



Guía de alojamiento

Amazon GameLift Servers



Amazon GameLift Servers: Guía de alojamiento

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon GameLift Servers?	1
Características principales de	1
Cómo trabajar con Amazon GameLift Servers	4
Opciones de alojamiento de juegos	5
EC2 administrado	6
Contenedores administrados	6
Alojamiento con Anywhere	7
Alojamiento híbrido	8
Funcionamiento	9
Componentes básicos	9
Cómo funcionan conjuntamente los componentes	16
Experiencia del jugador	18
Proporcione un juego rápido a los jugadores de todo el mundo	18
Ayude a los jugadores a unirse a los juegos rápidamente	18
Idee formas creativas de agrupar a los jugadores en las sesiones de juego	19
Ofrezca conexiones fiables durante todo el juego	20
Mejore la experiencia de los jugadores basándose en datos reales	20
Integre las funciones mejoradas del juego con otros AWS servicios	21
Alojamiento de juegos en producción	21
Supervisión del estado y el rendimiento del alojamiento	22
Administración de actualizaciones y parches del servidor de juegos	22
Optimización del rendimiento y escalado	23
Solución de problemas y operaciones en tiempo real	24
Herramientas e integración	24
Recursos relacionados	25
Términos clave	25
Introducción	33
Antes de comenzar	33
Selección de la ruta	33
Deseo explorar Amazon GameLift Servers rápidamente	33
Deseo crear una solución de alojamiento personalizada	34
Ruta de aprendizaje recomendada	35
Configura una AWS cuenta	35
Inscríbase en una Cuenta de AWS	37

Creación de un usuario con acceso administrativo	37
Configuración de permisos de usuario para Amazon GameLift Servers	39
Para configurar el acceso mediante programación para usuarios, realice el siguiente procedimiento:	39
Configuración de acceso mediante programación para su juego	41
Ejemplos de permisos de IAM	42
Configuración de un rol de servicio de IAM	47
Obtención de herramientas de desarrollo de	52
Para servidores de juegos	52
Para clientes de juegos	55
Para ver los servicios de back-end de juegos	56
Para la administración de recursos	56
Para Amazon GameLift Servers Realtime	57
Tutorial: Incorporación rápida con el encapsulador de Amazon GameLift Servers	58
Requisitos previos	59
Descripción general de	59
Paso 1: Obtener y crear el encapsulador de servidor de juegos	59
Paso 2: Preparar la compilación del servidor de juegos	60
Paso 3: Configurar la contenedor para la flota	61
Paso 4: Cargar la compilación del servidor de juegos	63
Paso 5: Crea la flota gestionada EC2	63
Paso 6: Crea una sesión de juego y conéctate a ella	65
Paso 7: Gestiona y monitorea tu flota	66
Paso 8: Escala tus servidores de juegos	66
Solución de problemas comunes	68
Siguientes pasos	69
Ejemplos de Amazon GameLift Servers	69
Ejemplo de servidor de juegos personalizado	70
Creación de una solución de alojamiento de juegos	71
Hojas de ruta de desarrollo	72
Hoja de ruta de desarrollo de EC2 administrado	72
Hoja de ruta de desarrollo de contenedores administrados	79
Hoja de ruta de desarrollo de Anywhere	86
Hoja de ruta de desarrollo para el alojamiento híbrido	93
Preparación de un juego para su alojamiento	101
Preparación con el complemento de motor de juegos	102

Integración en un servidor de juegos	193
Empaquetado de una la compilación de servidor de juegos	222
Interacciones de Amazon GameLift Servers	236
Creación de un servicio de backend	243
Autenticación de los jugadores	243
Integración de la funcionalidad de cliente de juego	244
Backend sin servidor	250
WebSocketbackend basado	251
Implementación de flotas de alojamiento	253
Características de las flotas	254
Elección de una opción de alojamiento	255
flotas gestionadas EC2	257
Flotas de contenedores administradas	269
Flotas de Anywhere	293
Creación de una solución de alojamiento híbrida	304
Configuración de la ubicación de las sesiones de juego	307
Características de una cola	308
Prácticas recomendadas	309
Creación de una cola	310
Configuración de la notificación de eventos	314
Personalización de su solución de alojamiento de juegos	318
Compilaciones de servidores de juegos	319
Sesiones de jugador y emparejamientos	333
Ubicaciones de sesión de juego	335
Recursos de alojamiento	351
Administración de una solución de alojamiento de juegos	378
Administración de los recursos de alojamiento de juegos	378
Características de la consola de Amazon GameLift Servers	378
Panel de la consola	379
Compilaciones	381
Flotas	384
Alias	407
Cola de sesiones del juego	410
Preparativos para el lanzamiento del juego	413
Preparativos	414
Preparativos para la prueba	414

Preparación para el lanzamiento	415
Planificación posterior al lanzamiento	415
Administra los recursos de alojamiento de juegos con CloudFormation	416
Prácticas recomendadas	416
Uso de pilas CloudFormation	417
Actualización de las compilaciones	422
DDoSProtección S	425
Ventajas principales	425
Cómo funciona Player Gateway	426
Flujo de tráfico	426
Conceptos clave	426
Supervisar la pasarela del jugador	428
IPv4 y IPv6 compatibilidad	428
Habilita la pasarela de jugadores en las flotas	429
Requisitos previos	429
Modos Player Gateway	429
IPv4 y IPv6 compatibilidad	430
Integra la pasarela del jugador en un juego	430
Integración de back-end	431
Integración con el cliente	433
Realización de una prueba de integración	435
Prácticas recomendadas	435
Pruebas y solución de problemas	436
Preparación para el desarrollo iterativo	436
Creación de un entorno de pruebas basado en la nube	438
Configuración de pruebas locales	442
Configuración de pruebas locales (antiguo)	448
Depuración de problemas relacionados con flotas	457
Problemas al crear una flota	457
Problemas con los procesos del servidor	459
Problemas de eliminación de la flota	460
Conexión remota a instancias de flota	460
Acceso remoto a través de la consola	461
Acceso remoto con la AWS CLI	462
Conexión a una instancia (SDK 5 de servidor)	464
Conexión a una instancia (SDK 4.x, o anterior, de servidor)	466

Visualización de archivos en instancias remotas	469
Escalado	471
Para administrar la capacidad de la flota en la consola, realice el siguiente procedimiento:	472
Establecimiento de los límites de capacidad de alojamiento	472
Para establecer límites de capacidad.	473
Configuración manual de la capacidad de la flota	474
Suspensión del escalado automático	474
Para configurar manualmente la capacidad de la flota	475
Escalado automático de la capacidad de la flota	476
Escalado automático basado en objetivos	477
Escalado automático basado en reglas	479
Administre la ampliación de una flota To/From Zero	484
Comportamiento de escalado	484
Comportamiento de escalado	485
Configuración de Scale Zero To/From	485
Prácticas recomendadas	487
Escalado de flotas de contenedores	487
Supervisión	490
Comparación de métricas	491
Disponibilidad de las métricas por fuente	491
Temas	495
Supervise con CloudWatch	495
Dimensiones de las métricas	496
Métricas de flota	497
DDoMétricas de protección S (puerta de entrada al jugador)	515
Métricas de cola	517
Métricas de FlexMatch	521
Métricas de FleetIQ	525
Supervisión con métricas de telemetría de servidor	528
Ventajas de las métricas de telemetría	529
Antes de empezar	530
Implementación	531
Métricas disponibles	532
Funcionamiento	537
Organización y uso de los paneles	538
Escenarios de supervisión habituales	550

Guía para solucionar problemas	555
Registro de llamadas a la API	558
Amazon GameLift Serversinformación en CloudTrail	558
Descripción de las entradas de los archivos de registro de Amazon GameLift Servers	559
Registro de mensajes del servidor	562
EC2 frente al registro de contenedores	562
Registro para servidores personalizados	565
Seguridad	569
Protección de datos	570
Cifrado en reposo	572
Cifrado en tránsito	572
Privacidad del tráfico entre redes	573
Identity and Access Management	574
Público	574
Autenticación con identidades	574
Administración del acceso con políticas	576
Cómo Amazon GameLift Servers funciona con IAM	578
Ejemplos de políticas basadas en identidades	584
Resolución de problemas	590
AWS políticas gestionadas	592
Registro y supervisión con Amazon GameLift Servers	594
Validación de conformidad	595
Resiliencia	595
Seguridad de la infraestructura	596
Configuración y análisis de vulnerabilidades	597
Prácticas recomendadas de seguridad	598
Mantener actualizados los entornos de tiempo de ejecución de la flota	598
Protección de las configuraciones de puerto	600
Recursos de seguridad adicionales	601
Precios	602
Modelos de precios	602
Generación de estimaciones de precios de	603
Estimación del alojamiento administrado de	603
Estrategias de optimización de costos	607
Prácticas recomendadas	607
Herramientas de uso y costo de los recursos	609

