4	1	Non	La limite est fixée par compte.
UpdateRuntimeConfiguration	4	1	Non	La limite est fixée par compte.

Action d'API	Limite de débit en rafale	Limite de débit	Régulation au niveau des ressources	Remarques
UpdateScript	10	1	Non	La limite est fixée par compte.
ValidateMatchmakingRuleSet	4	1	Non	La limite est fixée par compte.

Régulation au niveau des ressources

Certaines opérations d'API sont soumises à une limitation au niveau des ressources afin d'éviter les problèmes liés aux touches de raccourci de la base de données. Tout en augmentant les limites au niveau de l'API pour ces opérations, elles peuvent toujours être limitées au niveau des ressources.

Les opérations d'API suivantes sont soumises à une limitation au niveau des ressources :

- DescribeGameSessionPlacement
- DescribeGameSessions
- DescribeMatchmaking
- DescribePlayerSessions

La limitation au niveau des ressources est appliquée à l'aide de touches d'accélération spécifiques qui incluent des identificateurs de ressources. Par exemple, DescribeGameSessionPlacement utilise la touche « Operation :GameLift/DescribeGameSessionPlacement, aws-account-placement-id : » qui applique la limitation par compte et par combinaison d'identifiant de placement. Cela évite aux clients de surcharger le système de demandes de ressources spécifiques.

Limites de l'API du SDK Amazon GameLift Servers

Le tableau suivant répertorie les limites de débit par défaut pour les opérations d'API du SDK Amazon GameLift Servers. Ces limites optimisent les performances et évitent la contention des

ressources dans les environnements de serveurs de jeux. Il est important de comprendre ces limites pour une intégration efficace côté serveur avec Amazon GameLift Servers.

Les limites du SDK du serveur sont appliquées à l'aide de clés de régulation spécifiques qui incluent des identifiants de compte et de processus. Certaines limites sont fixées par joueur afin d'éviter les abus de la part de joueurs individuels.

Note

Ce tableau est sujet à modification. La source officielle pour les limites actuelles est la configuration interne du SDC. Contactez le AWS Support pour obtenir le plus up-to-date d'informations si nécessaire.

Référence des limites de l'API du SDK du serveur

Action d'API	Limite de débit en rafale	Limite de débit	clé d'identification	Remarques
AcceptPlayerSession	3	1	aws-account-and-process-id-and-player-id	La limite est fixée par joueur et par processus. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est par joueur.
RemovePlayerSession	3	1	aws-account-and-process-id-and-player-id	La limite est fixée par joueur et par processus. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est par joueur.
ActivateGameSession	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.

Action d'API	Limite de débit en rafale	Limite de débit	touche d'étranglement	Remarques
ActivateHostProcessV2	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
StartMatchBackfill	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
DescribePlayerSessions	10	5	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
GetComputeCertificate	3	0.1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
GetCustomRoleCredentials	3	0.1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.

Action d'API	Limite de débit en rafale	Limite de débit	touche d'étranglement	Remarques
InitSDK	3	0.1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
RefreshHostProcess	3	0.1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
StopMatchmaking	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
ProcessEnding	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
ProcessReady	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.

Action d'API	Limite de débit en rafale	Limite de débit	touche d'étranglement	Remarques
UpdateGameSession	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
UpdateHostProcess	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
GetGameSessionId	10	5	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
GetTerminationTime	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.
UpdatePlayerSessionCreationPolicy	3	1	aws-account-and-process-identifiant	La limite est fixée par processus serveur. Il ne devrait jamais être nécessaire d'augmenter la limite, car c'est un processus par processus.

 Note

Il s'agit des limites par défaut. Si vous rencontrez un ralentissement sur ces derniers APIs, cela peut être dû à un ralentissement au niveau du compte. Pour demander une augmentation de limite, contactez le AWS Support.

Les limites par processus et par joueur sont conçues pour s'adapter aux habitudes d'utilisation normales et devraient rarement nécessiter un ajustement. Si vous êtes confronté à un problème de limitation APIs, passez en revue votre mise en œuvre pour détecter d'éventuelles opportunités d'optimisation avant de demander des augmentations de limites. La limitation au niveau du compte peut également s'appliquer par le biais de ces règles supplémentaires :

- SdkWebSocket/AccountLevelHighUsage, compte AWS
- SdkWebSocket/AccountLevelThrottle, compte AWS

Amazon GameLift Servers Balises ping UDP

Les balises ping UDP permettent de mesurer la latence du réseau entre les appareils des joueurs et les Amazon GameLift Servers sites d'hébergement. Avec les balises ping, vous pouvez collecter des données de latence précises pour prendre des décisions éclairées concernant le placement des serveurs de jeu et améliorer le matchmaking des joueurs en fonction des exigences de latence.

Comment fonctionnent les balises ping UDP

Amazon GameLift Servers fournit des points de terminaison UDP fixes (balises ping) sur chaque site d'hébergement où vous pouvez déployer des serveurs de jeu. Comme la plupart des serveurs de jeux communiquent via UDP, la mesure de la latence à l'aide de balises ping UDP fournit des résultats plus précis qu'à l'aide de pings ICMP. Les périphériques réseau gèrent souvent les paquets ICMP différemment des paquets UDP, ce qui peut entraîner des mesures de latence qui ne reflètent pas les performances réelles de vos joueurs.

Grâce aux balises ping UDP, votre client de jeu peut envoyer des messages UDP à ces points de terminaison et recevoir des réponses asynchrones, vous fournissant ainsi des mesures de latence qui reflètent mieux les conditions de trafic réelles du jeu entre l'appareil du joueur et les sites d'hébergement potentiels. Les points de terminaison sont permanents et restent disponibles tant qu'ils sont compatibles Amazon GameLift Servers avec l'hébergement de jeux sur ce site.

Cas d'utilisation courants des balises ping UDP

Vous pouvez utiliser les balises ping UDP de différentes manières pour optimiser l'expérience réseau de votre jeu.

Choisir des emplacements d'hébergement optimaux

Collectez des données de latence dans différentes régions géographiques afin d'identifier les meilleurs emplacements principaux et secondaires pour héberger des serveurs de jeu pour votre base de joueurs.

Placer les sessions de jeu en fonction de la latence des joueurs

Incluez les données de latence des joueurs lorsque vous demandez de nouvelles sessions de jeu pour vous aider à choisir les emplacements offrant l'expérience la plus faible en termes de latence.

Optimisation du matchmaking en fonction de la latence

Fournissez des données de latence des joueurs lors de la demande de matchmaking pour aider à faire correspondre des joueurs ayant des profils de latence similaires et à placer les sessions de jeu dans des emplacements optimaux pour les joueurs correspondants.

Note

Lorsque vous créez des demandes de matchmaking, vous ne devez pas fournir d'informations de latence aux sites où vous n'avez pas de flottes. Si c'est le cas, vous Amazon GameLift Servers pourriez essayer de placer la session de jeu dans des endroits où la flotte est insuffisante, ce qui entraînera l'échec des demandes de matchmaking.

Obtenir les points de terminaison des balises

Pour récupérer les informations de domaine et de port de la balise ping pour les Amazon GameLift Servers emplacements, utilisez l'opération [ListLocations](#) API. L'ensemble des emplacements renvoyés par cette API dépend de celui Région AWS que vous spécifiez lorsque vous l'appellez (ou de votre région par défaut si vous n'en spécifiez pas). Lorsque vous appelez depuis :

- Région d'origine d'une flotte prenant en charge les sites multiples : l'API renvoie des informations pour tous les sites d'hébergement

- Région d'origine d'une flotte qui prend en charge un seul emplacement : l'API renvoie des informations pour cet emplacement

Notez que si vous appelez cette API en utilisant un emplacement qui ne peut être qu'un emplacement distant dans un parc de plusieurs sites, l'API renverra une erreur car ce type de site ne possède pas de point de terminaison de service.

Consultez le tableau des sites pris en charge [AWS Emplacements pris en charge](#) pour identifier les régions d'origine qui prennent en charge les flottes à sites uniques ou multiples.

Exemple

```
aws gamelift list-locations --region ap-northeast-2
```

Cela Région AWS prend en charge les flottes multi-sites, donc plusieurs sites seront renvoyés. Voici un exemple de l'une des valeurs renvoyées :

```
[...]
{
  "LocationName": "ap-northeast-1",
  "PingBeacon": {
    "UDPEndpoint": {
      "Domain": "gamelift-ping.ap-northeast-1.api.aws",
      "Port": 7770
    }
  }
}
```

Important

Mettez en cache les informations de la balise ping plutôt que d'appeler `ListLocations` avant chaque mesure de latence. Les informations de domaine et de port sont statiques et l'API n'est pas conçue pour les demandes volumineuses.

Implémentation de mesures de

Suivez ces bonnes pratiques lors de la mise en œuvre de mesures de latence à l'aide de balises ping UDP :

1. Stockez les informations de la balise ping en utilisant l'une des approches suivantes :

- Codez en dur les points de terminaison dans votre client de jeu.
- Mettez les informations en cache dans le backend de votre jeu.
- Mettez en œuvre un mécanisme de mise à jour périodique (journalière/hebdomadaire) pour actualiser les informations.

2. Envoyer des messages ping UDP :

- Mettez ce que vous voulez dans le corps du message, à condition qu'il ne soit pas vide, et que la taille maximale des messages soit inférieure à 300 octets.
- Respectez les limites de débit suivantes pour chaque point de vente :
 - 3 transactions par seconde (TPS) par adresse IP unique de l'expéditeur et combinaison de ports
 - 1000 TPS par adresse IP d'expéditeur unique

3. Calculez la latence :

- Envoyez plusieurs pings à chaque emplacement pour calculer une latence moyenne.
- Envisagez d'envoyer des pings simultanés à plusieurs emplacements pour obtenir des résultats plus rapides.
- Utilisez la logique de nouvelle tentative selon les besoins pour envoyer de nouveaux paquets pour tout paquet qui n'a pas été renvoyé dans un court laps de temps (généralement 1 à 3 secondes), car le protocole UDP n'offre pas une garantie de livraison à 100 %.
- Calculez le décalage horaire entre l'envoi d'un message et la réception de la réponse.
- Si une grande partie des pings UDP envoyés à un emplacement ne reçoivent toujours pas de réponse, calculez le temps de latence en utilisant les pings ICMP adressés à nos points de [terminaison de Amazon GameLift Servers service](#) standard comme solution de rechange.

Tip

Nous vous recommandons d'inclure le port 7770 partout où vous documentez la liste des ports que vos joueurs doivent avoir ouverts sur leur réseau local. C'est une autre raison pour laquelle vous devriez avoir une solution de rechange pour mesurer la latence (en utilisant ICMP, par exemple) au cas où ce port serait bloqué.

Exemples de code

Voici quelques exemples simples montrant comment envoyer des pings UDP et calculer le temps de latence.

C++

```
#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <chrono>

int main() {
    // Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    const char* domain = "gamelift-ping.ap-south-1.api.aws";
    const int port = 7770;
    const char* message = "Ping";           // Your message
    const int num_pings = 3;                 // Number of pings to send

    // Create socket
    int sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
        std::cerr << "Error creating socket" << std::endl;
        return 1;
    }

    // Resolve domain name to IP address
    struct hostent* host = gethostbyname(domain);
    if (host == nullptr) {
        std::cerr << "Error resolving hostname" << std::endl;
        close(sock);
        return 1;
    }

    // Set up the server address structure
    struct sockaddr_in server_addr;
    std::memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
```

```
server_addr.sin_port = htons(port);
std::memcpy(&server_addr.sin_addr, host->h_addr, host->h_length);

// Set socket timeout
struct timeval tv;
tv.tv_sec = 1; // 1 second timeout
tv.tv_usec = 0;
if (setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv)) < 0) {
    std::cerr << "Error setting socket timeout" << std::endl;
    close(sock);
    return 1;
}

double total_latency = 0;
int successful_pings = 0;

for (int i = 0; i < num_pings; ++i) {
    auto start = std::chrono::high_resolution_clock::now();

    // Send the message
    ssize_t bytes_sent = sendto(sock, message, std::strlen(message), 0,
                                (struct sockaddr*)&server_addr,
sizeof(server_addr));

    if (bytes_sent < 0) {
        std::cerr << "Error sending message" << std::endl;
        continue;
    }

    // Receive response
    char buffer[1024];
    socklen_t server_addr_len = sizeof(server_addr);
    ssize_t bytes_received = recvfrom(sock, buffer, sizeof(buffer), 0,
                                       (struct sockaddr*)&server_addr,
&server_addr_len);

    auto end = std::chrono::high_resolution_clock::now();

    if (bytes_received < 0) {
        std::cerr << "Error receiving response or timeout" << std::endl;
    } else {
        std::chrono::duration<double, std::milli> latency = end - start;
        total_latency += latency.count();
        successful_pings++;
    }
}
```

```

        std::cout << "Received response, latency: " << latency.count() << " ms"
<< std::endl;
    }

    // Wait a bit before next ping
    usleep(1000000); // 1s
}

// Close the socket
close(sock);

if (successful_pings > 0) {
    double avg_latency = total_latency / successful_pings;
    std::cout << "Average latency: " << avg_latency << " ms" << std::endl;
} else {
    std::cout << "No successful pings" << std::endl;
}

return 0;
}

```

C#

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Diagnostics;
using System.Threading.Tasks;

class UdpLatencyTest
{
    static async Task Main()
    {
        // Replace with Amazon GameLift Servers UDP ping beacon domain for your
        // desired location
        string domain = "gamelift-ping.ap-south-1.api.aws";
        int port = 7770;
        string message = "Ping";           // Your message
        int numPings = 3;                  // Number of pings to send
        int timeoutMs = 1000;              // Timeout in milliseconds
    }
}

```

```
        await MeasureLatency(domain, port, message, numPings, timeoutMs);
    }

    static async Task MeasureLatency(string domain, int port, string message, int
numPings, int timeoutMs)
    {
        using (var udpClient = new UdpClient())
        {
            try
            {
                // Resolve domain name to IP address
                IPAddress[] addresses = await Dns.GetHostAddressesAsync(domain);
                if (addresses.Length == 0)
                {
                    Console.WriteLine("Could not resolve domain name.");
                    return;
                }

                IPEndPoint endPoint = new IPEndPoint(addresses[0], port);
                byte[] messageBytes = Encoding.UTF8.GetBytes(message);

                // Set receive timeout
                udpClient.Client.ReceiveTimeout = timeoutMs;

                double totalLatency = 0;
                int successfulPings = 0;
                var stopwatch = new Stopwatch();

                for (int i = 0; i < numPings; i++)
                {
                    try
                    {
                        stopwatch.Restart();

                        // Send message
                        await udpClient.SendAsync(messageBytes, messageBytes.Length,
endPoint);

                        // Wait for response
                        UdpReceiveResult result = await
ReceiveWithTimeoutAsync(udpClient, timeoutMs);

                        stopwatch.Stop();
                        double latency = stopwatch.Elapsed.TotalMilliseconds;
```