Supervisión del rendimiento	610
Guías de referencia	612
API de servicio (AWS SDK)	612
Administración de los recursos de alojamiento de Amazon GameLift Servers	613
Inicio de sesiones de juego y unión de los jugadores	617
SDK 5.x del servidor	618
Actualizaciones en el SDK 5.x del servidor	619
Migración al SDK 5.x del servidor	621
SDK del servidor de C++	623
SDK del servidor de C#	677
SDK del servidor de Go	724
SDK del servidor de C++ (Unreal): acciones	756
SDK del servidor de las versiones 4 y anteriores	804
SDK del servidor para C++: acciones	805
SDK del servidor para C#: acciones	829
SDK del servidor para Unreal: acciones	851
Eventos de ubicación de sesión de juego	867
Sintaxis de eventos de ubicación	868
PlacementFulfilled	869
PlacementCancelled	871
PlacementTimedOut	872
PlacementFailed	873
Versiones de AMI	874
Ubicaciones de servicio	876
AWS Ubicaciones compatibles	876
Ubicaciones para el alojamiento administrado	882
Ubicaciones para Amazon GameLift Servers Anywhere	884
Ubicaciones para Amazon GameLift Servers FlexMatch	884
Amazon GameLift Servers en China	884
Cuotas y puntos de conexión del servicio	885
Límites de la API de Amazon GameLift Servers	885
Referencia de límites de la API	885
Limitación de nivel de recurso	894
Límites de la API del SDK de Amazon GameLift Servers	895
Señalizadores de pings de UDP	899
Cómo funcionan los señalizadores de pings de UDP	900

Casos de uso habituales de los señalizadores de pings de UDP	900
Obtención de puntos de conexión de señalizador	901
Implementación de mediciones de latencia	902
Ejemplos de código	903
Notas de la versión y versiones del SDK	911
Versiones del SDK	911
Notas de la versión	922
.....	cmlxxvi

¿Qué es Amazon GameLift Servers?

Utilice Amazon GameLift Servers para implementar, operar y escalar servidores dedicados y de bajo costo en la nube para juegos multijugador basados en sesiones. Basado en una infraestructura informática AWS global, Amazon GameLift Servers ayuda a ofrecer servidores de juegos de alto rendimiento y alta fiabilidad, a la vez que escala dinámicamente el uso de los recursos para satisfacer la demanda de los jugadores de todo el mundo.

Características principales de

Alcance global y baja latencia

- Implementa el alojamiento en AWS regiones y Zonas Locales de todo el mundo para llevar tus juegos a nuevos jugadores. Consulte una lista completa de [ubicaciones admitidas por Amazon GameLift Servers](#).
- Ofrezca la latencia más baja posible y una excelente experiencia de jugador, dondequiera que se encuentren sus jugadores.
- Utilice colas de ubicación en múltiples ubicaciones para ubicar a los jugadores de forma dinámica en las sesiones de juego con una experiencia de juego óptima.

Escalado automático para flotas administradas

- Optimice la capacidad de la flota para lograr una mayor rentabilidad al tiempo que mantiene la calidad del servicio.
- Escale la capacidad según la demanda de los jugadores mediante el escalado automático basado en objetivos
- Mantenga un margen de capacidad para dar cabida a la afluencia repentina de jugadores.

Opciones de alojamiento flexibles

Consulte más detalles en [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

- Alojamiento de juegos en la nube, administrado por Amazon GameLift Servers
 - Alojamiento de servidores de juego en la infraestructura de Amazon Elastic Compute Cloud (Amazon EC2). Elija entre los tipos de recursos, incluidas las instancias Graviton optimizadas

para computación (familia C), optimizadas para memoria (familia R) y rentables basadas en ARM.

- Personalice la forma en que Amazon GameLift Servers administra todos los aspectos de su solución de alojamiento de juegos, desde la implementación de servidores de juegos y el inicio de las sesiones de juego hasta el emparejamiento de jugadores y la incorporación de estos a los juegos.
- Utilice flotas con múltiples ubicaciones para agilizar la implementación global de los servidores de juegos.
- Beneficiarse del alojamiento de bajo costo con las [instancias de spot de Amazon EC2](#).
- Implemente servidores de juegos Windows o Linux con alojamiento de EC2 administrado.
- Implemente contenedores basados en Linux en su servidor de juegos con alojamiento de contenedores administrado.
- Alojamiento de juegos administrado por usted, con Amazon GameLift Servers Anywhere
 - Aproveche las capacidades de ubicación y administración de sesiones de juego de Amazon GameLift Servers.
 - Autoadministre las implementaciones y el escalado de los servidores de juegos en su propio hardware, en su infraestructura en las instalaciones local o en otros proveedores de servicios en la nube.
- Alojamiento de juegos híbrido
 - Cree una solución de alojamiento de juegos que utilice una combinación de flotas administradas por Amazon GameLift Servers y flotas de Anywhere que funcionen con las soluciones existentes o admitan la migración.

Administración de las sesiones de juego

- Permita que Amazon GameLift Servers administre los procesos del servidor de juegos por usted, monitoree el estado y la disponibilidad del servidor de juegos y ubique las nuevas sesiones de juego bajo solicitud.
- Utilice las características de ubicación de sesiones de juego para ayudar a los jugadores a encontrar las sesiones de juego que ofrezcan la mejor experiencia de juego posible y unirse a ellas. Puede dejar que Amazon GameLift Servers tome las decisiones o puede personalizar los criterios de ubicación, como el costo, la latencia de los jugadores y la ubicación geográfica.

- Cree colas de ubicación de las sesiones de juego para procesar de forma eficiente las nuevas solicitudes de sesiones de juego. Utilice características para establecer un equilibrio entre la calidad y la velocidad de ubicación que se adapte al juego y a las expectativas de los jugadores.
- Aproveche las características opcionales de sesión de jugador para reservar espacios para los jugadores en las sesiones de juego, validar a los jugadores durante la conexión y controlar la disponibilidad de los espacios. Busque o filtre las sesiones de juego existentes para encontrar espacios disponibles.

Emparejamiento avanzado con FlexMatch

- Añada un sistema de emparejamiento consistente que esté totalmente integrado con su solución de alojamiento de Amazon GameLift Servers.
- Cree reglas de emparejamiento personalizadas basadas en los atributos y preferencias de los jugadores, como el nivel de habilidad o la elección del personaje, y utilice una lógica compleja cuando sea necesario.
- Organice partidas por equipos para favorecer el juego competitivo o cooperativo. Equilibra a los jugadores en la composición de los and/or partidos del equipo.
- Optimice el algoritmo de emparejamiento para equilibrar la calidad de las partidas y los tiempos de espera.
- Permita las partidas de 2 a 200 jugadores simultáneos.
- Permita la reposición de emparejamientos para rellenar los espacios disponibles para los jugadores y mantener las sesiones de juego llenas.

Supervisión integral

- Obtenga métricas en tiempo real sobre el uso y el rendimiento de las sesiones de juego y la actividad de los jugadores, así como del rendimiento del alojamiento y del estado de las flotas administradas.
- Intégrelo con Amazon CloudWatch para obtener alertas y paneles.
- Acceda a los registros del servidor y a la monitorización de eventos con fines de solución de problemas.
- Analice los datos de rendimiento y de uso para optimizar el escalado (flotas administradas) o la asignación de recursos (flotas autoadministradas).

Personalización y extensibilidad

- Personalice la lógica de ubicación de las sesiones de juego, lo que incluye establecer prioridades en cuanto a latencia, costo y ubicación.
- Para el alojamiento administrado, elija sus recursos de alojamiento de juegos y proporcione instrucciones para lanzar y ejecutar los servidores de juegos en cada instancia. Para los contenedores administrados, refine la asignación de recursos a contenedores.
- Defina sus propias reglas de emparejamiento.
- Implemente estrategias personalizadas para el escalado de la capacidad.
- Añade código de servidor de juegos para integrarlo con tus AWS servicios existentes, como bases de datos o herramientas de almacenamiento de contenido.
- Cree un servicio de backend con características personalizadas para administrar las solicitudes de incorporación de los jugadores (por ejemplo, las partidas de jugadores), gestionar la autenticación o añadir una lógica y persistencia personalizadas.

Integración con el AWS ecosistema

- Utilice servicios como Amazon DynamoDB, Amazon Simple Storage Service y Amazon Aurora DSQL para la persistencia del estado del juego.
- Implemente Amazon Cognito para la autenticación de los jugadores.
- Procese el análisis de juegos con Amazon Kinesis y Amazon S3.
- Añada un chat de voz con Amazon Chime SDK.
- Cree funciones de juego personalizadas con AWS Lambda Amazon API Gateway.

Cómo trabajar con Amazon GameLift Servers

Utilice estas herramientas para trabajar con Amazon GameLift Servers.

AWS CLI

Utilice AWS Command Line Interface (AWS CLI) para realizar llamadas al AWS SDK, incluida la API de servicio para Amazon GameLift Servers. Consulte [Introducción a la AWS CLI](#) en la Guía del usuario de AWS Command Line Interface .

Consola de Amazon GameLift Servers

Utilice la [Consola de administración de AWS de Amazon GameLift Servers](#) para configurar los recursos, administrar las implementaciones de servidores de juegos y monitorizar las métricas de rendimiento y de uso. La consola de Amazon GameLift Servers ofrece una alternativa a la GUI para administrar los recursos mediante programación o con la AWS CLI.

Amazon GameLift Servers SDKs

Amazon GameLift Servers SDKs Contienen las bibliotecas necesarias para establecer la comunicación entre los clientes, los servidores y los servicios del juego y el Amazon GameLift Servers servicio. Para obtener más información, consulte [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#).

AWS CloudFormation

Se utilizan AWS CloudFormation para modelar, aprovisionar y administrar AWS los recursos de su solución de alojamiento de juegos tratando la infraestructura como código. Cree plantillas que describan los recursos y CloudFormation automatice las tareas de configuración e implementación de los recursos en las ubicaciones que especifique.

opciones de alojamiento de juegos de Amazon GameLift Servers

Amazon GameLift Servers ofrece diversas opciones para el alojamiento de servidores de juegos multijugador basados en sesiones. Tanto si desea configurar un alojamiento en la nube administrado por Amazon GameLift Servers como si desea incorporar recursos de alojamiento de los que ya disponga, puede utilizar Amazon GameLift Servers para compilar la solución de alojamiento que necesita para sus jugadores. Para obtener más información sobre cómo debería ser una solución de alojamiento de juegos, consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#).

Temas

- [EC2 administrado](#)
- [Contenedores administrados](#)
- [Alojamiento con Anywhere](#)
- [Alojamiento híbrido](#)

EC2 administrado

Descargue el trabajo de administrar sus servidores de juegos de producción a Amazon GameLift Servers con el alojamiento de EC2 administrado. Benefíciense de las optimizaciones para servidores de juegos multijugador sin renunciar al alto rendimiento y la fiabilidad de Amazon Elastic Compute Cloud (Amazon EC2) y a la infraestructura de computación global de AWS .

[Primeros pasos para crear una solución de alojamiento de EC2 administrada por Amazon GameLift Servers](#)

Características

- Aloje servidores de juegos multijugador que se ejecuten en sistemas operativos Amazon Linux o Windows Server.
- Utilice Amazon GameLift Servers para implementar y administrar sus servidores de juegos personalizados en recursos de alojamiento de Nube de AWS. Elija el tipo de hardware y dónde implementarlo, y configure detalles adicionales solo cuando sea necesario.
- Configure el administrador de procesos en tiempo de ejecución para mantener automáticamente los procesos del servidor de juegos según corresponda para alojar las sesiones de juego.
- Configure una monitorización de estado personalizada para ayudar a Amazon GameLift Servers a detectar y resolver problemas en servidores de juegos con un rendimiento deficiente.
- Benefíciense de la supervisión del rendimiento de Amazon GameLift Servers. Monitorice métricas como el rendimiento del hardware y los ciclos de vida de los procesos del servidor. También puede descargar y almacenar los registros de las sesiones de juego.
- Utilice Amazon GameLift Servers para administrar el acceso entre procesos del servidor de juegos y otros recursos de AWS .
- Para el alojamiento de producción, automatice las implementaciones y la administración del alojamiento de juegos con AWS CloudFormation plantillas y AWS Cloud Development Kit (AWS CDK) Aproveche las herramientas y los servicios de integración y entrega continuas (CI/CD), como AWS CodePipeline.

Contenedores administrados

Amazon GameLift Servers proporciona una solución de alojamiento en la nube completa para servidores de juegos en contenedores, de modo que pueda disfrutar de las principales ventajas del uso de contenedores, como la portabilidad, la agilidad y la tolerancia a errores. Al igual que con

el alojamiento de EC2 administrado, el alojamiento en contenedores administrados implementa y ejecuta los contenedores en instancias de Amazon EC2.

[Comience a desarrollar una solución de alojamiento administrada por Amazon GameLift Servers para su servidor de juegos en contenedores.](#)

Características

- Desarrolle una arquitectura personalizada con contenedores ligeros para ejecutar su compilación del servidor de juegos, las dependencias y el software auxiliar.
- Utilice las herramientas de Docker para crear una imagen de contenedor basada en Linux. Almacene imágenes en un repositorio de Amazon Elastic Container Registry (Amazon ECR) para su implementación.
- Utilice Amazon GameLift Servers para implementar y administrar sus servidores de juegos personalizados en recursos de alojamiento de Nube de AWS. Elija el tipo de hardware y dónde implementarlo, y configure detalles adicionales solo cuando sea necesario.
- Administre el ciclo de vida de la flota de alojamiento con herramientas para modelar las versiones de los servidores de juegos. Implemente actualizaciones del servidor de juegos y otros cambios de configuración según sea necesario.
- Haga un seguimiento del rendimiento del alojamiento de juegos con métricas de rendimiento específicas de los contenedores. Supervise el estado de los recursos de la flota mediante métricas de hardware.
- Administre los recursos de la flota de contenedores mediante AWS CloudFormation plantillas para Amazon GameLift Servers.

Alojamiento con Anywhere

Con el alojamiento de Anywhere, puede aprovechar las características de administración de sesiones de juego de Amazon GameLift Servers, incluido el emparejamiento y las métricas de las sesiones de juego, para alojar los servidores de juegos donde desee. Autoadministre las implementaciones de sus servidores de juegos, la supervisión del estado de los servidores de juegos y el escalado de la capacidad.

Note

Las flotas de Anywhere son especialmente útiles como entornos de prueba para permitir un desarrollo de juegos rápido e iterativo. Configure una flota de Anywhere para su propia

estación de trabajo local o para un recurso de alojamiento basado en la nube. Para obtener más información sobre cómo probar Anywhere, consulte [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#).

[Comience a desarrollar una solución de alojamiento de Amazon GameLift Servers Anywhere para su juego.](#)

Características

- Implemente servidores de juegos en su propio hardware, en su infraestructura en las instalaciones o en otros proveedores de servicios en la nube. Utilice el hardware disponible cerca de sus jugadores, en cualquier lugar.
- Sírvasse de las características de administración de sesiones de Amazon GameLift Servers para supervisar la disponibilidad de los servidores de juegos, iniciar nuevas sesiones de juego e incorporar a los jugadores a espacios de sesiones de juego.

Alojamiento híbrido

Para el alojamiento en producción, puede optar por un enfoque híbrido, con flotas de Anywhere para los recursos de alojamiento que suministre y administre, y flotas administradas de Amazon GameLift Servers de reposición cuando sea necesario. Una solución híbrida utiliza los mismos procesos para iniciar las sesiones de juego y conectar a los jugadores a fin de ofrecer una experiencia de juego perfectamente integrada. Puede supervisar y recopilar métricas de uso de todos los alojamientos, independientemente del tipo de recurso.

Estos son algunos escenarios habituales en los que conviene utilizar una solución híbrida:

- Ampliar una solución de alojamiento a Nube de AWS. Complemente las capacidades de su solución de alojamiento existente (hardware en las instalaciones u otro alojamiento basado en la nube) añadiendo el alojamiento administrado de Amazon GameLift Servers. Con el alojamiento administrado, puede aumentar su capacidad de alojamiento o añadir capacidad de ampliación para escalar verticalmente de forma rápida y pagar solo por los recursos cuando los necesite. También puede aprovechar la presencia global del servicio de Amazon GameLift Servers para llegar a más jugadores de todo el mundo y ofrecer la experiencia multijugador de baja latencia que esperan.
- Prepararse para la migración a un alojamiento basado en la nube. Si tiene planes de migrar a Nube de AWS, una solución de alojamiento híbrida es una forma viable de hacer la transición

de una forma gradual, tal y como necesite. Le permite crear la solución que necesita de forma inmediata mientras se prepara para el futuro.

- Menor latencia para los jugadores en ubicaciones distintas a las que Amazon GameLift Servers presta servicio. Si ya utiliza el alojamiento administrado por Amazon GameLift Servers, es posible que necesite ayudar a los jugadores en determinadas situaciones inusuales. Por ejemplo, puede que desee llegar mejor a los jugadores que se encuentran en ubicaciones inusualmente remotas o ayudar a los jugadores con necesidades de acceso especiales. Añada ubicaciones de alojamiento personalizadas y utilice Amazon GameLift Servers Anywhere para administrar esas ubicaciones junto con sus recursos de alojamiento administrados.