```

        totalLatency += latency;
        successfulPings++;

        string response = Encoding.UTF8.GetString(result.Buffer);
        Console.WriteLine($"Ping {i + 1}: {latency:F2}ms - Response:
{response}");
    }
    catch (SocketException ex)
    {
        Console.WriteLine($"Ping {i + 1}: Failed - {ex.Message}");
    }
    catch (TimeoutException)
    {
        Console.WriteLine($"Ping {i + 1}: Timeout");
    }

    // Wait before next ping
    await Task.Delay(1000); // 1s between pings
}

if (successfulPings > 0)
{
    double averageLatency = totalLatency / successfulPings;
    Console.WriteLine($"\\nSummary:");
    Console.WriteLine($"Successful pings: {successfulPings}/
{numPings}");
    Console.WriteLine($"Average latency: {averageLatency:F2}ms");
}
else
{
    Console.WriteLine($"\\nNo successful pings");
}
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}

static async Task<UdpReceiveResult> ReceiveWithTimeoutAsync(UdpClient client,
int timeoutMs)
{

```

```
        using var cts = new System.Threading.CancellationTokenSource(timeoutMs);
        try
        {
            return await client.ReceiveAsync().WaitAsync(cts.Token);
        }
        catch (OperationCanceledException)
        {
            throw new TimeoutException("Receive operation timed out");
        }
    }
}
```

Python

```
import socket
import time
import statistics
from datetime import datetime

def udp_ping(host, port, timeout=2):
    """
    Send a UDP ping and return the round trip time in milliseconds.
    Returns None if timeout occurs.
    """
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(timeout)
    message = b'ping'

    try:
        # Resolve hostname first
        try:
            socket.gethostbyname(host)
        except socket.gaierror as e:
            print(f"Could not resolve hostname: {e}")
            return None

        start_time = time.time()
        sock.sendto(message, (host, port))

        # Wait for response
        data, server = sock.recvfrom(1024)
        end_time = time.time()
```



```

        # Calculate round trip time in milliseconds
        rtt = (end_time - start_time) * 1000
        return rtt

    except socket.timeout:
        print(f"Request timed out")
        return None
    except Exception as e:
        print(f"Error: {type(e).__name__}: {e}")
        return None
    finally:
        sock.close()

def main():
    # Replace with Amazon GameLift Servers UDP ping beacon domain for your desired
    location
    host = "gamelift-ping.ap-south-1.api.aws"
    port = 7770
    num_pings = 3
    latencies = []

    print(f"\nPinging {host}:{port} {num_pings} times...")
    print(f"Start time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")

    for i in range(num_pings):
        print(f"Ping {i+1}:")
        rtt = udp_ping(host, port)

        if rtt is not None:
            print(f"Response from {host}: time={rtt:.2f}ms")
            latencies.append(rtt)

        # Wait 1 second between pings
        if i < num_pings - 1:
            time.sleep(1)
        print()

    # Calculate and display statistics
    print("-" * 50)
    print(f"Ping statistics for {host}:")
    print(f"    Packets: Sent = {num_pings}, Received = {len(latencies)}, "
          f"Lost = {num_pings - len(latencies)} "
          f"f'({((num_pings - len(latencies)) / num_pings * 100):.1f}% loss)")

```

```
    if latencies:
        print("\nRound-trip latency statistics:")
        print(f"    Minimum = {min(latencies):.2f}ms")
        print(f"    Maximum = {max(latencies):.2f}ms")
        print(f"    Average = {statistics.mean(latencies):.2f}ms")