-

[Comience a desarrollar una solución de alojamiento híbrida de Amazon GameLift Servers para sus juegos.](#)

Cómo funciona el alojamiento con Amazon GameLift Servers

Amazon GameLift Servers es un servicio que proporciona servidores dedicados y de bajo costo, aprovisionamiento de infraestructura, escalado y administración de sesiones para juegos multijugador basados en sesiones. El servicio ofrece herramientas y características flexibles que puede personalizar para sus juegos o utilizarlas en colaboración con sus propios sistemas. Aproveche el alojamiento administrado por Amazon GameLift Servers para poder centrarse en crear experiencias de juego, en lugar de administrar las operaciones del servidor.

En este tema se describen los componentes básicos de una solución completa de alojamiento con Amazon GameLift Servers y cómo estos funcionan juntos para ofrecer sesiones de juego multijugador a los jugadores.

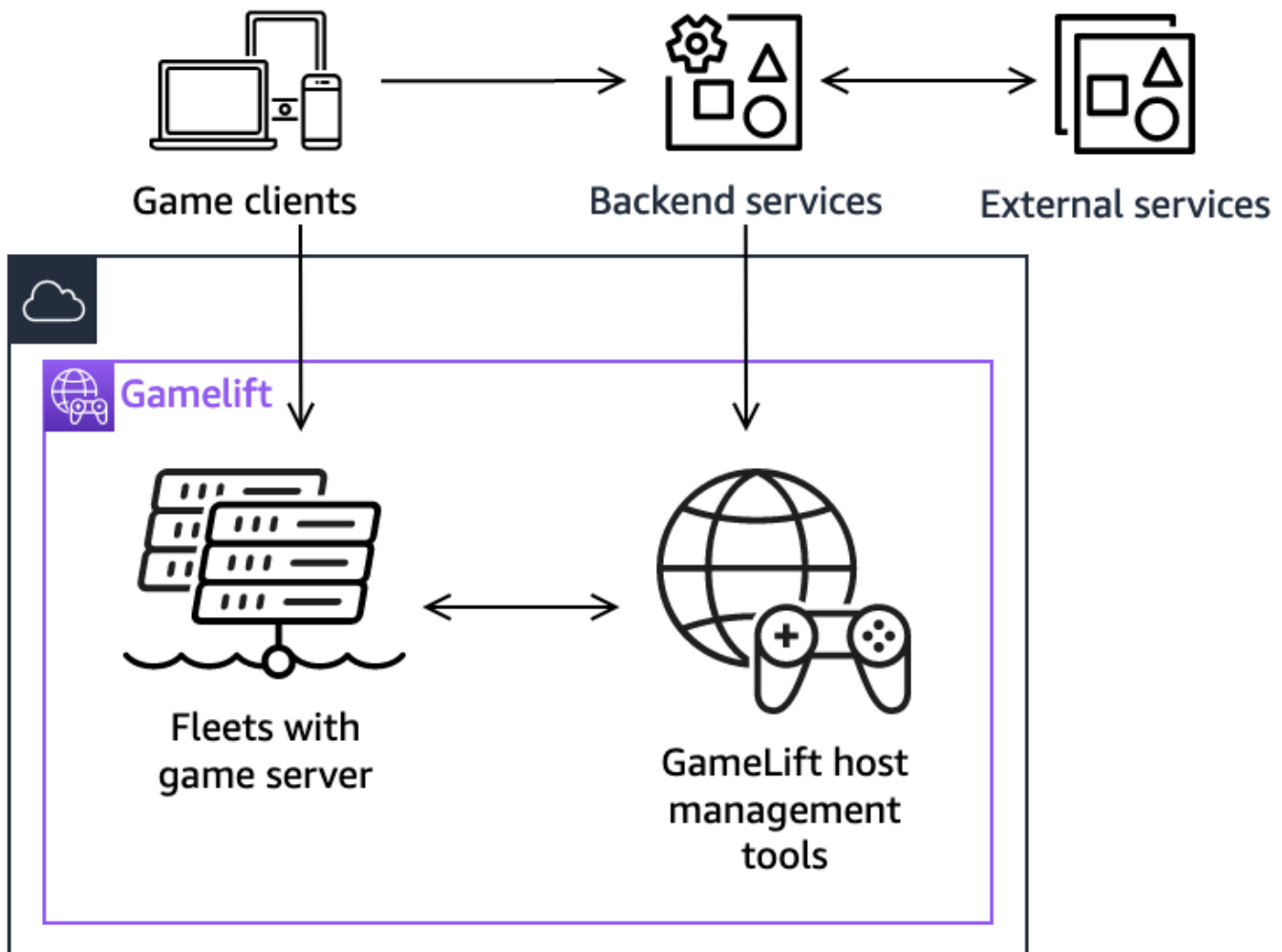
Componentes centrales de una solución de alojamiento de juegos

Una solución completa de alojamiento de Amazon GameLift Servers consta de varios componentes esenciales que funcionan juntos para ofrecer experiencias de juego multijugador. Algunos de estos componentes los debe compilar y administrar usted mismo, mientras que otros los proporciona Amazon GameLift Servers y se administran en función de sus opciones de configuración.

Comprender cómo interactúan estos componentes y sus respectivas funciones es fundamental para desarrollar una solución de alojamiento de juegos eficaz. Los componentes incluyen:

- El cliente del juego se ejecuta en los dispositivos de los jugadores.

- El servicio de backend permite a los clientes de juego comunicarse con Amazon GameLift Servers para obtener información sobre las sesiones de juego y unirse a las partidas.
- El software de servidor de juegos funciona con recursos de alojamiento de juegos y aloja sesiones de juego para un grupo de jugadores.
- El sistema de ubicación inicia las sesiones de juego y empareja a los jugadores con las partidas.
- Las flotas de alojamiento de juegos proporcionan servidores de juegos en una o más ubicaciones geográficas.
- El sistema de administración de alojamiento de juegos supervisa el estado del alojamiento de juegos y administra la capacidad.



Cliente de juegos

Un cliente de juego es el software de juego que se ejecuta en el dispositivo de un jugador. Inicia la ubicación de jugadores en una sesión de juego comunicándose con un servicio de backend y se conecta directamente a un servidor de juegos para participar en el juego.

Funciones clave para el alojamiento de juegos

- Enviar las solicitudes de incorporación al servicio de backend. Incluir datos relevantes de los jugadores y datos de las sesiones de juego como parte de la solicitud de incorporación.
- Recopilar datos de latencia para el cliente de juego e incluir esa información en la solicitud de incorporación.
- Recibir información de conexión de las sesiones de juego desde el servicio de backend y utilizarla para la conexión a un servidor de juegos.
- Gestionar los escenarios de conexión de sesiones, como la verificación de jugadores, la aceptación de emparejamientos para Amazon GameLift Servers FlexMatch, la interrupción de sesiones, o las desconexiones.

Quién es el responsable

Usted añade la funcionalidad de alojamiento de juegos a su software de cliente de juegos y configura la comunicación con el servicio de backend.

Servicio de backend

El servicio de backend es una capa de coordinación entre los clientes de juego y el servicio Amazon GameLift Servers. Controla todas las comunicaciones con el servicio para solicitar la ubicación de las sesiones de juego y recuperar la información sobre las sesiones de juego y las sesiones de jugador. El uso de un servicio de backend es una práctica recomendada que mantiene una comunicación segura con el servicio en nombre de los clientes de juego y evita tener que compartir credenciales y permisos confidenciales de AWS con los clientes de juego.

El servicio de backend implementa una estrategia de agrupamiento de jugadores para las sesiones de juego. Dispone de una gran flexibilidad en cuanto a cómo se asigna a los jugadores a las sesiones de juego. Puede configurar un explorador de sesiones de juego para los jugadores o permitir los grupos definidos por los jugadores. También puede agrupar las solicitudes de sesión de juego a medida que vayan llegando, formar grupos de jugadores (posiblemente con una clasificación

previa) e iniciar una sesión de juego para cada grupo. Con el sistema de emparejamiento de FlexMatch, puede formar emparejamientos de jugadores y reponer emparejamientos ya existentes.

Funciones clave para el alojamiento de juegos

- Autenticar las comunicaciones de los clientes de juego.
- Realizar solicitudes a Amazon GameLift Servers para iniciar nuevas sesiones de juego, obtener información de las sesiones de juego o unirse a sesiones de juego existentes. Incluir los datos relevantes de las sesiones de juego y de los jugadores en las solicitudes según sea necesario.
- Si lo desea, crear solicitudes de sesiones de jugadores para supervisar más de cerca la disponibilidad de las sesiones de juego. Use las sesiones de los jugadores para reservar espacios para las sesiones de juego, validar a los jugadores cuando se conecten y hacer un seguimiento de las desconexiones de los jugadores.
- Responda a los clientes de juego con los detalles de conexión de las sesiones de juego y otra información solicitada.

Quién es el responsable

Usted se encarga de crear el servicio de backend en un entorno administrado por usted para satisfacer los requisitos del juego e implementar la forma en que los jugadores entran en los juegos. Integre el AWS SDK para realizar llamadas a la API del Amazon GameLift Servers servicio. Si lo desea AWS Lambda, cree su servicio de backend AWS con servicios como Amazon Simple Storage Service (Amazon S3) y Amazon Cognito.

Servidor de juegos

El servidor de juegos es un software de servidor personalizado que administra el estado del juego, procesa las acciones de los jugadores y sincroniza el juego entre varios jugadores conectados. El servidor de juegos mantiene la comunicación con el servicio Amazon GameLift Servers para administrar el alojamiento de las sesiones de juego.

Funciones clave para el alojamiento de juegos

- Comuníquese con Amazon GameLift Servers para:
 - informar del estado (listo para alojar sesiones, listo para aceptar jugadores, estado general).
 - responder a llamadas de servicio (iniciar o finalizar sesiones de juego).
- Administre el ciclo de vida de las sesiones de juego para alojar una sesión de juego a la vez por proceso.

- Coordínelo con otros AWS servicios para obtener una funcionalidad adicional.
- Si lo desea, valide las conexiones de nuevos jugadores.

Quién es el responsable

Usted crea el software de servidor de juegos. Usted integra el SDK del servidor para Amazon GameLift Servers y añade funcionalidad para establecer una conexión con el servicio y permitir la administración de las sesiones del juego.

Flota de alojamiento de juegos

Una flota de alojamiento es un conjunto de recursos de computación que ejecutan sus servidores de juegos. Los recursos de la flota se pueden distribuir en varias ubicaciones geográficas para ofrecer a los jugadores una experiencia de juego de baja latencia, estén donde estén. Cada recurso de flota ejecuta uno o más procesos del servidor del juego, que se comunican directamente con Amazon GameLift Servers ellos. Cada proceso del servidor de juegos puede alojar una sesión de juego a la vez.

Las características y la funcionalidad de la flota varían según el tipo de alojamiento de la flota. Las flotas gestionadas despliegan recursos en ellas Nube de AWS y son gestionadas por Amazon GameLift Servers. Las flotas Anywhere son recursos de computación proporcionados por el cliente que se administran desde fuera de Amazon GameLift Servers.

Funciones clave para el alojamiento de juegos

- Aprovisiona recursos de alojamiento de servidores de juegos.
 - Las flotas administradas implementan instancias Amazon EC2 basadas en la nube con una amplia gama de ajustes de configuración. La configuración de la flota determina la potencia de computación de cada instancia de la flota, la ubicación física de las instancias y otros detalles. Las flotas administradas añaden o eliminan instancias de EC2 en respuesta al escalado de la capacidad
 - Las implementaciones de flotas de Anywhere se autoadministran. La flota puede consistir en hardware físico u otros recursos en la nube y configurarse según sea necesario.
- Instale el entorno de tiempo de ejecución y el software del servidor de juegos.
 - Las instancias de flota de Amazon EC2 administradas se implementan con la compilación de servidor de juegos y una imagen de máquina de Amazon (AMI) con un entorno de tiempo de ejecución compatible.

- Las instancias de flotas de contenedores administradas se implementan con la versión del servidor de juegos y una AMI optimizada para contenedores con herramientas de Docker y otros componentes para que funcionen con Amazon ECS.
- En cualquier lugar, las flotas se implementan con el sistema operativo y el software de servidor de juegos que elija.
- Gestione el ciclo de vida de los procesos del servidor de juegos. Aplica instrucciones de tiempo de ejecución preconfiguradas para iniciar y detener los procesos en cada recurso de la flota.

Quién es el responsable

Esto depende del tipo de alojamiento de la flota:

- Flotas gestionadas de Amazon EC2: subes la versión de tu servidor de juegos Amazon GameLift Servers, que la almacena para desplegarla en las instancias de la flota. Amazon GameLift Servers proporciona la infraestructura de la flota. Nube de AWS Configura la flota y le indica cómo ejecutar los servidores del juego en cada instancia de la flota.
- Flotas de contenedores gestionadas: empaquetas las instrucciones de compilación y tiempo de ejecución del servidor de juegos en una imagen de contenedor para guardarla en Amazon ECR para su implementación. Amazon GameLift Servers proporciona la infraestructura de la flota. Nube de AWS Usted proporciona una arquitectura de contenedores y configura la flota para alojar sus contenedores.
- Flotas en cualquier lugar: usted aprovisiona toda la infraestructura y administra la implementación del software del servidor. Crea las flotas de Anywhere con las que conectar tus recursos activos de alojamiento de juegos con Amazon GameLift Servers.

Sistema de ubicación de sesiones de juego

El sistema de ubicación de las sesiones de juego localiza los servidores de juego disponibles para alojar nuevas sesiones de juego. El sistema utiliza información en tiempo real sobre la disponibilidad del servidor del juego para tomar decisiones de ubicación óptimas.

En Amazon GameLift Servers, las colas son el mecanismo principal para la ubicación de las sesiones de juego. Una cola de sesiones de juego utiliza algoritmos, que puede configurar, para colocar las sesiones de juego y obtener el mejor resultado posible. Puede priorizar las ubicaciones en función de factores como el menor costo de alojamiento y la menor latencia de jugadores, así como

configurar una cola para buscar en varias ubicaciones geográficas. Como alternativa a las colas, puede designar una flota específica para alojar las sesiones de juego.

Funciones clave para el alojamiento de juegos

- Procesa las solicitudes de ubicación de sesiones de juego recibidas del servicio de backend.
- Realice las ubicaciones en función de la información en tiempo real sobre la disponibilidad de los recursos de alojamiento.
- Usa los datos de latencia de los jugadores y otros datos para priorizar las opciones de ubicación (solo colas)
- Solicita los procesos del servidor del juego para iniciar nuevas sesiones de juego.
- Actualice la información de la conexión de la sesión de juego cuando la sesión de juego esté lista para aceptar jugadores.
- Si lo desea, puede configurar el emparejamiento de FlexMatch para crear partidas de jugadores y solicitar la ubicación de las sesiones de juego para las partidas.

Quién es el responsable

Amazon GameLift Servers suministra el sistema de ubicación. El comportamiento de ubicación se configura mediante la forma en que el servicio de backend realiza las solicitudes de ubicación y, opcionalmente, configurando las colas de las sesiones de juego y los emparejadores de FlexMatch. Usa la Amazon GameLift Servers consola, el AWS SDK o el AWS CLI para crear y configurar colas y emparejadores para tu juego.

Sistema de gestión de alojamiento de juegos

El sistema de gestión del alojamiento de juegos es la columna vertebral operativa que coordina y supervisa todos los aspectos de su solución de alojamiento de juegos. Este sistema proporciona la inteligencia y la automatización que hacen posibles las capacidades de colocación y escalado de Amazon GameLift Servers.

Funciones clave para el alojamiento de juegos

- Realiza un seguimiento del estado y la disponibilidad en tiempo real de los procesos de los servidores de juegos, las sesiones de juego y las sesiones de los jugadores en todas las flotas para facilitar la ubicación de las sesiones de juego y el escalado automático de la capacidad.
- Supervisa el estado y el rendimiento de la flota.
- Recopile y analice las métricas de actividad del alojamiento de juegos.

- Proporcione herramientas de escalado de capacidad, incluido el escalado automático en función de la demanda de los jugadores y la utilización de la flota.
- Gestione las actualizaciones del software del servidor de juegos y las instrucciones de tiempo de ejecución.

Quién es el responsable

- **Flotas gestionadas:** Amazon GameLift Servers proporciona un seguimiento de la disponibilidad, métricas sobre la actividad de alojamiento de juegos y el rendimiento del hardware, y herramientas de escalado de la capacidad. Utilizas AWS las herramientas disponibles para gestionar las actualizaciones del software del servidor de juegos, modificar las instrucciones de tiempo de ejecución y personalizar el uso de las métricas (por ejemplo, configurarlas CloudWatch para la monitorización). Puede configurar una política de escalado personalizada y modificarla según sea necesario.
- **Flotas de Anywhere:** Amazon GameLift Servers proporciona un seguimiento de la disponibilidad y métricas de la actividad de alojamiento de juegos. Tú gestionas los cambios en la configuración de la flota, incluidas las actualizaciones del software del servidor de juegos y las instrucciones de tiempo de ejecución. Usted crea sistemas para monitorear el rendimiento de la flota y administrar el escalado de la capacidad de la flota.