    print(f"\nEnd time: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

if __name__ == "__main__":
    main()
```

Notes de mise à jour pour Amazon GameLift Servers

Les notes de mise à jour Amazon GameLift Servers fournissent des détails sur les nouvelles fonctions, les nouvelles mises à jour et les nouveaux correctifs concernant le service.

Versions SDK

Les tableaux suivants répertorient toutes les Amazon GameLift Servers versions contenant des informations sur la version du SDK. Il n'est pas nécessaire d'utiliser un équivalent SDKs pour les intégrations de votre serveur de jeu et de votre client. Toutefois, les versions antérieures d'un SDK peuvent ne pas prendre entièrement en charge les dernières fonctionnalités d'un autre SDK.

Amazon GameLift Servers SDKs et les plugins sont open source. Consultez [Obtenez des outils Amazon GameLift Servers de développement](#). Pour obtenir les dernières versions, adressez-vous à l'[Amazon GameLift Servers GitHub organisation](#).

Version actuelle

1996
 1997
 1998
 1999
 2000
 2001
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 2024
 2025
 2026
 2027
 2028
 2029
 2030

Versions précédentes

Womens' rights

© 2008 Kur

subinstruments

client

Préface

⊕#

 $+$

Unreal

[illegible]

libu

Version

Intérieur

13

533000352

ou

version

ultérieur

2

520000

NET

3) Version

Intérieur

12

54000362

ou

version

ultérieur

2

534000852

100

Amazon GameLift
SDK
clients

Unreal
client

version
ultérieur

version
ultérieur

version
ultérieur

version
ultérieur

version
ultérieur

Amazon GameLift
SDK
clients

Unreal

Unreal Engine 4.26.0

Unreal Engine 4.26.0

Unreal Engine 4.26.0

Unreal Engine 4.26.0

Amazon GameLift
SDK
clients

Unreal

5.0.0.002
ou
version
ultérieur

5.0.0.002
ou
version
ultérieur

5.0.0.002
ou
version
ultérieur

5.0.0.002
ou
version
ultérieur

Amazon GameLift
SDK
clients
client

SDK
SDK
+
Unreal

[5.0.0.002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[5.0.0.002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[5.0.0.002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[5.0.0.002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

Amazon GameLift
SDK
clients

Unreal

5.12.0002
ou
version
ultérieur

5.12.0002
ou
version
ultérieur

5.12.0002
ou
version
ultérieur

5.12.0001
ou
version
ultérieur

Amazon GameLift
SDK
clients

SDK

+
Unreal

[3.0.1200](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1200-2](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1202](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1202](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

Amazon GameLift
SDK
clients
client

SDK

+
Unreal

[3.0.1002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

[3.0.1002](#)
[ou](#)
[version](#)
[ultérieur](#)
[e](#)

BirthtimeS

Wiederkehr

Techniques

client

Proté

Ⓢ#

+

Unreal

30320

novembre

Session

Intérieur

30812002

bu

version

Intérieur

3431002

ou

version

Intérieur

3

32310-1

U

version

Intérieur

Amazon GameLift
SDK
clients
client

SDK
SDK
+
Unreal

[Amazon GameLift SDK-0](#)
ou
version
ultérieur
e

[Amazon GameLift SDK-1](#)
ou
version
ultérieur
e

[Amazon GameLift SDK-1](#)
ou
version
ultérieur
e

[Amazon GameLift SDK-2](#)
ou
version
ultérieur
e

Amazon GameLift
SDK
clients
client

SDK
SDK
+
Unreal

[3.7.1009-0](#)
ou
version
ultérieur
e

[3.7.1007-0](#)
ou
version
ultérieur
e

[3.7.1001](#)
ou
version
ultérieur
e

[3.7.1003-0](#)
ou
version
ultérieur
e

Amazon GameLift
SDK
clients

SDK

+
Unreal

[3.3.0-0](#)
ou
version
ultérieur

[3.3.0-201](#)
ou
version
ultérieur

[3.2.10-2](#)
ou
version
ultérieur

[3.2.10-1](#)
ou
version
ultérieur

Amazon GameLift
SDK
clients
client

SDK

+
Unreal

[3.2.105-1](#)
ou
version
ultérieur
e

[3.2.1201](#)
ou
version
ultérieur
e

[3.2.1201](#)
ou
version
ultérieur
e

[3.1.1201](#)
ou
version
ultérieur
e

Amazon GameLift
SDK
clients

Unreal

3.17.201

3.17.201

3.17.201

3.17.201

Amazon GameLift
SDK
clients
client

SDK

+
Unreal

[3.13.0 1-1](#)
ou
version
ultérieur
e

[3.13.0 10-1](#)
ou
version
ultérieur
e

[3.13.0 201](#)
ou
version
ultérieur
e

[3.13.0 201](#)
ou
version
ultérieur
e

Notes de mise à jour

Les notes de mise à jour suivantes sont indiquées dans l'ordre chronologique. Les dernières mises à jour sont répertoriées en premier. La première version de Amazon GameLift Servers date de 2016. Pour les notes de mise à jour datées plus tôt que celles répertoriées ici, consultez les liens de date de publication dans [Versions SDK](#).

12 août 2025 : le Amazon GameLift Servers plugin pour Unreal Engine prend désormais en charge la version 5.6 de l'UE

Version du plugin mise à jour :

- [Plugin pour Unreal](#), version 3.0.1 (inclut le SDK du serveur pour Unreal, version 5.3.1)

Amazon GameLift Servers publie de nouvelles versions du plugin pour Unreal Engine et du SDK du serveur pour Unreal Engine. Ces dernières versions incluent les mises à jour suivantes :

- Ils sont désormais compatibles avec Unreal Engine 5.6.
- Les problèmes de compilation lors de l'empaquetage des cibles clientes Android sont résolus.

En savoir plus :

- [Intégration de jeux avec le Amazon GameLift Servers plugin pour Unreal Engine, Guide du Amazon GameLift Servers](#) développeur
- [Intégration Amazon GameLift Servers dans un projet Unreal Engine, guide du Amazon GameLift Servers](#) développeur

24 juin 2025 : Amazon GameLift Servers lance des balises ping UDP pour aider à mesurer la latence du réseau en temps réel pour les jeux multijoueurs

Versions du SDK mises à jour :

- KIT DE DÉVELOPPEMENT LOGICIEL AWS 1.11.595

Amazon GameLift Servers publie pour une mise à disposition générale un ensemble de points de terminaison fixes appelés balises ping UDP pour vous aider à mesurer avec précision le temps de latence entre les appareils des joueurs et l'emplacement des serveurs de jeu. Les points de

terminaison des balises ping UDP sont disponibles dans toutes les régions AWS mondiales et zones locales prises en charge par Amazon GameLift Servers, à l'exception des régions de AWS Chine.

La plupart des jeux multijoueurs utilisent le protocole UDP (User Datagram Protocol) comme principal protocole de transmission de paquets en raison de ses avantages en termes de performances pour les jeux en temps réel. Comprendre et optimiser la latence du réseau est essentiel pour offrir la meilleure expérience possible aux joueurs. Les balises ping UDP constituent un moyen cohérent et fiable de mesurer la latence réelle des paquets UDP entre les joueurs et les serveurs de jeu, vous aidant ainsi à prendre de meilleures décisions concernant la player-to-server correspondance et le placement des sessions de jeu.

L'[ListLocations](#) API a été étendue pour inclure des informations sur le domaine et le port des points de terminaison dans la liste des emplacements qu'elle renvoie, ce qui facilite l'accès par programmation aux points de terminaison.

Votre client de jeu peut envoyer des messages UDP à ces points de terminaison et recevoir des réponses asynchrones avec les mêmes données, ce qui vous donne des mesures de latence qui reflètent mieux les conditions de trafic réelles du jeu entre l'appareil d'un joueur et les sites d'hébergement potentiels. Ces points de terminaison sont permanents et restent disponibles tant qu'ils sont compatibles Amazon GameLift Servers avec l'hébergement de jeux sur ce site.

En savoir plus :

- [Balises ping UDP](#), Guide Amazon GameLift Servers du développeur
- [ListLocations](#), Référence Amazon GameLift Servers d'API

24 juin 2025 : Amazon GameLift Servers ajout de la prise en charge de la localisation pour la Thaïlande et la Malaisie

Grâce Amazon GameLift Servers à l'hébergement géré, vous pouvez désormais déployer des ressources de serveurs de jeux à Bangkok, en Thaïlande, et à Kuala Lumpur, en Malaisie, pour étendre la portée de vos jeux aux joueurs de toute l'Asie du Sud-Est. Ces nouvelles régions contribuent à réduire le temps de latence et à améliorer l'expérience de jeu des joueurs de ces zones.

Les sites suivants Régions AWS sont disponibles en tant que sites distants pour les flottes multisites. Pour commencer à organiser des sessions de jeu dans ces lieux, ajoutez-les en tant que sites distants à une flotte multi-sites nouvelle ou existante. Avec les flottes multisites, vous pouvez gérer directement la capacité d'hébergement de chaque site.

- Asie-Pacifique (Thaïlande) ap-southeast-7 ()
- Asie-Pacifique (Malaisie) ap-southeast-5 ()

Ils ne Régions AWS sont pas activés par défaut pour un AWS compte. Vous devez vous inscrire dans chaque région avant de pouvoir y déployer Amazon GameLift Servers des ressources.

En savoir plus :

- [Amazon GameLift Serverspoints de service](#), Guide du développeur Amazon GameLift Servers d'hébergement
- [Mettre à jour les emplacements du parc](#), Guide du développeur Amazon GameLift Servers d'hébergement
- [Activez ou désactivez AWS les régions dans votre AWS compte](#), Guide de référence sur la gestion des comptes.

29 mai 2025 : Amazon GameLift Servers sortie de nouvelles versions du SDK du serveur et des plugins pour Unreal Engine et Unity

Versions du SDK mises à jour :

- [SDK du serveur C++](#), version 5.3.0
- [SDK du serveur C#, version 5.3.0](#)
- [SDK du serveur Go](#), version 5.3.0
- [SDK de serveur pour Unreal, version 5.3.0](#)
- [SDK serveur pour](#) Unity, version 5.3.0

Versions du plugin mises à jour :

- [Plugin pour Unreal](#), version 3.0.0
- [Plug-in pour Unity](#), version 3.1.0

Les nouvelles versions du SDK du serveur pour C++, C#, Go, Unreal et Unity, ainsi que les nouvelles versions du plugin pour Unreal Engine et Unity, sont désormais open source. Ils sont tous disponibles dans l'[Amazon GameLift Servers GitHub organisation](#). Pour obtenir des informations détaillées sur les mises à jour, consultez les notes de publication et les fichiers readme de chaque référentiel.

Principales mises à jour du SDK du serveur :

- Validation côté client et réponses aux erreurs améliorées sur tous les serveurs. SDKs
- La fonction de [OnProcessTerminate](#) rappelle possède désormais une logique par défaut pour mettre fin au processus du serveur de jeu.
- La fonction utilise [InitSDK\(\)](#) désormais un jeton d'idempotence pour prendre en charge plusieurs tentatives.
- Le [OnUpdateGameSession](#) rappelle peut désormais transmettre des données pour `autoBackfillMode`.

Principales mises à jour du plugin :

- Le plugin pour Unreal Engine dispose désormais d'un processus d'installation et de configuration plus rationalisé, avec plus d'automatisation et moins de prérequis (CMakeOpenSSL et la chaîne d'outils de compilation croisée Unreal).
- Le plugin pour Unreal Engine a amélioré l'interface utilisateur du EC2 flux de travail géré, notamment en prenant en charge les espaces dans les chemins de construction du client et du serveur du jeu. En outre, vous pouvez désormais ajouter des arguments de ligne de commande lors du lancement d'un client de jeu depuis l'éditeur.
- Le plugin pour Unreal Engine prend désormais en charge les versions intégrées au UE5 serveur ARM.

En savoir plus :

- [Amazon GameLift Servers plugin pour Unreal Engine](#), Guide Amazon GameLift Servers du développeur
- [Amazon GameLift Servers plugin pour Unity \(SDK serveur 5.x\)](#), Guide Amazon GameLift Servers du développeur
- [SDK de serveur 5.x pour Amazon GameLift Servers](#), Guide Amazon GameLift Servers du développeur
- [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#), Guide Amazon GameLift Servers du développeur
- [Intégrer Amazon GameLift Servers dans un projet Unity](#), Guide Amazon GameLift Servers du développeur

24 avril 2025 : le SDK 5 du serveur C# prend Amazon GameLift Servers actuellement en charge .NET 8

Versions du SDK mises à jour :

- SDK du serveur C#, version 5.2.1

Pour les développeurs de jeux utilisant C#, vous pouvez désormais utiliser .NET 8 comme framework cible pour vos Amazon GameLift Servers projets. Avec .NET 8, vous pouvez tirer parti des améliorations de performances, notamment une compilation améliorée just-in-time (JIT), une optimisation de l'utilisation de la mémoire et des temps de démarrage plus rapides. Si vous utilisez actuellement .NET 6, nous vous recommandons de planifier une migration vers .NET 8, notamment en mettant à jour le SDK de votre serveur C# vers la dernière version. Microsoft a annoncé le support de .NET 8, avec des correctifs de sécurité et des mises à jour techniques, jusqu'en novembre 2026.

[Téléchargez la dernière version du SDK du serveur C# pour Amazon GameLift Servers](#)

En savoir plus :

- [Obtenez des outils Amazon GameLift Servers de développement](#), Guide Amazon GameLift Servers du développeur

27 mars 2025 : Amazon GameLift Servers étend la prise en charge des familles d' EC2 instances de nouvelle génération

Versions du SDK mises à jour :

- AWS SDK 1.11.535

Vous pouvez désormais affiner l'hébergement de votre serveur de jeu en Amazon GameLift Servers choisissant parmi une plus grande sélection d' EC2 instances Amazon appartenant à des familles d'instances de 5e à 8e génération. Chaque nouvelle EC2 génération offre des avancées en matière de EC2 calcul, de mémoire et de mise en réseau, les instances de 8e génération fournissant des instances de pointe basées AWS sur Graviton4 et Intel Xeon. Les instances de nouvelle génération sont disponibles dans les familles d'instances suivantes :

- [Usage général](#) (série M)
- [Optimisé pour le calcul](#) (série C)

- [Mémoire optimisée](#) (série R)

Vous pouvez également choisir des variantes offrant un stockage local (d), une mise en réseau améliorée (n) et des architectures de processeur spécifiques (Intel/AMD/Graviton – i/a/g). Les instances de nouvelle génération sont disponibles dans toutes Régions AWS les régions prises en charge Amazon GameLift Servers, à l'exception des régions de AWS Chine. Pour en savoir plus, consultez [Amazon GameLift Servers points de service](#).

Utilisez ces nouveaux types d'instances avec des EC2 flottes Amazon GameLift Servers gérées et des flottes de conteneurs gérées. Lorsque vous remplacez votre hébergement de jeu existant par un nouveau type d'instance (même architecture), il vous suffit de déployer de nouvelles flottes en conservant tous les paramètres de configuration inchangés, à l'exception du type d'instance.

En savoir plus :

- [Amazon GameLift Servers Tarification des instances](#)
- [CreateFleet Amazon GameLift Servers](#) Référence d'API
- [CreateContainerFleet](#), Référence Amazon GameLift Servers d'API

11 mars 2025 : Amazon GameLift Servers lance le nouveau Game Server Wrapper pour une intégration rapide

Le nouveau wrapper de serveur de jeu réduit Amazon GameLift Servers considérablement le temps nécessaire pour héberger un serveur de jeu sur Amazon GameLift Servers. Aucune modification de code n'étant requise, vous pouvez utiliser le wrapper pour ajouter des fonctionnalités de gestion de session de jeu de base à votre jeu et les déployer sur une flotte Amazon GameLift Servers Anywhere, une EC2 flotte gérée ou une flotte de conteneurs gérés. Cet outil est idéal pour effectuer une évaluation pratique des Amazon GameLift Servers fonctionnalités, à l'aide de votre propre serveur de jeu ou d'un exemple de jeu. Il est également utile pour déployer rapidement des itérations de serveurs de jeu, par exemple pour le prototypage ou les tests rapides.

Grâce aux fonctionnalités de base de gestion des sessions de jeu, votre serveur de jeu peut établir une connexion avec le Amazon GameLift Servers service, répondre aux demandes de démarrage et d'arrêt de sessions de jeu, et s'arrêter lorsqu'une session de jeu est terminée.

[Téléchargez le wrapper du serveur de jeu depuis GitHub](#). Consultez le fichier readme GitHub pour savoir comment installer et utiliser le wrapper avec vos projets de jeu.

6 mars 2025 : Amazon GameLift c'est maintenant Amazon GameLift Servers et Amazon GameLift Streams

Amazon GameLift est un service entièrement géré destiné à aider les développeurs à créer, à faire évoluer et à proposer les jeux les plus exigeants au monde. Avec la mise à disposition générale d'Amazon GameLift Streams, Amazon propose GameLift désormais à la fois un serveur de jeu évolutif et des fonctionnalités de streaming fluides.

[Amazon GameLift Servers](#) donne aux développeurs de jeux la capacité de déployer, d'exploiter et de faire évoluer des serveurs de jeux dédiés. Vous pouvez déployer des serveurs de jeu à hautes performances dans le cloud en quelques minutes et les adapter à la hausse ou à la baisse pour répondre à la demande des joueurs. Construit sur un environnement informatique AWS éprouvé, il Amazon GameLift Servers prend en charge 100 millions de joueurs simultanés dans une seule partie, 100 000 joueurs ajoutés par seconde et 9 000 nouvelles instances de calcul par minute. Et avec une sécurité de niveau professionnel, un matchmaking pour le plus grand nombre et pay-as-you-go une flexibilité, il vous aide à démarrer, que vous travailliez sur une nouvelle idée de jeu ou que vous dirigiez un jeu avec des millions de joueurs.

[Amazon GameLift Streams](#) aide les développeurs de jeux à proposer des expériences de streaming de jeux à une résolution allant jusqu'à 1080p et à 60 frames-per-second (images par seconde) sans aucun décalage perceptible sur des appareils tels que iOS, Android, FireOS et PCs pour les joueurs. Grâce à une AWS offre unique, les éditeurs peuvent déployer leur contenu de jeu en quelques minutes, sans modification, sur des instances GPU entièrement gérées dans le cloud et le diffuser via le AWS Network Backbone directement sur n'importe quel appareil du joueur final doté d'un navigateur Web. Les joueurs peuvent commencer à jouer en quelques secondes, sans attendre le téléchargement ou l'installation, et cela offre une expérience de jeu presque identique à celle d'un jeu local sur un PC ou une console de jeu.

14 janvier 2025 : Amazon GameLift Servers sortie d'un support étendu pour la priorisation du placement des sessions de jeu

Versions du SDK mises à jour :

- AWS SDK 1.11.485

En réponse aux commentaires des clients, nous publions une nouvelle fonctionnalité qui vous permet de hiérarchiser les emplacements pour les demandes de placement individuelles lors de sessions de jeu. Pour vos files d'attente configurées pour hiérarchiser le placement par emplacement, vous

pouvez désormais fournir une liste personnalisée d'emplacements prioritaires avec chaque demande de placement.

Cette nouvelle fonctionnalité permet aux clients de modifier dynamiquement les priorités de localisation pour chaque demande de placement selon leurs besoins. Cette flexibilité supplémentaire vous permet de mieux répondre à l'évolution des conditions, telles que la localisation des joueurs, la charge de la flotte ou l'état des serveurs. Il peut également aider les clients qui souhaitent personnaliser davantage la manière dont les lieux de placement sont sélectionnés.

En savoir plus :

- [Prioriser le placement des sessions de jeu](#), guide Amazon GameLift Servers du développeur
- [StartGameSessionPlacement](#), Référence Amazon GameLift Servers d'API

2 janvier 2025 : Amazon GameLift Servers lance un nouveau support pour mettre fin aux sessions de jeu

Versions du SDK mises à jour :

- AWS SDK 1.11.477

En réponse aux commentaires des clients, nous publions une nouvelle fonctionnalité qui vous permet de terminer plus facilement des sessions de jeu individuelles. Avec cette version, vous pouvez désormais mettre fin à une session de jeu directement dans la Amazon GameLift Servers console ou en utilisant le AWS CLI AWS SDK pour Amazon GameLift Servers.

Cette nouvelle fonctionnalité répond à la nécessité de résoudre les sessions de jeu qui restent actives mais en mauvais état, ce qui empêche les ressources informatiques d'héberger de nouvelles sessions de jeu. Auparavant, les clients devaient accéder à distance à l'ordinateur pour terminer manuellement une session de jeu.

Vous avez le choix entre deux méthodes de résiliation. La première méthode tente de mettre fin à une session de jeu en utilisant sa séquence d'arrêt personnalisée, qui peut inclure des actions pour avertir les joueurs et résoudre les données du jeu. La seconde méthode force le processus du serveur à s'arrêter, ce qui met immédiatement fin à la session de jeu. Cette seconde méthode garantit que la session de jeu se termine même lorsque le processus du serveur ne répond pas.

En savoir plus :

- [Arrêter une session de jeu à l'aide de la Amazon GameLift Servers console](#), Guide Amazon GameLift Servers du développeur
- [TerminateGameSession](#), Référence Amazon GameLift Servers d'API

19 décembre 2024 : Amazon GameLift Servers lance le support du plugin de moteur de jeu pour Managed Containers

Versions du plugin mises à jour :

Amazon GameLift Serversplugin pour Unreal Engine, version 2.0.0

- Mise à niveau pour prendre en charge le SDK 5.2.0 du serveur C++ avec prise en charge des conteneurs gérés.
- Ajout du support pour Unreal Engine 5.4 et 5.5.

Amazon GameLift Serversplugin pour Unity, version 3.0.0

- Mise à niveau pour prendre en charge le SDK 5.2.0 du serveur C++ avec prise en charge des conteneurs gérés.
- Support pour Unity 2021.3 LTS et 2022.3 LTS pour Windows et Mac OS.

Le Amazon GameLift Servers plugin pour les moteurs de jeu Unreal et Unity fournit des outils et des flux de travail qui simplifient les étapes de mise en service d'un jeu. Amazon GameLift Servers Amazon GameLift Serverest un service d'hébergement cloud entièrement géré que les développeurs de jeux peuvent utiliser pour gérer et faire évoluer des serveurs de jeu dédiés aux jeux multijoueurs basés sur des sessions.

Les dernières versions du plugin offrent les améliorations suivantes :

- Flux de travail guidé pour l'hébergement avec Managed Containers. Ce flux de travail vous explique les étapes de configuration d'une image de conteneur avec le logiciel de votre serveur de jeu et de déploiement d'une solution d'hébergement basée sur le cloud pour votre serveur de jeu. Le flux de travail propose deux scénarios de déploiement différents : un déploiement simple et un déploiement plus complet avec une file d'attente de placement des sessions de jeu et un

FlexMatch système de matchmaking. Chaque scénario génère des flottes de Amazon GameLift Servers conteneurs et des AWS ressources de soutien.

- Processus amélioré pour configurer les profils AWS utilisateur et gérer les informations d' AWS accès pour l'utilisation des plugins. Vous pouvez gérer plusieurs profils pour travailler avec différents AWS comptes, utilisateurs de comptes et régions.
- Fonctionnalité supplémentaire pour mettre à jour les flottes de conteneurs existantes. Vous pouvez déployer de nouvelles images de conteneur (par exemple pour les mises à jour de version du serveur de jeu) et modifier les paramètres de configuration de la flotte sans avoir à recommencer depuis le début.
- Flux de travail améliorés pour l'hébergement avec les flottes Amazon GameLift Servers Anywhere et Managed EC2. Les améliorations basées sur les commentaires des clients incluent de meilleurs conseils avec des conseils et des liens vers des ressources utiles.

Les scénarios de déploiement pour les conteneurs gérés et les EC2 solutions gérées utilisent des AWS CloudFormation modèles pour créer et déployer les AWS ressources pour chaque scénario. Ces modèles sont inclus dans le téléchargement du Amazon GameLift Servers plugin et sont modifiables. Vous pouvez les utiliser tels quels ou les modifier pour votre jeu.

En savoir plus :

- [Plugin pour Unreal : déployez votre jeu sur une flotte de conteneurs gérée, guide Amazon GameLift Servers](#) du développeur
- [Plugin pour Unity : déployez votre jeu sur un parc de conteneurs géré](#), guide Amazon GameLift Servers du développeur
- [Téléchargez le plugin pour Unreal sur GitHub](#)
- [Téléchargez le plugin pour Unity sur GitHub](#)
- [Hébergement de jeux avec Amazon GameLift Servers](#)
- [Forum Amazon GameLift Servers](#)

12 novembre 2024 : Amazon GameLift Servers lance l'hébergement de jeux multijoueurs avec Managed Containers

Versions du SDK mises à jour :

- AWS SDK 1.11.445

- SDK pour serveurs, version 5.2.0 (toutes les langues)

Amazon GameLift Servers publie pour une mise à disposition générale une nouvelle solution d'hébergement pour les charges de travail des serveurs de jeux conteneurisés. Avec cette version, les développeurs de jeux peuvent désormais tirer parti des avantages de la conteneurisation, notamment des environnements cohérents et sécurisés, un processus de déploiement simplifié et une utilisation optimisée des ressources.

Les flottes de conteneurs gérées utilisent EC2 des instances Amazon qui sont gérées en Amazon GameLift Servers votre nom et en fonction de vos configurations. Vous créez une architecture de conteneur personnalisée pour votre jeu et fournissez des images de conteneur en les stockant dans un référentiel Amazon Elastic Container Registry (Amazon ECR). Les flottes de conteneurs ne sont disponibles que pour les serveurs de jeux basés sur Linux. Les serveurs de jeu doivent être intégrés au SDK Server 5.2.0 ou supérieur.

Avec les flottes de conteneurs gérées, vous bénéficiez des mêmes avantages qu'avec les EC2 flottes gérées. Cela inclut la prise en charge des types d'instances On-Demand et Spot, le dimensionnement intelligent des capacités, le placement des sessions de jeu avec des files d'attente et le matchmaking. Vous obtenez également les mêmes indicateurs que les autres types de flottes, ainsi que de nouveaux indicateurs pour les conteneurs. Les autres caractéristiques des flottes de conteneurs incluent :

- Harmonisation avec l'expérience sans serveur pour les charges de travail conteneurisées. Exécutez un processus de serveur de jeu par conteneur et regroupez de nombreux conteneurs sur chaque instance de flotte pour une utilisation optimale des ressources. Si vous préférez avoir des conteneurs avec plusieurs processus de serveur de jeu, vous pouvez utiliser l'Amazon GameLift Servers agent pour une gestion automatisée des hôtes.
- Création de flotte rationalisée. Les flottes de conteneurs sont conçues pour nécessiter des paramètres de configuration de déploiement minimaux, avec des suggested/default valeurs raisonnables. Vous pouvez rapidement déployer une flotte fonctionnelle, puis personnaliser les paramètres individuels selon vos besoins.
- Outils de gestion des versions pour l'architecture des conteneurs. Vous pouvez désormais mettre à jour une définition de groupe de conteneurs (similaire à une « tâche » de conteneur), gérer plusieurs versions et spécifier la version à déployer dans une flotte.
- Outils de mise à jour du parc. Avec les flottes de conteneurs, vous n'avez plus besoin de créer une nouvelle flotte lorsque vous souhaitez publier une mise à jour de version du serveur de jeu. Au lieu

de cela, vous pouvez désormais mettre à jour l'image de votre conteneur et déployer les mises à jour sur les flottes existantes.

Vous pouvez créer des flottes de Amazon GameLift Servers conteneurs n'importe Région AWS où, prend en Amazon GameLift Servers charge les flottes multisites, et vous pouvez déployer des instances de flotte de conteneurs sur n'importe quel site distant pris en charge. Pour en savoir plus, consultez [Amazon GameLift Serverspoints de service](#). Les conteneurs gérés ne sont actuellement pas disponibles dans les régions de AWS Chine.

En savoir plus :

- Article de blog : [Tirez parti de conteneurs entièrement gérés pour héberger des jeux multijoueurs à l'échelle mondiale sur Amazon GameLift Servers](#)
- [Conteneurs gérés](#)présentation, guide Amazon GameLift Servers du développeur
- [Comment fonctionnent les conteneurs dans Amazon GameLift Servers](#), Guide Amazon GameLift Servers du développeur
- [Feuille de route de développement pour l'hébergement avec des conteneurs Amazon GameLift Servers gérés](#), Guide Amazon GameLift Servers du développeur
- [CreateContainerFleet](#), Référence Amazon GameLift Servers d'API

19 septembre 2024 : Amazon GameLift Servers publication d'une mise à jour du SDK du serveur C++ et des plugins pour Unreal Engine

Versions du SDK mises à jour :

SDK pour serveur C++, version 5.1.3

- Nouvelles fonctionnalités de journalisation. Vous pouvez désormais accéder aux journaux des demandes du SDK.
- Fiabilité améliorée de la transmission des messages du SDK. Le SDK utilise désormais des mécanismes de reconnexion plus robustes pour assurer la restauration en cas d'interruption du réseau ou de pertes de messages aléatoires.

Versions du plugin mises à jour :

Amazon GameLift Serversplugin pour Unreal Engine, version 1.1.2

- Mise à niveau pour prendre en charge la dernière version du SDK 5.1.3 du serveur C++.
- Dans le Amazon GameLift Servers plugin pour Unreal Engine, lorsque vous recherchez un exécutable de construction de serveur pour une flotte, vous avez désormais la possibilité de parcourir tous les fichiers.

Plug-in SDK C++ Server pour Unreal, version 5.1.2

- Mise à niveau pour prendre en charge la dernière version du SDK 5.1.3 du serveur C++.

En savoir plus :

- [Intégration de jeux avec le Amazon GameLift Servers plugin pour Unreal Engine, Guide du Amazon GameLift Servers développeur](#)
- [Amazon GameLift Servers téléchargements de plugins et de SDK](#)

5 septembre 2024 : Amazon GameLift Servers améliore l'observabilité du processus de création de flotte

Sur la base des commentaires des clients, nous avons clarifié le Amazon GameLift Servers flux de travail pour créer une EC2 flotte gérée et la préparer à accueillir des sessions de jeu. Ces améliorations sont les suivantes :

- Nous avons fourni des descriptions plus spécifiques et précises de chaque phase du processus de création de flotte. Cette visibilité améliorée permet d'identifier et de résoudre les problèmes plus rapidement.
- Les phases de création et d'activation permettent de mieux séparer les tâches de déploiement d'instance (création) des tâches de démarrage des processus du serveur de jeu et de connexion au Amazon GameLift Servers service (activation). Ce changement permet de mieux identifier la cause probable des problèmes. En outre, vous pouvez désormais vous connecter à distance aux flottes lorsqu'elles sont en phase d'activation.
- Deux nouveaux événements de création de flotte indiquent le succès ou l'échec des scripts d'installation du serveur de jeu. Si la version de votre serveur de jeu inclut un script d'installation, Amazon GameLift Servers tente de l'exécuter et émet l'un des nouveaux événements suivants :
 - FLEET_CREATION_COMPLETED_INSTALLER
 - FLEET_CREATION_FAILED_INSTALLER

En savoir plus :

- [Comment fonctionne Amazon GameLift Servers la création de flottes](#), Guide Amazon GameLift Servers du développeur
- [Déboguer les incidents de flotte Amazon GameLift Servers](#), Guide Amazon GameLift Servers du développeur
- [Type de données d'événement](#), référence Amazon GameLift Servers d'API

15 août 2024 : Amazon GameLift Servers améliore l'expérience sur console

Sur la base des commentaires de nos clients, nous avons apporté les modifications suivantes à l'expérience [Amazon GameLift Serversconsole](#) :

- Vos préférences d'affichage des pages sont désormais automatiquement enregistrées auprès de l'utilisateur de votre AWS compte et appliquées chaque fois que vous revenez sur la page. Les préférences d'affichage vous permettent de choisir les informations à inclure dans un tableau, par exemple sur la page de liste des flottes. Personnalisez vos préférences d'affichage à l'aide de l'icône située dans le coin supérieur droit d'un tableau.
- Le flux de travail de création de flotte pour EC2 les flottes gérées a été rationalisé pour combiner la sélection des emplacements des flottes et des types d'instances. Nous vous avons facilité la tâche pour trouver le type d'instance adapté à votre flotte, même lorsque vous modifiez vos sélections d'emplacements.

En savoir plus :

- [Création d'une EC2 flotte Amazon GameLift Servers gérée](#), Guide Amazon GameLift Servers du développeur

25 juillet 2024 : Amazon GameLift Servers ajoute le soutien à la zone locale AWS du Nigéria

Grâce Amazon GameLift Servers à l'hébergement géré, vous pouvez désormais déployer des ressources de serveurs de jeux au Nigéria, en Afrique de l'Ouest, et étendre la portée de vos jeux aux joueurs de toute l'Afrique. Utilisez AWS les Zones Locales pour placer les serveurs de jeu

géographiquement plus près de vos joueurs afin de réduire le temps de latence et d'améliorer considérablement le gameplay.

Pour commencer immédiatement à organiser des sessions de jeu au Nigéria, ajoutez la nouvelle zone locale du Nigéria en tant que site distant à une flotte multi-sites nouvelle ou existante. Si votre jeu l'utilise Amazon GameLift ServersFlexMatch, mettez à jour les flottes dans votre file d'attente de matchmaking pour inclure la nouvelle zone locale. Avec les flottes multisites, vous pouvez gérer directement la capacité d'hébergement de chaque site.