Cómo funcionan juntos los componentes

Cuando se implementa un recurso de computación de alojamiento de juegos

- **Software de servidor de juegos instalado:** el proceso se instala con un entorno de tiempo de ejecución y la compilación del servidor de juegos
- **Lanzamientos del servidor de juegos:** se lanza al menos una instancia del ejecutable del servidor de juegos, con parámetros de inicio opcionales, en el equipo.
- **El servidor de juegos se conecta a Amazon GameLift Servers:** como parte de sus acciones de inicio, el proceso del servidor de juegos llama al SDK del servidor para inicializar una conexión con el servicio.
- **El servidor de juegos informa que está listo para albergar una sesión de juego:** el proceso del servidor de juegos completa las acciones de inicio y, a continuación, llama al SDK del servidor para informar de que está listo. Comienza a informar sobre el estado de salud en función de su configuración.

- Amazon GameLift Servers monitoriza la disponibilidad: el servicio registra la disponibilidad del servidor del juego en función de la ubicación de la sesión de juego. También actualiza las métricas de las flotas, las instancias y las sesiones de juego para hacer un seguimiento del uso y la capacidad.
- El servidor del juego espera a que se le asigne una sesión de juego: el proceso del servidor del juego mantiene su estado inactivo mientras espera a que se le solicite iniciar una sesión de juego.

Cuando un jugador quiere unirse a una partida

- El jugador inicia una partida: un jugador inicia tu cliente de juego, se autentica en tu servicio de backend y solicita unirse a una partida.
- El servicio de backend solicita una sesión de juego: su servicio de backend llama a Amazon GameLift Servers para buscar o crear una sesión de juego adecuada, en función de su estrategia de agrupamiento de jugadores. La solicitud puede incluir datos del jugador o del juego para usarlos en la sesión de juego.
- Amazon GameLift Servers coloca la sesión de juego: al iniciar una nueva sesión de juego, el sistema de ubicación identifica una ubicación óptima para albergar la sesión y selecciona un proceso de servidor de juego disponible. El servicio solicita al proceso seleccionado que inicie una nueva sesión de juego y transmite los datos del jugador o del juego. Si el emparejamiento de FlexMatch está en uso, el emparejador primero crea una partida y, a continuación, solicita la ubicación de la partida.
- El servidor del juego inicia la sesión de juego: el servidor del juego inicia los pasos para iniciar una sesión de juego. Una vez completado, informa a Amazon GameLift Servers de que está listo para aceptar conexiones de jugadores.
- Información de conexión enviada al cliente del juego: cuando el servidor del juego actualiza su estado, Amazon GameLift Servers proporciona la información de conexión de la sesión del juego. El servicio de backend recibe esta información y la entrega al cliente de juegos.
- El jugador se conecta a la sesión de juego: el cliente de juego utiliza la información de conexión para conectarse directamente al servidor de juegos y empezar a jugar.
- Amazon GameLift Servers supervisa el estado de la sesión de juego: el proceso del servidor de juegos informa sobre el estado general, el estado de conexión opcional del jugador y el estado de la sesión de juego para hacer un seguimiento de la disponibilidad de las sesiones de juego en curso.
- El proceso del servidor del juego se cierra: el proceso del servidor del juego finaliza la sesión de juego, informa del estado y, a continuación, se apaga automáticamente.

Amazon GameLift Servers y la experiencia del jugador

Los jugadores esperan experiencias de juego multijugador rápidas, justas y fiables. Amazon GameLift Servers ofrece estas experiencias al proporcionar a los desarrolladores de juegos herramientas que mejoran directamente la forma en que los jugadores se conectan a los juegos multijugador e interactúan con ellos. En este tema se describen las principales ventajas que ofrecen las características de Amazon GameLift Servers a los jugadores.

Proporcione un juego rápido a los jugadores de todo el mundo

Los jugadores quieren conectarse a servidores cercanos con un retraso mínimo. Amazon GameLift Servers garantiza experiencias de conexión óptimas mediante una ubicación inteligente y una infraestructura global:

- Los jugadores disfrutan de un rendimiento uniforme en todo el mundo: organiza los juegos en AWS las regiones y Zonas Locales disponibles en todo el mundo para minimizar la distancia entre los jugadores y los servidores de juego (consulta [Ubicaciones de servicio de Amazon GameLift Servers](#)). Puedes elegir entre una amplia gama de tipos de EC2 instancias de Amazon en estas ubicaciones para encontrar los mejores recursos informáticos para alojar tus cargas de trabajo de juegos.
- Los jugadores se conectan a los mejores servidores disponibles: Amazon GameLift Servers ubica a los jugadores en las sesiones de juego con la latencia más baja posible mediante la recopilación de datos de la red de los clientes de juego y la búsqueda de servidores disponibles en múltiples ubicaciones. Utilice señalizadores de pings de UDP para recopilar datos de latencia precisos. Configure políticas de latencia para equilibrar la calidad de la conexión con tiempos de espera aceptables para los jugadores.
- Los jugadores de diferentes regiones pueden jugar juntos: las ubicaciones de las sesiones de juego permiten equilibrar las sesiones de juego con jugadores de grupos regionales y, al mismo tiempo, mantener una latencia aceptable. Los jugadores de las regiones menos pobladas pueden encontrar partidas sin sacrificar la calidad de la conexión.

Ayude a los jugadores a unirse a los juegos rápidamente

Los jugadores quieren un acceso rápido a los juegos independientemente de la demanda. Amazon GameLift Servers proporciona una serie de herramientas de escalado de capacidad que le ayudan a garantizar que los jugadores puedan encontrar juegos durante los periodos de mayor y menor actividad:

- Los jugadores pueden encontrar servidores disponibles fácilmente: con las herramientas de escalado automático, puede asegurarse de que los recursos de alojamiento estén listos cuando los necesite. El seguimiento de objetivos administra un margen de capacidad flexible para anticiparse a los patrones de demanda de los jugadores para el juego y escalarlos de manera proactiva.
- Los jugadores disfrutan de un rendimiento fluido durante los eventos especiales: combine el escalado automático con técnicas de escalado manual para manejar las afluencias repentinas tras eventos planificados, como actualizaciones del juego, campañas de marketing o torneos.
- Los jugadores no tienen que esperar a que los servidores se inicien: en todos los recursos de alojamiento de juegos, los servidores de juegos están preiniciados; así están siempre listos para alojar nuevas sesiones en cuanto los jugadores se unen.

Idee formas creativas de agrupar a los jugadores en las sesiones de juego

Los jugadores quieren competir o colaborar con otros jugadores en términos justos y equilibrados. Amazon GameLift Servers ofrece una serie de opciones para agrupar a los jugadores en sesiones de juego:

- Los jugadores se agrupan para optimizar la experiencia de juego: puede manejar el agrupamiento de jugadores de la forma que mejor se adapte al juego mediante una serie de características. Permita a los jugadores buscar o explorar las sesiones de juego o agrupar las solicitudes de los jugadores a su llegada en función del nivel de habilidad, la latencia y otros atributos. Puede manejar la agrupación de jugadores por su cuenta o puede implementar el emparejamiento de FlexMatch con reglas de juego personalizables.
- Los jugadores pueden unirse a las partidas activas: cree grupos de jugadores para iniciar nuevas sesiones de juego o utilice las sesiones de jugador para monitorizar los espacios disponibles en las sesiones existentes y añadir nuevos jugadores. Con FlexMatch, rellene automáticamente las sesiones con jugadores correctamente emparejados.
- Los amigos pueden jugar juntos como equipo: la compatibilidad con equipos de jugadores permite que grupos de amigos jueguen juntos. Las sesiones de juego pueden llenar los espacios restantes con otros jugadores o restringir el acceso. Con FlexMatch, los amigos pueden formar equipos.
- Los jugadores pueden incorporar datos personalizados a las sesiones de juego: incluya información sobre el juego y los jugadores en las solicitudes de sesiones de juego y de sesiones de jugador. La información se transmite al servidor de juegos para su uso en una sesión de juego.
- Los jugadores no esperan indefinidamente a que haya emparejamientos perfectos: FlexMatch le permite flexibilizar los requisitos de emparejamiento con el paso del tiempo para evitar que los

jugadores se sientan frustrados por esperar demasiado en la cola. Crea reglas de flexibilización progresiva que prioricen los aspectos más importantes de la calidad de los emparejamientos.

- Los jugadores pueden optar por aceptar o rechazar los emparejamientos: FlexMatch le permite personalizar los flujos de trabajo para permitir que los jugadores acepten o rechacen los posibles emparejamientos.

Ofrezca conexiones fiables durante todo el juego

Los jugadores esperan un juego coherente, sin interrupciones ni desconexiones. Amazon GameLift Servers ofrece una administración y protección integrales de las sesiones:

- Los jugadores disfrutan de un rendimiento de alta calidad en AWS la nube: con el alojamiento gestionado por Amazon GameLift Servers, los jugadores disponen de servidores de juegos rápidos y fiables que se ejecutan en una infraestructura AWS informática.
- Valide a los jugadores cuando se conecten: al habilitar las sesiones de jugador del juego, puede hacer que Amazon GameLift Servers reserve espacios para los jugadores en las sesiones de juego y validar a los jugadores cuando se conecten.
- Los jugadores pueden volver a conectarse y mantener el progreso del juego: Amazon GameLift Servers permite la reconexión tanto en caso de interrupciones del lado del servidor como del lado del cliente.
- Las partidas de los jugadores están protegidas contra la finalización: la característica opcional de protección de las sesiones de juego evita que las sesiones activas finalicen durante los eventos de escalado u otras interrupciones.

Mejore la experiencia de los jugadores basándose en datos reales

Los jugadores esperan que los juegos funcionen de manera coherente y que mejoren con el tiempo. Amazon GameLift Servers proporciona supervisión y análisis exhaustivos que ayudan a optimizar la experiencia de los jugadores:

- Los jugadores experimentan menos interrupciones: la supervisión en tiempo real del rendimiento de la flota, las sesiones de juego y la actividad de los jugadores permite identificar los problemas antes de que afecten de forma significativa al juego. Personalice los cierres y migraciones de las sesiones de juego para que se realicen sin incidencias.
- Los jugadores se benefician de una supervisión proactiva del estado de los servidores: Amazon GameLift Servers proporciona una supervisión continua del estado de los servidores y reemplaza

automáticamente los servidores de juegos defectuosos para minimizar el impacto de los fallos de hardware o software. Configure parámetros de comprobación de estado para que se ajusten a los distintos requisitos del juego.

- Los jugadores obtienen mejoras del juego basadas en datos: aproveche los registros de las sesiones de juego y los análisis detallados para revelar patrones de comportamiento de los jugadores y rendimiento del servidor. Añada soporte de registro personalizado para eventos específicos del juego.

Integre las funciones mejoradas del juego con otros AWS servicios

Los jugadores quieren funciones integradas, como el chat de voz, la autenticación segura y el progreso continuo. Amazon GameLift Servers se puede integrar con otros AWS servicios para ofrecer experiencias de juego completas:

- Los jugadores se benefician de una autenticación simplificada: la integración de Amazon Cognito admite varios proveedores de identidad y métodos de autenticación, a la vez que mantiene seguras las identidades de los jugadores en todas las sesiones de juego, con procesos de inicio de sesión simplificados para los jugadores que regresan.
- Los jugadores pueden conservar su progreso en el juego entre sesiones: la integración de Amazon DynamoDB se puede utilizar para almacenar el progreso de los jugadores, los inventarios y los datos persistentes con un acceso de alto rendimiento que garantiza un impacto mínimo en el juego y, al mismo tiempo, permite la continuidad entre sesiones.
- Los jugadores se benefician de información analítica: la integración de Amazon Kinesis y Amazon Simple Storage Service (Amazon S3) procesa el análisis del juego para recopilar información sobre el comportamiento y las preferencias de los jugadores, lo que permite realizar análisis en tiempo real que ayudan a los desarrolladores a adaptarse a los cambios en los patrones de los jugadores y a mantener la frescura y el atractivo de los juegos.

Administración del alojamiento de juegos en producción con Amazon GameLift Servers

Una vez que haya lanzado el juego con Amazon GameLift Servers, deberá administrar la infraestructura de alojamiento del juego para garantizar un rendimiento, una fiabilidad y una experiencia de juego óptimos. En este tema se describen las principales características y herramientas para gestionar los ciclos de vida del alojamiento de juegos durante la producción.

Supervisión del estado y el rendimiento del alojamiento

Amazon GameLift Servers proporciona herramientas de supervisión integrales que le ayudan a monitorizar el estado y el rendimiento de su infraestructura de alojamiento de juegos:

- **Métricas de rendimiento:** supervise los indicadores clave de rendimiento de las flotas administradas, como el uso de las instancias, las sesiones de los jugadores y la ubicación de las sesiones de juego. Realiza un seguimiento de estas métricas en Amazon GameLift Servers o en Amazon CloudWatch, donde también puedes configurar alarmas para recibir notificaciones cuando las métricas superen los umbrales.
- **Métricas y eventos de la flota:** monitorice métricas específicas de la flota, incluidos los procesos activos del servidor, las sesiones de juego disponibles y las sesiones de jugadores activos, para garantizar una capacidad óptima.
- **Eventos y métricas de ubicación de las sesiones de juego:** monitorice métricas y eventos para controlar el estado y el rendimiento del sistema de ubicación de sesiones de juego.
- **Registros de sesiones de juego:** acceda a los registros de las sesiones de juego y analícelos para identificar problemas y deducir patrones de comportamiento de los jugadores.
- **Registros de script de Realtime:** en el caso de los juegos que utilizan servidores en tiempo real, monitorice la ejecución y el rendimiento de los scripts mediante registros detallados.
- **AWS Health Dashboard:** manténgase informado sobre el estado del AWS servicio que podría afectar a sus Amazon GameLift Servers despliegues.

Puede acceder a estas herramientas de monitoreo a través de la Amazon GameLift Servers consola, la AWS CLI o usar el AWS SDK Amazon GameLift Servers para crear paneles y soluciones de monitoreo personalizados.

Administración de actualizaciones y parches del servidor de juegos

Mantener los servidores de juegos actualizados es fundamental para la seguridad y el rendimiento, así como para añadir nuevas características. Amazon GameLift Servers ofrece varios enfoques de administración de las actualizaciones:

- **Administración de compilaciones:** cargue y administre varias versiones de las compilaciones de su servidor de juegos. Cada compilación dispone de control de versiones y se puede implementar en diferentes flotas.

- **Sustitución de flotas:** cree nuevas flotas con actualizaciones de la versión de la AMI y de la compilación del servidor de juegos, y traslade gradualmente el tráfico de las flotas antiguas a flotas nuevas utilizando alias y colas de ubicación de sesiones de juego.
- **Actualizaciones de script:** en el caso de los servidores en tiempo real, actualice los scripts de los servidores sin sustituir toda la flota cargando nuevas versiones de los scripts.
- **Actualizaciones de contenedores:** en el caso de las implementaciones basadas en contenedores, actualice las imágenes y definiciones de contenedores. Implemente nuevas versiones de los servidores de juegos en las flotas existentes. Sustituya las flotas para actualizar las versiones de la AMI.
- **Implementaciones automatizadas:** utilice AWS CodePipeline y cree CI/CD canalizaciones AWS CodeDeploy para las actualizaciones automatizadas de los servidores de juegos.

Cuando planifique las actualizaciones, considere la posibilidad de utilizar estrategias de blue/green despliegue para minimizar las interrupciones que puedan afectar a los jugadores activos y permitir que se reviertan rápidamente si se detectan problemas.

Optimización del rendimiento y escalado

A medida que su base de jugadores evolucione, deberá ajustar la configuración del alojamiento para mantener un rendimiento y una rentabilidad óptimos:

- **Escalado automático:** configure políticas de escalado de la flota basándose en métricas como el número de jugadores o el uso de las sesiones de juego para ajustar automáticamente la capacidad.
- **Ajustes de implementación regional:** añada o elimine regiones de las implementaciones multirregionales basándose en los datos demográficos de los jugadores y los requisitos de latencia.
- **Gestión de colas:** optimice las configuraciones de las colas de ubicación de sesiones de juego para equilibrar la experiencia de los jugadores y los costos de alojamiento.
- **Selección del tipo de instancia:** analice las métricas de rendimiento para determinar los tipos de instancia más rentables para los requisitos de su servidor de juegos.
- **Uso de instancias de spot:** utilice las instancias de spot para cargas de trabajo no críticas a fin de reducir costos, con estrategias alternativas adecuadas que redirijan a instancias bajo demanda.
- **Ajuste de FlexMatch:** perfeccione las reglas y los algoritmos de emparejamiento basándose en los datos y comentarios de jugadores reales.

Revise periódicamente CloudWatch las métricas y los informes de costes para identificar las oportunidades de optimización e implementar los cambios a través de la Amazon GameLift Servers consola o la API.