La région mère Région AWS de la zone locale de Lagos, au Nigéria, est la région d'Afrique (Le Cap) af-south-1 (), Amazon GameLift Servers qui est également prise en charge en tant que site éloigné. Le nom de la zone locale du Nigéria af-south-1-las-1 est.

En savoir plus :

- [Amazon GameLift Serverspoints de service](#), Guide Amazon GameLift Servers du développeur
- [Mettre à jour les emplacements du parc](#), Guide Amazon GameLift Servers du développeur

02 juillet 2024 : lancement d'un Amazon GameLift Servers nouvel outil de console pour afficher les données de session des joueurs

La Amazon GameLift Servers console propose désormais un outil de recherche de session de joueur qui vous permet de récupérer les informations de session de joueur par identifiant de session de jeu, identifiant de session joueur ou identifiant de joueur. Les jeux qui utilisent le FlexMatch matchmaking génèrent automatiquement des sessions de joueur pour chaque joueur apparié. Pour tous les autres jeux, les sessions de jeu sont facultatives.

Vous trouverez l'outil de recherche de session de joueur dans la navigation principale de la Amazon GameLift Servers console. Consultez les sessions individuelles des joueurs ou comparez les données des sessions de plusieurs joueurs. Vous pouvez également ouvrir les données de session du joueur lorsque vous consultez la page détaillée d'une session de jeu.

En savoir plus :

- [Sessions de jeu et de jeu dans la Amazon GameLift Servers console](#), Guide Amazon GameLift Servers du développeur

24 avril 2024 : Amazon GameLift Servers publication d'un aperçu des flottes de conteneurs

Amazon GameLift Servers propose désormais un aperçu des flottes de conteneurs, qui vous offrent une portabilité, une évolutivité, une tolérance aux pannes et une agilité améliorées.

Dans les flottes de conteneurs, les EC2 instances Amazon hébergent un ou plusieurs de vos conteneurs. Ces conteneurs incluent votre serveur de jeu ainsi que tout ce dont il a besoin, y compris les dépendances et les configurations. Les exemples de dépendances incluent SDKs les progiciels. Une fois que vous avez chargé votre conteneur dans votre Amazon Elastic Container Registry privé, Amazon GameLift Servers il remplit votre flotte avec le conteneur.

Pour fonctionner dans une flotte de conteneurs, votre serveur de jeu doit fonctionner sous Linux et être intégré au SDK 5.x. Dans un parc de conteneurs, vous pouvez contrôler avec précision les ressources d'hébergement afin d'optimiser la consommation de ressources telles que les unités CPU et la mémoire. Vous pouvez également héberger plusieurs serveurs de jeu dans un conteneur afin de réduire l'utilisation des ressources.

Dans un parc de conteneurs, vous bénéficiez des mêmes avantages que les autres types de flottes, tels que les types d'instances à la demande, le dimensionnement (automatique et manuel), les files d'attente et le matchmaking. Vous obtenez également les mêmes indicateurs que les autres types de flottes, ainsi que de nouveaux indicateurs pour les conteneurs. Les flottes de conteneurs vous permettent de toucher les joueurs du monde entier dans les régions suivantes :

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

Pour atteindre encore plus de régions et de zones locales, créez des flottes de conteneurs multi-sites.

En savoir plus :

- [Gestion de l'hébergement avec Amazon GameLift Servers des conteneurs](#), Guide Amazon GameLift Servers du développeur

- [CreateContainerGroupDefinition](#), Référence Amazon GameLift Servers d'API

13 février 2024 : Amazon GameLift Servers lance des améliorations et simplifie l'installation du Amazon GameLift Servers plugin pour Unreal Engine SDKs

Versions du SDK mises à jour :

- Kit de développement logiciel Go Server, version 5.1.0
- SDK du serveur C#, version 5.1.2
- SDK pour serveur C++, version 5.1.2

Nous avons apporté les améliorations suivantes :

- Amélioration de la fiabilité du SDK en ajoutant une reconnexion automatique en cas d'interruption du réseau.
- [Go] Vous pouvez désormais appeler `InitSDK()` avec ou sans paramètres de serveur. Les serveurs de jeu qui s'exécutent sur EC2 des flottes Amazon GameLift Servers gérées lisent les paramètres du serveur directement à partir des variables d'environnement. Les serveurs de jeu des flottes Amazon GameLift Servers Anywhere doivent appeler `InitSDK()` avec les paramètres du serveur.

Versions du plugin mises à jour :

- Amazon GameLift Serversplugin pour Unreal Engine, version 1.1.0
- Amazon GameLift Serversplugin pour Unity, version 2.1.0
- Plug-in SDK C++ Server pour Unreal, version 5.1.1
- Plug-in SDK du serveur C# pour Unity, version 5.1.2

Nous avons apporté les améliorations suivantes :

- [Amazon GameLift Serversplugin pour Unreal Engine] Mise à jour des instructions d'installation et simplification de l'emballage. Ce plugin inclut désormais la dernière version du SDK C++ Server pour Unreal.
- Les plugins ont été mis à niveau pour prendre en charge la dernière version du SDK du serveur pour Amazon GameLift Servers.

En savoir plus :

- [Intégration de jeux avec le Amazon GameLift Servers plugin pour Unreal Engine, Guide du Amazon GameLift Servers développeur](#)
- [Amazon GameLift Serverstéléchargements de plugins et de SDK](#)

14 décembre 2023 : Amazon GameLift Servers ajout de la possibilité de mettre à jour les propriétés de jeu des sessions de jeu actives

Vous avez déjà pu définir les propriétés du jeu lors de la création de sessions de jeu et rechercher des propriétés spécifiques dans les sessions de jeu. Vous pouvez désormais également ajouter et mettre à jour ces propriétés dans une session de jeu active.

Par exemple, vos joueurs votent sur une carte sur laquelle ils veulent jouer. Votre client de jeu appelle `UpdateGameSession` pour modifier une `GameProperty` valeur en `{"Key": "map", "Value": "jungle"}`. Votre jeu implémente ensuite la nouvelle carte pour les joueurs de la session de jeu.

Les administrateurs du jeu peuvent également récupérer des données utiles à partir des propriétés du jeu en utilisant `SearchGameSessions` cette opération. Par exemple, les administrateurs peuvent répertorier les sessions de jeu ayant une `Status` valeur de `ACTIVE` et cette propriété de jeu `{"Key": "map", "Value": "desert"}`.

En savoir plus :

- [the section called “Ajouter Amazon GameLift Servers à un client de jeu”](#), Guide Amazon GameLift Servers du développeur
- [GameProperty](#), Référence Amazon GameLift Servers d'API
- [UpdateGameSession](#), Référence Amazon GameLift Servers d'API
- [SearchGameSessions](#), Référence Amazon GameLift Servers d'API

21 novembre 2023 : Amazon GameLift Servers lance le support pour les outils d'infrastructure en tant que code tels que Terraform et Pulumi optimisés par API de commande du Cloud AWS

Vous pouvez désormais gérer l'ensemble de votre pile de Amazon GameLift Servers ressources à l'aide des outils d'infrastructure en tant que code (IaC). Ces outils incluent AWS CloudFormation,

ainsi que des outils tiers tels que Terraform et Pulumi. Grâce à cette assistance supplémentaire, vous pouvez désormais vous concentrer sur le développement de votre jeu et tirer parti de DevOps stratégies pour gérer les ressources, le CI/CD et le déploiement auprès de vos clients.

Vous pouvez également désormais provisionner et configurer tous les types de Amazon GameLift Servers ressources à l'aide de l'API AWS Cloud Control. Vous pouvez continuer à utiliser les ressources à l'aide du Amazon GameLift Servers APIs ou des AWS CloudFormation modèles pour Amazon GameLift Servers.

Pour plus de détails sur les Amazon GameLift Servers ressources disponibles via laC, consultez la [Amazon GameLift Servers référence](#) du type de Amazon GameLift Servers ressource.

En outre, vous pouvez désormais redimensionner automatiquement vos flottes à l'aide de AWS CloudFormation modèles ou de l'API AWS Cloud Control en utilisant la nouvelle propriété [Fleet](#) : `ScalingPolicies`.

L'API Cloud Control fournit aux développeurs un ensemble standard de ressources permettant de APIs créer, lire, mettre à jour, supprimer et répertorier (CRUDL) sur des centaines de AWS services et de nombreux outils tiers tels que Terraform et Pulumi.

En savoir plus :

- [AWS CloudFormation](#)
- [AWS API de contrôle du cloud](#)
- [AWS Fournisseur CC Terraform](#)
- [Pulumi](#)

16 novembre 2023 : Amazon GameLift Servers mise à jour du plugin autonome pour Unity

Versions du SDK mises à jour : Amazon GameLift Servers plugin pour Unity, version 2.0.0

Le Amazon GameLift Servers plugin pour Unity fournit des outils et des flux de travail qui simplifient les étapes nécessaires à la mise en service de votre jeu Unity pour l'hébergement dans le cloud Amazon GameLift Servers. Amazon GameLift Server est un service entièrement géré qui permet aux développeurs de jeux de gérer et de faire évoluer des serveurs de jeux dédiés pour les jeux multijoueurs basés sur des sessions.

Avec cette version, le plugin pour Unity est mis à jour pour utiliser les dernières Amazon GameLift Servers fonctionnalités, notamment la version 5.x du SDK pour le serveur et la prise en charge des tests locaux avec Amazon GameLift Servers Anywhere. Le plugin est compatible avec les versions Unity 2021.3 LTS et 2022.3 LTS de Unity.

Les principales fonctionnalités du plugin incluent :

- Workflows d'interface utilisateur guidés dans l'éditeur Unity pour les scénarios suivants :
 - Testez l'intégration de votre jeu en Amazon GameLift Servers utilisant votre station de travail locale en tant qu'hôte. Ce flux de travail vous permet de configurer une flotte Amazon GameLift Servers Anywhere pour votre machine locale, de lancer des instances de votre serveur de jeu et de votre client, de demander une session de Amazon GameLift Servers jeu et de rejoindre le jeu.
 - Déployez une solution d'hébergement cloud pour votre serveur de jeu intégré avec des AWS ressources Amazon GameLift Servers gérées EC2 et de support. Ce flux de travail vous aide à configurer votre jeu pour l'hébergement dans le cloud et propose trois scénarios de déploiement :
 - Déployez le serveur de jeu sur une seule flotte.
 - Déployez le serveur de jeu sur un ensemble de flottes de Spot à faible coût dans plusieurs AWS régions.
 - Déployez le serveur de jeu avec un FlexMatch système de matchmaking.
- Possibilité de configurer des profils utilisateur liés à un AWS compte utilisateur et de définir une AWS région par défaut. Vous pouvez gérer plusieurs profils pour travailler dans différents AWS comptes, utilisateurs de comptes et régions.
- Des fonctionnalités spéciales qui aident à rationaliser les processus Amazon GameLift Servers d'intégration et de déploiement, notamment :
 - Chaque solution d'hébergement inclut des AWS ressources de support, notamment un pool d'utilisateurs Amazon Cognito qui fournit une validation unique des joueurs IDs et des joueurs. Les solutions incluent également un compartiment Amazon S3 pour le stockage, les notifications d'événements Amazon SNS, les AWS Lambda fonctions et d'autres ressources.
 - Pour le flux de travail Anywhere, le plugin automatise les paramètres de serveur requis.
 - Pour le EC2 flux de travail Amazon, chaque solution de déploiement fournit un service client intégré utilisant les fonctions Lambda. Le service principal se situe entre le client du jeu et le Amazon GameLift Servers service et gère tous les appels directs vers le Amazon GameLift Servers service.
- Contenu destiné aux tests d'intégration, y compris les ressources et le code d'un exemple de jeu multijoueur simple illustrant l'intégration du serveur de jeu et du client de jeu.

- Documentation du plugin avec des conseils d'intégration détaillés et un exemple de code.

Tous les scénarios de déploiement, y compris pour les EC2 flottes Anywhere et Amazon, utilisent des AWS CloudFormation modèles pour décrire et déployer les AWS ressources nécessaires à la solution de votre jeu. Ces modèles sont inclus dans le téléchargement du Amazon GameLift Servers plugin. Vous pouvez les utiliser tels quels ou les personnaliser pour votre jeu.

En savoir plus :

- [Amazon GameLift Serversplugin pour Unity \(SDK serveur 5.x\)](#), Guide Amazon GameLift Servers du développeur
- [Téléchargez le plugin sur GitHub](#)
- [À propos de Amazon GameLift Servers l'hébergement](#)
- [Forum Amazon GameLift Servers](#)

2 novembre 2023 : Amazon GameLift Servers ajoute la prise en charge des informations d'identification partagées

Versions du SDK mises à jour : AWS SDK 1.11.193

La nouvelle fonctionnalité d'informations d'identification Amazon GameLift Servers partagées permet aux applications déployées sur EC2 des flottes gérées d'interagir avec d'autres AWS ressources. Cette mise à jour concerne les applications que vous regroupez et déployez ainsi que les fichiers binaires du serveur de jeu intégrés au SDK du serveur version 5.x ou ultérieure. (Les exécutables du serveur de jeu peuvent déjà demander des informations d'identification à l'aide de l'action `GetFleetRoleCredentials()` 5.x du SDK du serveur.)

Par exemple, si vous souhaitez déployer la version de votre serveur de jeu avec un CloudWatch agent Amazon afin de collecter des métriques d' EC2 instance et d'autres données, l'agent a besoin d'une autorisation pour interagir avec vos CloudWatch ressources. Pour ce faire, vous devez d'abord configurer un rôle AWS Identity and Access Management IAM () autorisé à utiliser les CloudWatch ressources, puis configurer une flotte avec le rôle IAM et les informations d'identification partagées activés. Lorsque vous Amazon GameLift Servers déployez la version de votre serveur de jeu sur chaque EC2 instance, il génère un fichier d'informations d'identification partagé et le stocke sur l'instance. Toutes les applications de l'instance peuvent utiliser les informations d'identification partagées. Amazon GameLift Serversactualise automatiquement les informations d'identification temporaires pendant toute la durée de vie de l'instance.

Vous pouvez activer les informations d'identification partagées lorsque vous créez un EC2 parc géré en utilisant les méthodes suivantes :

- Dans le flux de travail de création de flotte sur Amazon GameLift Servers console.
- Lorsque vous appelez l'opération de l'API de service `CreateFleet` à l'aide du nouveau paramètre `InstanceRoleCredentialsProvider`.
- Lorsque vous appelez l'opération AWS CLI `aws gamelift create-fleet` avec le paramètre `instance-role-credentials-provider`.

En savoir plus :

- [Communiquez avec d'autres AWS ressources de vos flottes](#), guide Amazon GameLift Servers du développeur
- [CreateFleet, InstanceRoleCredentialsProvider](#), Référence Amazon GameLift Servers d'API
- [Configuration d'un rôle de service IAM](#), Guide Amazon GameLift Servers du développeur

28 septembre 2023 : Amazon GameLift Servers sortie d'un nouveau plugin autonome pour Unreal Engine Engine

Versions du SDK mises à jour : Amazon GameLift Servers plugin pour Unreal Engine version 1.0.0

Le Amazon GameLift Servers plugin pour Unreal Engine fournit des outils et des flux de travail qui simplifient les étapes nécessaires à la mise en place d'un jeu Amazon GameLift Servers pour l'hébergement dans le cloud. Amazon GameLift Servers est un service entièrement géré qui permet aux développeurs de jeux de gérer et de faire évoluer des serveurs de jeux dédiés pour les jeux multijoueurs basés sur des sessions. Le plugin prend en charge les versions UE 5.0, 5.1 et 5.2. Les principales fonctionnalités sont les suivantes :

- Les flux de travail guidés de l'interface utilisateur [dans l'éditeur Unreal] suivent les étapes suivantes :
 - Testez l'intégration de votre jeu en Amazon GameLift Servers utilisant votre station de travail locale en tant qu'hôte. Ce flux de travail vous permet de configurer une flotte Amazon GameLift Servers Anywhere pour votre machine locale, de lancer des instances de votre serveur de jeu et de votre client, de demander une session de jeu et d'obtenir les informations de connexion pour la nouvelle session de jeu. Amazon GameLift Servers

- Déployez une solution d'hébergement EC2 cloud Amazon pour votre serveur de jeu intégré. Ce flux de travail vous aide à configurer votre jeu pour l'hébergement dans le cloud et propose trois scénarios de déploiement différents : déploiement sur une flotte unique, déploiement sur un ensemble de flottes ponctuelles dans plusieurs régions ou déploiement sur un ensemble de flottes avec un FlexMatch entremetteur. La solution pour chaque scénario de déploiement inclut des Amazon GameLift Servers ressources et des AWS ressources de support.
- Possibilité de configurer des profils utilisateur liés à un utilisateur du AWS compte et de définir une AWS région par défaut. Vous pouvez gérer plusieurs profils pour travailler dans différents AWS comptes, utilisateurs de comptes et régions.
- Des fonctionnalités spéciales qui aident à rationaliser les processus Amazon GameLift Servers d'intégration et de déploiement, notamment :
 - Chaque solution d'hébergement inclut AWS des ressources de support, notamment un pool d'utilisateurs Amazon Cognito de base qui fournit un joueur unique IDs, un compartiment Amazon S3 pour le stockage, des notifications d'événements Amazon SNS et des fonctions. AWS Lambda
 - Pour le flux de travail Anywhere, le plug-in automatise les paramètres de serveur requis à l'aide d'arguments de ligne de commande.
 - Pour le EC2 flux de travail Amazon, chaque solution de déploiement fournit un service client intégré utilisant les fonctions Lambda. Le service principal reçoit les demandes des clients du jeu et les transmet au Amazon GameLift Servers service.
- Du contenu pour les tests d'intégration, notamment une carte de jeu de démarrage et deux cartes de test avec des plans de base et des éléments d'interface utilisateur.
- Documentation du plugin avec des conseils d'intégration détaillés et un exemple de code.

Tous les scénarios de déploiement, y compris pour les EC2 flottes Anywhere et Amazon, utilisent AWS CloudFormation des modèles pour décrire les solutions. Le plugin utilise ces modèles lors du déploiement de Amazon GameLift Servers ressources pour votre jeu. Ces modèles sont inclus dans le téléchargement du Amazon GameLift Servers plugin et sont modifiables. Vous pouvez les utiliser tels quels ou les modifier pour votre jeu.

En savoir plus :

- [Amazon GameLift Serversplugin pour Unreal Engine](#), Guide Amazon GameLift Servers du développeur
- [Téléchargez le plugin sur GitHub](#)

- [À propos de Amazon GameLift Servers l'hébergement](#)
- [Forum Amazon GameLift Servers](#)

17 août 2023 : Amazon GameLift Servers propose l'hébergement de serveurs de jeux avec des processeurs AWS Graviton

Versions du SDK mises à jour : AWS SDK 1.11.144

Amazon GameLift Servers Vous pouvez désormais héberger vos jeux dans le cloud à l'aide d' EC2 instances dotées de processeurs AWS Graviton. Conçues à l' AWS aide de processeurs basés sur ARM64, les instances Graviton offrent le meilleur rapport qualité-prix pour les charges de travail dans le cloud EC2, avec une amélioration allant jusqu'à 40 % par rapport aux instances x86 comparables. Les derniers processeurs Graviton3 offrent des performances de calcul jusqu'à 25 % supérieures à celles des versions précédentes.

Avec Amazon GameLift Servers, vous pouvez désormais sélectionner l'une des nouvelles instances de la famille AWS Graviton :

- Instances basées sur Graviton2 : c6g, c6gn, r6g, m6g, g5g
- Instances basées sur Graviton3 : c7g, r7g, m7g

En savoir plus :

- [AWS Processeur Graviton](#) : découvrez les avantages et les utilisations pratiques des instances basées sur EC2 Graviton.
- [Premiers pas avec Graviton](#) : obtenez un aperçu des instances basées sur Graviton et des informations sur la façon dont les applications s'exécutent sur celles-ci en fonction de leur système d'exploitation, de leurs langues et de leur durée d'exécution.

Note

Les instances Graviton Arm nécessitent un Amazon GameLift Servers serveur basé sur le système d'exploitation Linux. Le SDK Server 5.1.1 ou une version ultérieure est requis pour C++ et C#. Le SDK Server 5.0 ou une version ultérieure est requis pour Go. Ces instances ne prennent pas out-of-the-box en charge l'installation de Mono sur Amazon Linux 2023 (AL2023) ou Amazon Linux 2 (AL2).

27 juillet 2023 : Amazon GameLift Servers sortie du SDK pour serveur 5.1.0 avec un support supplémentaire pour le développement de Unity

Versions du SDK mises à jour : SDK serveur pour C++, C#/Unity, Unreal 5.1.0

La dernière version du SDK pour Amazon GameLift Servers serveur fournit des mises à jour pour C++, C# et le plugin Unreal, ainsi qu'un nouveau plugin à utiliser avec le moteur de jeu Unity. Les développeurs de jeux intègrent le SDK Amazon GameLift Servers du serveur dans les serveurs de jeux sur Amazon GameLift Servers lesquels ils déploient l'hébergement.

La dernière version du SDK du serveur contient les mises à jour suivantes, qui incluent un certain nombre de demandes de clients :

- Télécharger des packages de SDK spécifiques à une langue : le [site de Amazon GameLift Servers téléchargement](#) mis à jour contient des packages de SDK pour chaque langue. Vous pouvez télécharger les versions actuelles ou antérieures.
- Nouveau plug-in SDK pour serveur C# pour Unity — Le nouveau package SDK pour serveur pour Unity contient des bibliothèques C# intégrées que vous pouvez installer à l'aide du gestionnaire de packages dans Unity Editor (voir le nouveau guide d'intégration de [Unity](#)). Ces bibliothèques incluent les dépendances requises via UnityNuGet. Vous pouvez utiliser ce plugin avec Unity 2020.3 LTS, 2021.3 LTS et 2022.3 LTS pour Windows et Mac OS. Il prend en charge les profils .NET Framework et .NET Standard de Unity, avec .NET Standard 2.1 et .NET 4.x.
- Solution .NET consolidée pour C# — Le SDK du serveur pour C# prend désormais en charge .NET Framework 4.6.2 (mis à niveau depuis 4.6.1) et .NET 6.0 dans une seule solution. .NET Standard 2.1 est disponible avec les bibliothèques créées par Unity.
- Mises à jour 5.1.0 du SDK Server
 - [C++, C#, Unreal] Vous pouvez désormais appeler `InitSDK( )` avec ou sans paramètres de serveur. Les serveurs de jeu qui s'exécutent sur EC2 des flottes Amazon GameLift Servers gérées lisent les paramètres du serveur directement à partir des variables d'environnement. Les serveurs de jeu des flottes Amazon GameLift Servers Anywhere doivent appeler `InitSDK( )` avec les paramètres du serveur.
 - [C++, C#, Unreal] Les appels du SDK du serveur ont amélioré les messages d'erreur.
 - [SDK C++] Pour améliorer les délais de création du SDK du serveur, l'indicateur de génération `-DRUN_CLANG_FORMAT` est désactivé par défaut. Vous pouvez l'activer avec `-DRUN_CLANG_FORMAT=1`.

- [SDK C++] Lorsque vous créez les bibliothèques sans les bibliothèques standard (-DGAMELIFT_USE_STD=0), les types de std:: données InitSDK() ne sont plus utilisés.
- Documentation étendue du SDK 5.x pour serveurs
 - Guides de référence du SDK serveur mis à jour pour C++, C#/Unity et Unreal, y compris une couverture étendue de tous les types de données.
 - [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)
 - [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#)
 - [SDK 5.x du serveur C++ \(Unreal\) pour Amazon GameLift Servers -- Actions](#)
 - Nouvelles versions des guides d'intégration du SDK 5 du serveur pour les plugins Unity et Unreal
 - [Intégrer Amazon GameLift Servers dans un projet Unity](#)
 - [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#)
- Mises à jour de documentation supplémentaires
 - Documentation révisée pour les opérations Amazon GameLift Servers de l'API de service [GetComputeAccess](#) et [GetInstanceAccess](#) pour clarifier les procédures d'accès à distance en fonction de la version Amazon GameLift Servers du SDK du serveur utilisée.
 - Descriptions révisées [GameSessionPlacement](#) pour documenter la façon dont les informations de session de jeu sont transitoires lorsqu'un placement est « en attente ».

13 juillet 2023 : Amazon GameLift Servers ajoute des indicateurs relatifs au matériel du parc

Vous pouvez désormais suivre les indicateurs de performance du matériel pour vos EC2 flottes Amazon GameLift Servers gérées. Les métriques incluent les métriques d'EC2 instance relatives à l'utilisation du processeur, au volume du trafic réseau et à read/write l'activité du disque. En Amazon GameLift Servers effet, ces métriques décrivent toutes les instances actives d'un emplacement de flotte. Vous pouvez consulter ces statistiques matérielles du parc à l'aide d'un CloudWatch tableau de bord Amazon dans le AWS Management Console. Vous pouvez également les consulter dans la Amazon GameLift Servers console dans les détails du parc.

En savoir plus :

- [Surveillez Amazon GameLift Servers avec Amazon CloudWatch](#) (Mesures pour les flottes), Guide Amazon GameLift Servers du développeur

29 juin 2023 : Amazon GameLift Servers lance le support pour Amazon Linux 2023

Versions du SDK mises à jour : AWS SDK 1.11.111

Amazon GameLift Servers les clients peuvent désormais utiliser le système d'exploitation Amazon Linux 2023 pour héberger leurs serveurs de jeux. AL2023 propose plusieurs améliorations, AL2 notamment en matière de sécurité. Ce système d'exploitation est disponible partout, Régions AWS à l'exception des régions de Chine.

Les clients peuvent utiliser les nouveaux systèmes d'exploitation Linux et continuer à recevoir des mises à jour de sécurité critiques lorsque le support d'Amazon Linux (AL1) prendra fin en décembre 2023. Support pour Amazon Linux 2 jusqu'au 30 juin 2025.

En savoir plus :

- [Amazon GameLift Servers FAQ sur le serveur Linux](#)
- [Comparaison entre Amazon Linux 2 et Amazon Linux 2023](#)
- Amazon GameLift Servers Liens de référence de l'API :
 - [AWS Action du SDK CreateBuild](#)
 - [commande CLI upload-build](#)
 - [commande CLI create-build](#)

25 mai 2023 : Amazon GameLift Servers FleetIQ ajout d'un filtre pour exclure les emplacements de session de jeu sur les instances épuisées

Versions du SDK mises à jour : AWS SDK 1.11.87

Si vous l'utilisez Amazon GameLift Servers FleetIQ pour l'hébergement de jeux, vous pouvez désormais empêcher le placement de sessions de jeu sur des instances actuellement épuisées. Les instances épuisées sont signalées comme devant être arrêtées, mais elles peuvent toujours être sélectionnées pour héberger de nouvelles sessions de jeu si aucune autre ressource d'hébergement n'est disponible. Grâce à cette nouvelle fonctionnalité, vous pouvez totalement exclure l'utilisation d'instances drainantes.

Utilisez cette fonctionnalité lorsque vous appelez `ClaimGameServer` pour trouver des serveurs de jeu disponibles. Ajoutez le nouveau `FilterOption` paramètre et définissez le statut des instances autorisées sur `ACTIVE` uniquement. En réponse, Amazon GameLift Servers FleetIQ examine

uniquement les instances actives lors de la recherche et de la réclamation d'un serveur de jeu disponible.

En savoir plus :

- [ClaimGameServer](#) dans la Référence d'API Amazon GameLift Servers
- [Comment FleetIQ fonctionne](#) dans le guide du Amazon GameLift Servers FleetIQ développeur

16 mai 2023 : Amazon GameLift Servers prend en charge l'étiquetage de la répartition des coûts pour les flottes

Amazon GameLift Servers les clients peuvent désormais utiliser des balises de répartition des AWS Billing coûts pour organiser leurs coûts d'hébergement de jeux. Vous pouvez attribuer des balises de répartition des coûts aux ressources de chaque Amazon GameLift Servers EC2 flotte afin de suivre la contribution de vos flottes aux coûts d'hébergement globaux.

En savoir plus :

- [Gérez votre Amazon GameLift Servers frais d'hébergement](#)
- [Utilisation des balises de répartition des AWS coûts](#), guide de AWS Billing l'utilisateur

20 avril 2023 : Amazon GameLift Servers lance le support pour Windows Server 2016

Versions du SDK mises à jour : AWS SDK 1.11.63

Amazon GameLift Servers les clients peuvent désormais utiliser le système d'exploitation Windows Server 2016 pour héberger leurs serveurs de jeux. Ce système d'exploitation est disponible dans tous les pays Régions AWS. Les clients peuvent utiliser le nouveau système d'exploitation Windows et continuer à recevoir des mises à jour de sécurité critiques alors que Microsoft mettra fin au support de Windows Server 2012 en octobre 2023.

À compter d'aujourd'hui, les nouveaux clients qui ont besoin d'un environnement d'exécution Windows doivent spécifier Windows Server 2016 lors de la création de nouvelles versions de serveurs de jeux destinées à l'hébergement. Les clients existants peuvent continuer à créer de nouvelles versions et flottes avec Windows Server 2012, mais ils doivent terminer la migration avec Windows Server 2016 avant la date de fin du support de Microsoft, le 10 octobre 2023.

Cette mise à jour inclut les modifications de service suivantes :

- Lorsque vous créez une version de serveur de jeu à l'aide des commandes du Amazon GameLift Servers SDK ou de la CLI, vous devez désormais définir explicitement le système d'exploitation. Il n'existe plus de valeur par défaut. Pour déployer votre serveur de jeu sur Windows Server 2016, utilisez la valeur `WINDOWS_2016`.
- Lorsque vous créez une version de serveur de jeu à l'aide de la Amazon GameLift Servers console, vous devez sélectionner un système d'exploitation parmi les valeurs disponibles. Si vous êtes déjà un client disposant de flottes Windows Server 2012 actives, vous pouvez choisir `WINDOWS_2012` soit `WINDOWS_2016`.

En savoir plus :

- Amazon GameLift Servers Liens de référence de l'API :
 - [commande CLI `upload-build`](#)
 - [commande CLI `create-build`](#)
 - [AWS Action du SDK `CreateBuild`](#)
- [Amazon GameLift Servers FAQ pour Windows 2012](#)

13 avril 2023 : Amazon GameLift Servers lance le SDK 5.x du serveur pour Unreal

Versions du SDK mises à jour : Server SDK 5.0.0 pour Unreal

La dernière version du plugin Amazon GameLift Servers léger pour Unreal Engine est désormais basée sur le SDK 5.x du Amazon GameLift Servers serveur. Pour commencer à intégrer votre environnement Unreal Engine, Amazon GameLift Servers consultez les liens suivants.

En savoir plus :

- [Intégrer Amazon GameLift Servers dans un projet Unreal Engine](#)
- [Ajouter Amazon GameLift Servers à votre serveur de jeu](#)
- [SDK 5.x du serveur C++ pour Amazon GameLift Servers -- Actions](#)

14 mars 2023 : Amazon GameLift Servers lance une nouvelle expérience sur console

La nouvelle Amazon GameLift Servers console inclut les améliorations suivantes :

- Navigation améliorée — Le nouveau volet de navigation facilite la navigation entre les Amazon GameLift Servers ressources.

- Amazon GameLift Servers page de destination — La nouvelle page de destination fournit des liens vers de la documentation utile, affiche une vue d'ensemble de Amazon GameLift Servers et fournit une assistance via des liens vers la documentation, les questions fréquemment posées et AWS re:Post.
- Statistiques Amazon améliorées : CloudWatch les Amazon GameLift Servers métriques sont désormais disponibles à la fois dans la Amazon GameLift Servers console et dans vos CloudWatch tableaux de bord. Cette mise à jour inclut également de nouvelles mesures relatives aux performances, à l'utilisation et aux sessions des joueurs.

En savoir plus :

- [Suivez l'hébergement de jeux dans le Amazon GameLift Servers console](#)
- [Construire un FlexMatch entremetteur](#)

14 février 2023 : prend Amazon GameLift Servers désormais en charge le chiffrement côté serveur pour les rubriques Amazon SNS

Le chiffrement côté serveur (SSE) pour les rubriques SNS chiffre vos données sensibles au repos. SSE utilise des touches AWS Key Management Service (AWS KMS) pour protéger le contenu de vos rubriques SNS.

En savoir plus :

- [Configurer la notification des événements pour le placement des sessions de jeu](#)
- [FlexMatch événements de matchmaking](#)
- [Chiffrement au repos](#)

9 février 2023 : le SDK Amazon GameLift Servers du serveur prend en charge .NET 6 avec C #10

Versions du SDK mises à jour : Server SDK 5.0.0 pour .NET 6. Aucune mise à jour du SDK n'est requise.

Si vous utilisez la plateforme de développement en temps réel Unity, continuez à utiliser le SDK 5.0.0 Amazon GameLift Servers du serveur avec .NET 4.6. Unity ne prend pas en charge .NET 6.

En savoir plus :

- Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la mise en [Amazon GameLift Serversroute](#)
- [SDK 5.x du serveur C# pour -- Actions Amazon GameLift Servers](#)

31 janvier 2023 : le SDK Amazon GameLift Servers du serveur prend en charge le langage Go

Versions du SDK mises à jour : Server SDK 5.0.0 pour Go

En savoir plus :

- Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la mise en [Amazon GameLift Serversroute](#)
- [SDK du serveur Go pour Amazon GameLift Servers -- Actions](#)

1er décembre 2022 : Amazon GameLift Servers lancement Amazon GameLift Servers du SDK 5.0 pour Amazon GameLift Servers Anywhere et Server

Versions du SDK mises à jour : AWS SDK 1.10.21, SDK serveur 5.0.0 pour C++ et C#

Amazon GameLift ServersAnywhere utilise les ressources de votre serveur de jeu pour héberger des serveurs de Amazon GameLift Servers jeu. Vous pouvez utiliser Amazon GameLift Servers Anywhere pour intégrer vos propres ressources informatiques au EC2 calcul Amazon GameLift Servers géré afin de répartir vos serveurs de jeu sur plusieurs types de calcul. Vous pouvez également utiliser Amazon GameLift Servers Anywhere pour tester vos serveurs de jeu de manière itérative sans télécharger le build à Amazon GameLift Servers chaque itération.

Points forts :

- Nouveaux types de flotte et de calcul Amazon GameLift Servers Anywhere
- Amazon GameLift ServersEnregistrement des ressources informatiques en tout lieu
- Cycle d'itération des tests amélioré

Amazon GameLift ServersLe SDK de serveur 5.0.0 apporte des améliorations au SDK de serveur existant et introduit un nouveau type de ressource, le calcul. Server SDK 5.0.0 prend en charge

Amazon GameLift Servers Anywhere et l'utilisation de vos propres ressources informatiques pour l'hébergement de serveurs de jeux.

En savoir plus :

- [SDK de serveur 5.x pour Amazon GameLift Servers](#)
- [Emplacement de la flotte](#)
- [Choisissez les ressources de calcul pour un parc géré](#)
- [Créez un Amazon GameLift Servers Flotte Anywhere](#)

25 août 2022 : Amazon GameLift Servers lance le support pour les Zones Locales

Versions du SDK mises à jour : AWS SDK 1.9.333

Amazon GameLift Servers est désormais disponible dans huit Zones Locales aux États-Unis d'Amérique, ce qui vous permet de déployer vos flottes au plus près des joueurs. Vous pouvez utiliser toutes les Amazon GameLift Servers fonctionnalités gérées avec les zones locales en ajoutant les zones locales à vos flottes.

Les Zones AWS Locales étendent les ressources et les services jusqu'à la périphérie du cloud, à proximité de grands centres urbains, industriels et informatiques (IT). Cela signifie que vous pouvez déployer des applications qui nécessitent une latence d'une milliseconde à un chiffre plus près des utilisateurs finaux ou des centres de données sur site.

En savoir plus :

- [Amazon GameLift Servers zones locales](#)
- [Emplacement de la flotte](#)
- [Création d'une EC2 flotte Amazon GameLift Servers gérée](#)

28 juin 2022 : Amazon GameLift Servers lance une nouvelle expérience de console opt-in

La nouvelle Amazon GameLift Servers console inclut les améliorations suivantes :

- Navigation améliorée — Le nouveau volet de navigation facilite la navigation entre les Amazon GameLift Servers ressources.

- Amazon GameLift Serverspage de destination — La nouvelle page de destination fournit des liens vers de la documentation utile, affiche une vue d'ensemble de Amazon GameLift Servers et fournit une assistance via des liens vers la documentation, les questions fréquemment posées et AWS re:Post.
- Statistiques Amazon améliorées : CloudWatch les Amazon GameLift Servers métriques sont désormais disponibles à la fois dans la Amazon GameLift Servers console et dans vos CloudWatch tableaux de bord. Cette mise à jour inclut également de nouvelles mesures relatives aux performances, à l'utilisation et aux sessions des joueurs.

En savoir plus :

- [Suivez l'hébergement de jeux dans le Amazon GameLift Servers console](#)
- [Construire un FlexMatch entremetteur](#)

15 février 2022 : FlexMatch ajoute une règle composée et des améliorations supplémentaires

FlexMatchles utilisateurs ont désormais accès aux fonctionnalités suivantes :

- Règle composée — Ajout de la prise en charge des règles de matchmaking composées pour les matchs de 40 joueurs ou moins. Vous pouvez désormais utiliser des instructions logiques pour créer une règle composée afin de former une correspondance. En l'absence de règle composée dans votre ensemble de règles, pour qu'une correspondance soit établie, toutes les règles de l'ensemble de règles doivent être vraies. Avec les règles composées, vous pouvez choisir les règles à appliquer à l'aide des opérateurs logiques suivants : `and`, `or`, `not`, et `xor`.
- Sélection flexible des équipes — Expressions de propriétés de matchmaking mises à jour pour faciliter la sélection d'un sous-ensemble de toutes les équipes disponibles.
- Listes de chaînes plus longues : augmentation du nombre maximum de chaînes de 10 à 100 dans une liste de chaînes contenant les valeurs des attributs des joueurs.

En savoir plus :

- [Amazon GameLift ServersFlexMatchguide du développeur](#) :
 - [FlexMatchtypes de règles](#)
 - [FlexMatchexpressions de propriété](#)

- [AttributeValue: SL](#)

28 octobre 2021 : Amazon GameLift Servers ajoute le support pour les flottes multirégionales dans la région Asie-Pacifique (Osaka) ; Amazon GameLift ServersFleetIQ ajoute le support pour AWS les processeurs Graviton2

[Versions du SDK mises à jour : AWS SDK 1.9.133](#)

Amazon GameLift Servers est désormais disponible dans la région Asie-Pacifique (Osaka). Les développeurs de jeux peuvent désormais déployer des instances à Osaka à l'aide d' GameLift une flotte multirégionale.

Vous pouvez désormais utiliser les serveurs de jeu hébergés par Graviton2, basés sur l'architecture de processeur ARM, pour obtenir des performances accrues à moindre coût par rapport aux options de calcul équivalentes basées sur Intel.

Points forts :

- Amazon GameLift Servers est désormais disponible dans la région Asie-Pacifique (Osaka).
- Amazon GameLift ServersFleetIQles groupes de serveurs de jeu peuvent désormais être configurés pour gérer les familles d'instances Graviton2 c6g, m6g et r6g.

En savoir plus :

- [Amazon GameLift ServersFlotte multirégionale](#)
- [CreateGameServerGroup](#)
- [AWS processeur graviton](#)

20 septembre 2021 : Amazon GameLift Servers sortie du plugin pour Unity

Le Amazon GameLift Servers plugin pour Unity version 1.0.0 contient des bibliothèques et une interface utilisateur native qui facilitent l'accès aux Amazon GameLift Servers ressources et l'intégration Amazon GameLift Servers dans votre jeu Unity. Vous pouvez utiliser le Amazon GameLift Servers plugin pour Unity afin d'accéder à des AWS CloudFormation modèles pour des scénarios de jeu courants Amazon GameLift Servers APIs et de les déployer. Le plugin inclut également un exemple de jeu qui fonctionne avec les exemples de scénarios. Vous pouvez utiliser

Amazon GameLift Servers Local pour voir les messages transmis entre le client du jeu et le serveur de jeu afin de savoir comment un jeu classique interagit avec Amazon GameLift Servers.

Le plugin pour Unity prend en charge Unity 2019.4 LTS et 2020.3 LTS.

Points forts :

- Créez, exécutez et modifiez un exemple de jeu avec différents scénarios, ou créez le vôtre.
- Déployez AWS CloudFormation des exemples de scénarios pour des scénarios de jeu typiques, notamment l'authentification uniquement, une flotte à région unique, des flottes multirégionales avec file d'attente et système de matchmaking personnalisé, des flottes ponctuelles avec file d'attente et un système de matchmaking personnalisé, et. FlexMatch

En savoir plus :

- [Intégration de jeux avec le Amazon GameLift Servers plugin pour Unity](#)

30 juin 2021 : FlexMatch ajoute la règle BatchDistance

Vous pouvez utiliser le type de règle BatchDistance pour spécifier une chaîne ou un attribut numérique, ce qui apporte de nombreux avantages à chaque segment.

Points forts :

- Pour les matchs de grande envergure (>40 joueurs), au lieu d'équilibrer les joueurs uniquement en fonction de leurs compétences, vous pouvez désormais obtenir le même équilibre en fonction des compétences, des modes et des cartes. Assurez-vous que tous les participants au match appartiennent à une fourchette de compétences, associez plusieurs attributs numériques tels que la ligue ou le style de jeu, et regroupez en fonction d'attributs de chaîne tels que la carte ou le mode de jeu. Vous pouvez également créer des extensions au fil du temps. Par exemple, vous pouvez créer une extension pour permettre à un plus grand nombre de niveaux de compétence d'entrer dans le match au fur et à mesure que le joueur attend.

Pour les matchs de moins de 40 joueurs, vous pouvez utiliser une nouvelle expression de règles simplifiée.

3 juin 2021 : mises à jour du SDK client et du SDK serveur en Amazon GameLift Servers temps réel

Versions du SDK mises à jour : Realtime Client SDK 1.2.0, SDK Server 3.4.0 pour Unreal

Avec cette dernière mise à jour du SDK, vous pouvez désormais intégrer le IL2 CPP dans vos applications mobiles qui utilisent le SDK client RTS et suivre les meilleures pratiques en matière de frameworks. Vous pouvez également désormais créer le SDK du Amazon GameLift Servers serveur pour Unreal version 4.26. Cette mise à jour contient des composants qui s'intègrent à votre serveur de jeu Windows ou Linux, notamment les versions C++ et C# du SDK du Amazon GameLift Servers serveur, Amazon GameLift Servers Local et un plug-in Unreal Engine.

Points forts :

- Ajout de la prise en charge du IL2 CPP dans le SDK du client RTS et de la création des bibliothèques natives sous forme de frameworks, afin que vous puissiez créer des clients RTS pour les appareils mobiles les plus récents.
- Vous pouvez l'utiliser [DescribePlayerSessions\(\)](#) pour obtenir des informations pour une session solo, pour toutes les sessions d'une session de jeu ou pour toutes les sessions de joueur associées à un identifiant solo.
- Vous pouvez l'utiliser [GetInstanceCertificate\(\)](#) pour récupérer l'emplacement du fichier d'un certificat TLS codé PEM associé à la flotte et à ses instances.
- Support du SDK serveur créé pour Unreal version 4.26.
- Le SDK C# existant, version 4.0.2, a été vérifié comme compatible avec Unity 2020.3. Aucune mise à jour du SDK n'était requise.

En savoir plus :

- [Amazon GameLift ServersGuide du développeur](#) :
 - [DescribePlayerSessions\(\)](#)
 - [GetInstanceCertificate\(\)](#)

23 mars 2021 : Amazon GameLift Servers ajoute des notifications au placement des sessions de jeu

[Versions du SDK mises à jour : AWS SDK 1.8.168](#)

Vous pouvez désormais utiliser les événements pour surveiller l'activité de placement des sessions de jeu dans une file d'attente de session de jeu. Créez une rubrique Amazon Simple Notification Service (Amazon SNS) pour publier des notifications d'événements, ou configurez le suivi des événements à l'aide d'Events. CloudWatch

Points forts :

- Pour chaque file d'attente, vous pouvez définir une chaîne de texte personnalisée à inclure dans tous les messages relatifs aux événements.
- Lorsque vous utilisez une rubrique Amazon SNS, vous pouvez définir des conditions d'accès supplémentaires qui limitent la publication à des files d'attente spécifiques.

En savoir plus :

- Amazon GameLift ServersGuide du développeur :
 - [Configurer la notification des événements pour le placement des sessions de jeu](#) (nouveau)
 - [Événements de placement lors des sessions de jeu](#) (nouveau)
- [Référence d'API \(AWS SDK\)](#)
 - Nouveaux paramètres de file d'attente pour les sessions de jeu NotificationTarget et CustomEventData : [GameSessionQueue](#), [CreateGameSessionQueue](#), [UpdateGameSessionQueue](#)
- [Forum Amazon GameLift Servers](#)

16 mars 2021 : Amazon GameLift Servers ajout de flottes multirégionales, six nouvelles régions

[Versions du SDK mises à jour : AWS SDK 1.8.163](#)

Amazon GameLift Serversl'hébergement géré est désormais disponible dans 21 AWS régions. Les nouvelles régions sont le Cap (af-south-1), Bahreïn me-south-1 (), Hong Kong ap-east-1 (), Milan eu-south-1 (), Paris eu-west-3 () et Stockholm eu-north-1 ().

Grâce à la nouvelle fonctionnalité de Amazon GameLift Servers flottes multisites, vous pouvez désormais configurer une flotte unique pour héberger vos serveurs de jeu dans l'une ou l'ensemble des 20 régions Amazon GameLift Servers prises en charge (à l'exception de la région de Pékin). Cette fonctionnalité vise à réduire considérablement le travail requis pour configurer et gérer les

ressources Amazon GameLift Servers d'hébergement dans le monde entier. Des flottes multisites peuvent être créées dans les AWS régions suivantes : us-east-1 (Virginie du Nord), us-west-2 (Oregon), eu-central-1 (Francfort), (eu-west-1 Irlande), ap-southeast-2 (Sydney), ap-northeast-1 (Tokyo) et ap-northeast-2 (Séoul). Dans toutes les autres régions, vous pouvez continuer à configurer des flottes à site unique selon vos besoins. Toutes les flottes créées avant cette version sont des flottes à site unique. L'utilisation de flottes multisites n'a aucune incidence sur vos coûts d'hébergement. Amazon GameLift Servers la tarification est basée sur le type, l'emplacement et le volume d'instances que vous utilisez. (Pour plus d'informations, consultez [Amazon GameLift Servers les tarifs](#).) AWS CloudFormation le support pour les flottes multi-sites sera bientôt disponible.

 Note

Les flottes multisites ne sont pas disponibles dans les régions de Chine. Amazon GameLift Servers les ressources situées dans les régions de Chine ne peuvent pas interagir avec les ressources d'autres Amazon GameLift Servers régions ni être utilisées par celles-ci.

Points forts :

- Dans le cas d'un parc multisite, ajoutez explicitement une liste de sites distants. Amazon GameLift Servers déploie des instances du même type et de la même configuration, y compris la configuration de construction et d'exécution, dans la région d'origine de la flotte et sur tous les sites ajoutés.
- Ajustez les paramètres de capacité et la mise à l'échelle pour chaque site indépendamment. Les politiques de dimensionnement automatique s'appliquent à l'ensemble du parc, mais vous pouvez les activer ou les désactiver par emplacement.
- Démarrez de nouvelles sessions de jeu à des emplacements spécifiques de la flotte. Lorsque vous utilisez des files d'attente ou le matchmaking pour placer des sessions de jeu, vous pouvez désormais hiérarchiser le début des nouvelles sessions de jeu en fonction de l'emplacement, du coût d'hébergement et de la latence des joueurs.
- Obtenez des statistiques d'hébergement dans la Amazon GameLift Servers console, agrégées pour tous les sites d'une flotte ou ventilées par emplacement de flotte.

En savoir plus :

- [Blog sur les technologies liées aux jeux Amazon](#)

- [Référence d'API \(AWS SDK\)](#)
 - Nouvelles opérations de localisation de flottes :
[CreateFleetLocations](#), [DescribeFleetLocationAttributes](#), [DescribeFleetLocationCapacity](#),
[DescribeFleetLocationUtilization](#), [DeleteFleetLocations](#)
 - Mise à jour des opérations de flotte, avec un nouveau support multi-sites :
[CreateFleet](#), [UpdateFleetCapacity](#), [Décrire EC2 InstanceLimits](#), [DescribeInstances](#),
[StopFleetActions](#), [StartFleetActions](#)
 - Opérations de placement des sessions de jeu mises à jour, avec une nouvelle priorité et une nouvelle capacité de filtrage : [CreateGameSessionQueue](#), [DescribeGameSessionQueues](#),
[UpdateGameSessionQueue](#)
 - Opérations de création de sessions de jeu mises à jour, avec une nouvelle prise en charge de la localisation : [CreateGameSession](#), [DescribeGameSessions](#), [DescribeGameSessionDetails](#),
[SearchGameSessions](#)
- [Amazon GameLift ServersGuide du développeur](#) :
 - [Amazon GameLift Serverspoints de service](#)(mis à jour)
 - [Personnalisez votre Amazon GameLift Servers EC2 flottes gérées](#) (nouveau)

[Augmenter la capacité d'hébergement de jeux avec Amazon GameLift Servers](#)(mis à jour)
 - [Personnaliser une file d'attente de session de jeu](#) (nouveau)
 - [Détails du parc dans la Amazon GameLift Servers console](#)(mis à jour)
- [Forum Amazon GameLift Servers](#)

9 février 2021 : extension de la Amazon GameLift Servers prise en charge des instances AMD, en mode autonome FlexMatch

[Versions du SDK mises à jour : AWS SDK 1.8.139](#)

Cette version inclut les mises à jour suivantes :

- Amazon GameLift ServersFleetIQles groupes de serveurs de jeu peuvent désormais être configurés pour gérer les familles d'instances AMD C5a, M5a et R5a. Les types d' EC2 instances Amazon pris en charge, tels que listés pour le GameServerGroup [InstanceDefinition](#), incluent désormais les suivants :
 - c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge

- m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12x large, m5a.16x large, m5a.24xlarge
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12x large, r5a.16xlarge, r5a.24xlarge

Remarque : les instances AMD pour ne FleetIQ sont actuellement pas disponibles pour une utilisation dans la AWS région de Chine (Pékin). Voir [Disponibilité des fonctionnalités et différences de mise](#) en œuvre en Chine.

- Amazon GameLift Servers l'hébergement géré de jeux prend désormais en charge les instances AMD dans la région de Chine (Pékin), exploitées par Sinnet. Les nouvelles familles d'instances AMD incluent M5a et R5a. Les types d' EC2 instances pris en charge, tels qu'ils sont répertoriés pour le parc [InstanceType](#), incluent désormais les suivants :
 - m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12x large, m5a.16x large, m5a.24xlarge
 - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12x large, r5a.16xlarge, r5a.24xlarge
- Amazon GameLift Servers FlexMatch peut désormais être utilisée comme solution de matchmaking autonome dans la région de Chine (Pékin), exploitée par Sinnet. Les clients peuvent créer un FlexMatch système de jumelage dans la région de Pékin et configurer le [FlexMatchMode](#) paramètre sur STANDALONE. Pour plus d'informations sur FlexMatch l'hébergement Amazon GameLift Servers géré ou sur une solution autre que l'Amazon GameLift Servers hébergement, consultez le [guide du Amazon GameLift Servers FlexMatch développeur](#).
- Lorsque vous configurez les notifications d'événements pour Amazon GameLift Servers FlexMatch, vous pouvez désormais désigner une rubrique Amazon SNS FIFO comme cible de notification. Pour plus d'informations, consultez :
 - [MatchmakingConfiguration NotificationTarget](#), Référence Amazon GameLift Servers d'API
 - [Configuration de la notification des FlexMatch événements](#), guide Amazon GameLift Servers FlexMatch du développeur
 - [Présentation d'Amazon SNS FIFO : First-in-first-out Pub/Sub messagerie](#), blog d'actualités AWS

22 décembre 2020 : le SDK Amazon GameLift Servers du serveur prend en charge Unreal Engine 4.25 et Unity 2020

Versions du SDK mises à jour : Amazon GameLift Servers Server SDK 4.0.2, plugin Unreal version 3.3.3

La dernière version du SDK pour Amazon GameLift Servers serveurs contient les composants suivants :

- Le plugin Unreal mis à jour a été mis à jour pour être compatible avec Unreal Engine 4.25. L'API n'a pas été modifiée.
- Le SDK C# existant, version 4.0.2, a été vérifié comme compatible avec Unity 2020. Aucune mise à jour du SDK n'était requise.

Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la [mise Amazon GameLift Servers en route](#).

24 novembre 2020 : Amazon GameLift ServersFlexMatch désormais disponible pour les jeux hébergés n'importe où

[Versions du SDK mises à jour : AWS SDK 1.8.95](#)

Amazon GameLift ServersFlexMatch est un service de matchmaking personnalisable pour les jeux multijoueurs. Conçu initialement pour les utilisateurs d'hébergement Amazon GameLift Servers géré, il FlexMatch peut désormais être intégré à des jeux utilisant d'autres systèmes d'hébergement peer-to-peer, notamment l'informatique propriétaire sur site et les types primitifs de cloud computing. Les jeux utilisés Amazon GameLift Servers FleetIQ pour l'hébergement de jeux sur Amazon EC2 peuvent désormais implémenter le matchmaking avec FlexMatch.

FlexMatch fournit un algorithme de matchmaking robuste et un langage de règles qui vous donnent une grande latitude pour personnaliser le processus de matchmaking afin que les joueurs soient jumelés en fonction des caractéristiques clés des joueurs et de la latence signalée. En outre, FlexMatch propose un flux de travail de demande de matchmaking qui prend en charge des fonctionnalités telles que les parties entre joueurs, l'acceptation des joueurs et le remblayage des matchs. Lorsque vous utilisez FlexMatch un hébergement Amazon GameLift Servers géré ou Amazon GameLift Servers Realtime, le système de matchmaking recherche automatiquement des Amazon GameLift Servers ressources d'hébergement et démarre une nouvelle session de jeu pour les matchs nouvellement formés. Lors de l'utilisation en FlexMatch tant que service autonome, le système de matchmaking renvoie les résultats des matchs à votre jeu, qui peut ensuite démarrer une nouvelle session de jeu à l'aide de votre solution d'hébergement.

Les opérations d'API pour FlexMatch font partie de l'API de Amazon GameLift Servers service, qui est incluse dans le AWS SDK et le AWS Command Line Interface (AWS CLI). Cette version inclut ces mises à jour pour prendre en charge le matchmaking autonome :

- La ressource `API MatchmakingConfiguration` présente les modifications suivantes :
 - Nouvelle propriété, `FlexMatchMode` indique si le système de matchmaking est utilisé avec un hébergement Amazon GameLift Servers géré ou en tant que matchmaking autonome.
 - La propriété `nGameSessionQueueArn` n'est pas obligatoire lorsqu'elle `FlexMatchMode` est définie sur autonome.
 - Ces propriétés ne sont pas utilisées avec le matchmaking autonome : `AdditionalPlayerCount`, `BackfillModeGameProperties`, `GameSessionData`.
- La fonction de remblayage automatique n'est pas disponible avec le matchmaking autonome.

24 novembre 2020 : les instances AMD sont désormais disponibles sur Amazon GameLift Servers

[Versions du SDK mises à jour : AWS SDK 1.8.95](#)

La liste des types d' EC2 instances Amazon Amazon GameLift Servers actuellement pris en charge inclut trois nouvelles familles d'instances : C5a, M5a et R5a. Ces familles se composent d'instances optimisées pour le calcul AMD alimentées par des processeurs AMD EPYC fonctionnant à des fréquences allant jusqu'à 3,3. GHz. Les instances AMD sont compatibles x86 ; les jeux en cours d'exécution Amazon GameLift Servers peuvent être déployés sur des types d'instances AMD sans modification. Les nouvelles instances sont disponibles dans les AWS régions suivantes : États-Unis Est (Virginie du Nord et Ohio), États-Unis Ouest (Oregon et Californie du Nord), centre du Canada (Montréal), Amérique du Sud (Sao Paulo), Europe centrale (Francfort), UE ouest (Londres et Irlande), Asie-Pacifique sud (Mumbai), Asie-Pacifique nord-est (Séoul et Tokyo) et Asie-Pacifique sud-est (Singapour et Sydney).

Les nouvelles instances AMD incluent :

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge, c5a.24xlarge
- m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12x large, m5a.16x large, m5a.24xlarge
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12x large, r5a.16xlarge, r5a.24xlarge

En savoir plus :

- [Blog sur les technologies liées aux jeux Amazon](#)
- [Amazon GameLift Serverstarification des instances](#)
- [EC2 Instances Amazon dotées de processeurs AMD EPYC](#)
- [Forum Amazon GameLift Servers](#)

11 novembre 2020 : mise à jour de version Amazon GameLift Servers du SDK du serveur

Versions du SDK mises à jour : Amazon GameLift Servers Server SDK 4.0.2

La nouvelle version 4.0.2 du SDK pour serveurs corrige un problème connu lié au fonctionnement de l'API. `StartMatchBackfill()` Cette opération renvoie désormais une réponse correcte à une demande de remplissage correspondant.

Le problème n'a pas affecté le processus de remblayage des matchs, et le fonctionnement de cette fonctionnalité n'a pas changé. Le problème a peut-être eu un impact sur la messagerie du journal et la gestion des erreurs pour les demandes de remplacement des matchs.

Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la [mise Amazon GameLift Servers en route](#).

5 novembre 2020 : Nouvelles personnalisations de FlexMatch l'algorithme

FlexMatchles utilisateurs peuvent désormais ajuster les comportements par défaut suivants pour le processus de matchmaking. Ces personnalisations sont définies dans un ensemble de règles de matchmaking. Aucune modification n'a été apportée au Amazon GameLift Servers SDKs.

- Prioriser les tickets de remblayage : Vous pouvez choisir d'augmenter ou de diminuer la priorité des tickets de remblayage lorsque vous recherchez des correspondances acceptables. La priorisation des tickets de remblayage est utile lorsque la fonction de remplissage automatique est activée. Utilisez la propriété de l'algorithme `backfillPriority`.
- Pré-tri pour optimiser la cohérence et l'efficacité des matchs : configurez votre système de jumelage pour qu'il prétrie le pool de tickets avant de regrouper les tickets pour évaluation. En triant les tickets en fonction des attributs clés des joueurs, les matchs qui en résultent ont tendance à avoir des joueurs plus similaires en ce qui concerne ces attributs. Vous pouvez également améliorer l'efficacité du processus d'évaluation en effectuant un tri préalable sur les mêmes attributs que ceux utilisés dans les règles de match. Utilisez la propriété de l'algorithme `sortByAttributes` avec la `strategy` propriété définie sur « sorted ».

- Ajustez la façon dont les temps d'attente pour les extensions sont déclenchés : Choisissez entre le déclenchement des extensions en fonction de l'âge du ticket le plus récent (par défaut) ou du ticket le plus ancien dans un match incomplet. Le déclenchement sur le ticket le plus ancien a tendance à terminer les matchs plus rapidement, tandis que le déclenchement sur le ticket le plus récent améliore la qualité des matchs. Utilisez la propriété de l'algorithme `expansionAgeSelection`.

17 septembre 2020 : Amazon GameLift Servers mise à jour du SDK du serveur

Versions du SDK mises à jour : Amazon GameLift Servers Server SDK 4.0.1

Le nouveau SDK pour serveurs contient les mises à jour suivantes :

- API C# version 4.0.1
 - Le fonctionnement de l'API n'[TerminateGameSession\(\)](#) est plus pris en charge. Remplacez par un appel à pour mettre fin [ProcessEnding\(\)](#) à la fois à une session de jeu et au processus du serveur.
 - Un problème connu lié à l'opération [GetInstanceCertificate\(\)](#) est résolu.
 - L'opération renvoie [GetTerminationTime\(\)](#) désormais une valeur de type de données `AwsDateTimeOutcome`.
- API C++ version 3.4.1
 - L'opération n'[TerminateGameSession\(\)](#) est plus prise en charge. Remplacez-le par un appel à pour mettre fin [ProcessEnding\(\)](#) à la fois à une session de jeu et au processus du serveur.
- Plug-in Unreal Engine version 3.3.2
 - L'opération n'[TerminateGameSession\(\)](#) est plus prise en charge. Remplacez-le par un appel à pour mettre fin [ProcessEnding\(\)](#) à la fois à une session de jeu et au processus du serveur.
 - L'opération de rappel `OnUpdateGameSession` est ajoutée pour prendre en charge [FProcessParamètres](#) le remplissage des matchs.

Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la [mise Amazon GameLift Servers en route](#).

27 août 2020 : Amazon GameLift ServersFleetIQ pour l'hébergement de jeux avec Amazon EC2 (disponibilité générale)

[Versions du SDK mises à jour : AWS SDK 1.8.36](#)

La Amazon GameLift Servers FleetIQ solution d'hébergement de jeux cloud à faible coût sur Amazon EC2 est désormais généralement disponible. Amazon GameLift Servers FleetIQ permet aux développeurs d'héberger des serveurs de jeux directement sur des instances Amazon EC2 Spot en optimisant leur viabilité pour l'hébergement de jeux. Les développeurs de jeux peuvent l'utiliser Amazon GameLift Servers FleetIQ avec de nouveaux jeux ou pour augmenter la capacité des jeux existants. Cette solution prend en charge l'utilisation de conteneurs ou d'autres AWS services tels que AWS Shield et Amazon Elastic Container Service (Amazon ECS).

Cette version de disponibilité générale inclut les mises à jour suivantes de la Amazon GameLift Servers FleetIQ solution :

- Une nouvelle opération d'API `DescribeGameServerInstances` renvoie des informations, y compris le statut, sur toutes les instances actives d'un groupe de serveurs de Amazon GameLift Servers FleetIQ jeu.
- La nouvelle stratégie d'équilibrage configure un groupe de serveurs de jeu pour utiliser uniquement les instances à la demande. `ON_DEMAND_ONLY` Vous pouvez mettre à jour la stratégie d'équilibrage d'un groupe de serveurs de jeu à tout moment, ce qui permet de basculer entre l'utilisation d'instances ponctuelles et d'instances à la demande selon les besoins.
- Les éléments d'aperçu suivants ont été supprimés pour des raisons de disponibilité générale :
 - Utilisation de clés de tri personnalisées pour les ressources du serveur de jeu. Les serveurs de jeu peuvent être triés en fonction de l'horodatage des inscriptions.
 - Balisage des ressources du serveur de jeu.

16 avril 2020 : Amazon GameLift Servers mise à jour du SDK du serveur pour Unity et Unreal Engine

Versions du SDK mises à jour : Amazon GameLift Servers Server SDK 4.0.0, Local 1.0.5 Amazon GameLift Servers

La dernière version du kit SDK Amazon GameLift Servers Server contient les éléments mis à jour suivants :

- Version 4.0.0 du SDK C# mise à jour pour Unity 2019.
- La version 3.3.1 du plugin Unreal a été mise à jour pour les versions 4.22, 4.23 et 4.24 d'Unreal Engine.
- Amazon GameLift Servers La version locale 1.0.5 a été mise à jour pour tester les intégrations utilisant le SDK du serveur C# version 4.0.0.

Téléchargez la dernière version du SDK du Amazon GameLift Servers serveur lors de la [mise Amazon GameLift Servers en route](#).

2 avril 2020 : Amazon GameLift Servers FleetIQ disponible pour l'hébergement du jeu le EC2 (aperçu public)

[Versions du SDK mises à jour : AWS SDK 1.7.310](#)

La fonction Amazon GameLift Servers FleetIQ optimise la viabilité des instances Spot à faible coût pour une utilisation avec l'hébergement de jeux. Cette fonctionnalité est désormais étendue aux clients qui souhaitent gérer leurs ressources d'hébergement directement plutôt que via le service Amazon GameLift Servers géré. Cette solution prend en charge l'utilisation de conteneurs ou d'autres AWS services tels que AWS Shield et Amazon Elastic Container Service (Amazon ECS).

En savoir plus :

GameTech article de [blog](#) sur Amazon GameLift Servers FleetIQ

19 décembre 2019 : Amélioration de la gestion des AWS Amazon GameLift Servers ressources

[Versions du SDK mises à jour : AWS SDK 1.7.249](#)

Vous pouvez désormais tirer parti des outils de gestion des AWS Amazon GameLift Servers ressources avec les ressources. En particulier, toutes les Amazon GameLift Servers ressources clés (builds, scripts, flottes, files d'attente de sessions de jeu, configurations de matchmaking et ensembles de règles de matchmaking) se voient désormais attribuer des valeurs Amazon Resource Name (ARN). Un ARN de ressource fournit un identifiant cohérent unique dans toutes les AWS régions. Ils peuvent être utilisés pour créer des politiques d'autorisations spécifiques aux ressources AWS Identity and Access Management (IAM). Les ressources se voient désormais attribuer un ARN ainsi que l'identifiant de ressource préexistant, qui n'est pas spécifique à une région.

De plus, les ressources Amazon GameLift Servers prennent désormais en charge le balisage. Vous pouvez utiliser des balises pour organiser les ressources, créer des politiques d'autorisation IAM pour gérer l'accès à des groupes de ressources, personnaliser la ventilation des AWS coûts, etc. Lors de la gestion des balises pour les ressources Amazon GameLift Servers, utilisez les actions de l'API Amazon GameLift Servers `TagResource()`, `UntagResource()` et `ListTagsForResource()`.

En savoir plus :

- [TagResource](#) dans la Référence d'API Amazon GameLift Servers
- [Balisage des ressources AWS](#) dans les Références générales AWS
- [Noms des ressources Amazon](#) dans la référence AWS générale

14 novembre 2019 : Nouveaux AWS CloudFormation modèles, mises à jour dans la région de Chine (Pékin)

[Versions du SDK mises à jour : AWS SDK 1.7.210](#)

AWS CloudFormation modèles pour Amazon GameLift Servers

Amazon GameLift Servers les ressources peuvent désormais être créées et gérées via AWS CloudFormation. Les modèles de AWS CloudFormation construction et de flotte existants ont été mis à jour pour s'aligner sur les ressources actuelles, et de nouveaux modèles sont désormais disponibles pour les scripts, les files d'attente, les configurations de matchmaking et les ensembles de règles de matchmaking. AWS CloudFormation les modèles simplifient considérablement la gestion de groupes de AWS ressources connexes, en particulier lors du déploiement de jeux dans plusieurs régions.

En savoir plus :

- [Amazon GameLift Servers référence au type de ressource](#) dans le guide de AWS CloudFormation l'utilisateur
- [Gestion Amazon GameLift Servers héberger des ressources en utilisant AWS CloudFormation](#) dans le guide Amazon GameLift Servers du développeur

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