Solución de problemas y operaciones en tiempo real

Contar con una estrategia eficaz de solución de problemas y administración de las operaciones en tiempo real es fundamental para mantener una experiencia positiva para los jugadores:

- Notificaciones de eventos de la flota: configure notificaciones de Amazon Simple Notification Service para eventos de flota tales como actividades de escalado, finalizaciones de instancia o errores en la ubicación de las sesiones de juego.
- Depuración de la ubicación de sesiones de juego: utilice registros de ubicación detallados para identificar y resolver los problemas relacionados con la ubicación de las sesiones de juego.
- Comprobaciones de estado de los procesos del servidor: supervise el estado de los procesos del servidor y sustituya automáticamente los procesos que no superen las comprobaciones de estado.
- Acceso remoto: conéctese a instancias de la flota para solucionar problemas directamente mediante el uso de AWS Systems Manager Session Manager.
- Administración de alias: utilice alias para redirigir rápidamente el tráfico de jugadores y alejarlo de las flotas problemáticas sin necesidad de cambiar las configuraciones de clientes.
- Copia de seguridad y recuperación: implemente copias de seguridad periódicas de datos y configuraciones críticos del juego para permitir una recuperación rápida en caso de fallo.

Establezca protocolos y manuales de procedimientos claros de respuesta a problemas comunes a fin de minimizar el tiempo de inactividad y el impacto en los jugadores.

Herramientas e integración

Amazon GameLift Servers se integra con varios AWS servicios y herramientas de terceros para mejorar sus capacidades de gestión de la producción:

- AWS CloudFormation- Defina y gestione sus GameLift recursos como infraestructura como código para lograr despliegues coherentes.
- AWS Lambda: cree funciones sin servidor para automatizar las tareas de administración rutinarias y responder a eventos.
- Amazon EventBridge: cree arquitecturas basadas en eventos que respondan automáticamente a los cambios en su entorno. Amazon GameLift Servers

- AWS Integración con el SDK: utilícelo AWS SDKs para crear paneles y herramientas de administración personalizados que se adapten a sus necesidades específicas.
- Supervisión de terceros: intégreala con plataformas de supervisión y análisis de terceros mediante la exportación de CloudWatch métricas.

Aproveche estas integraciones para crear una solución de administración integral que se adapte a los flujos de trabajo y requisitos de su equipo.

Recursos relacionados

- [Supervisión de Amazon GameLift con Amazon CloudWatch](#)
- [Actualización de tus GameLift flotas de Amazon](#)
- [Ampliar la capacidad de alojamiento de juegos con Amazon GameLift](#)
- [Referencia GameLift del SDK de Amazon Server](#)

Términos clave de Amazon GameLift Servers

En este tema se proporcionan las definiciones de los términos técnicos clave que se utilizan en la documentación y los servicios de Amazon GameLift Servers.

Sesión de juego activa

Una sesión de juego que se está ejecutando actualmente y que acepta o aloja jugadores. Se utiliza como métrica para monitorear la utilización de la flota y la planificación de la capacidad. Consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

Alias

Un identificador fácil de usar para una flota que proporciona una capa de abstracción entre el cliente del juego y los recursos de la flota. Los alias te permiten actualizar las configuraciones de la flota sin cambiar el código del cliente del juego. Consulte [Abstracción de una designación de flota de Amazon GameLift Servers con un alias](#).

AMI (imagen de máquina de Amazon)

Imagen de máquina virtual preconfigurada que se utiliza para crear EC2 instancias de Amazon para flotas gestionadas. Contiene el sistema operativo, las bibliotecas de tiempo de ejecución y el software del agente de Amazon GameLift Servers necesarios para ejecutar los servidores de juegos. Consulte [Amazon GameLift Servers EC2 flotas gestionadas](#).

Flota de Anywhere

Tipo de flota que utiliza recursos informáticos proporcionados por el cliente y administrados fuera de Amazon GameLift Servers. Las flotas Anywhere permiten integrar la infraestructura existente o las soluciones de nube híbrida con la administración de las sesiones de juego Amazon GameLift Servers. Consulte [Flotas de Amazon GameLift Servers Anywhere](#).

Authentication token

Una credencial de seguridad generada por Amazon GameLift Servers que autentica los procesos del servidor de juegos que se ejecutan en los recursos de computación de Anywhere. Necesaria para que los procesos del servidor se comuniquen con el servicio Amazon GameLift Servers. Consulte [Flotas de Amazon GameLift Servers Anywhere](#).

Servicio de backend

Un nivel de coordinación entre los clientes del juego y el servicio Amazon GameLift Servers que controla la comunicación para las solicitudes de ubicación de las sesiones de juego y recupera la información de las sesiones de juego. Actúa como un intermediario seguro para proteger los recursos de alojamiento de ataques malintencionados. Consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#).

Compilación del servidor de juegos

Una versión empaquetada del software del servidor de juegos que incluye todos los archivos necesarios para ejecutar el servidor de juegos en los recursos de alojamiento de Amazon GameLift Servers. Contiene el ejecutable, las dependencias y los activos del servidor de juegos. Consulte [Empaquetado de una compilación de servidor de juegos para su implementación](#).

Computación

En las flotas de contenedores, representa una copia de un grupo de contenedores de servidores de juegos en una instancia de flota. Cada ordenador puede alojar una sesión de juego a la vez. Consulte [Cómo funcionan los contenedores en Amazon GameLift Servers](#).

Flota de contenedores

Tipo de flota administrada que ejecuta servidores de juegos en contenedores utilizando imágenes de contenedores en lugar de versiones tradicionales. Proporciona flexibilidad para implementar servidores de juegos con dependencias y tiempos de ejecución específicos. Consulte [Flotas de contenedores administradas de Amazon GameLift Servers](#).

Grupo de contenedores

Conjunto de contenedores que funcionan en conjunto para proporcionar la funcionalidad de un servidor de juegos. Similar a una tarea o pod de contenedores, que define el comportamiento de los contenedores, las dependencias y el uso compartido de recursos dentro de una flota de contenedores. Consulte [Cómo funcionan los contenedores en Amazon GameLift Servers](#).

Destino

En las colas de las sesiones de juego, una flota o un alias que pueden recibir solicitudes de ubicación en las sesiones de juego. Las colas pueden tener varios destinos con diferentes prioridades y políticas de latencia. Consulte [Personalización de colas de sesiones de juego](#).

Fleet

Conjunto de recursos de computación que ejecutan los servidores de juegos. Los recursos de la flota se pueden distribuir en varias ubicaciones geográficas y cada recurso ejecuta uno o más procesos del servidor de juegos. Consulte [Implementación de flotas de alojamiento para Amazon GameLift Servers](#).

Ubicación de la flota

Una AWS región o zona local específica donde se implementan las instancias de flota. Las flotas con múltiples ubicaciones pueden abarcar varias ubicaciones para reducir la latencia y mejorar la disponibilidad. Consulte [Implementación de flotas de alojamiento para Amazon GameLift Servers](#).

FlexMatch

Servicio de emparejamiento de Amazon GameLift Servers que organiza automáticamente las partidas de los jugadores en función de reglas configurables y solicita la ubicación de las sesiones de juego para las partidas completadas. Permite el emparejamiento basado en las habilidades, el equilibrio de los equipos y la reposición de partidas. Consulte [¿Qué es GameLift FlexMatch?](#)

Ciente de juegos

El software de juego que se ejecuta en el dispositivo de un jugador. Inicia la ubicación de jugadores en sesiones de juego comunicándose con un servicio de backend, y se conecta directamente a un servidor de juegos para participar en el juego. Consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#).

Propiedad del juego

Pares de datos personalizados de clave-valor asociados a una sesión de juego. Se utilizan para transmitir información de configuración, reglas del juego u otros metadatos a los servidores y clientes del juego. Consulte [GameProperty](#) en la referencia de la API.

Servidor de juegos

Software de servidor personalizado que administra el estado del juego, procesa las acciones de los jugadores y sincroniza el juego entre varios jugadores conectados. Se integra con el SDK del servidor de Amazon GameLift Servers para comunicarse con el servicio. Consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Activación del servidor de juegos

Proceso de iniciar e inicializar un proceso de servidor de juegos en un recurso de alojamiento. Incluye cargar el ejecutable del servidor de juegos, establecer la comunicación con Amazon GameLift Servers y prepararlo para alojar sesiones de juego. Consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Contenedor de servidor de juegos

Un contenedor dentro de un grupo de contenedores de servidores de juegos que ejecuta la aplicación de servidor de juegos. Se designa automáticamente como esencial para el ciclo de vida del grupo de contenedores. Consulte [Cómo funcionan los contenedores en Amazon GameLift Servers](#).

Grupo de contenedores de servidor de juegos

Tipo de grupo de contenedores que administra los contenedores en los que se ejecutan aplicaciones de servidores de juegos y software de soporte. Necesario para que las flotas de contenedores puedan albergar sesiones de juego y se puede replicar en todas las instancias de la flota. Consulte [Cómo funcionan los contenedores en Amazon GameLift Servers](#).

Proceso del servidor de juegos

Instancia en ejecución del ejecutable del servidor de juegos en un recurso de alojamiento. Cada proceso puede albergar una sesión de juego a la vez y comunicarse directamente con Amazon GameLift Servers para informar sobre el estado y recibir instrucciones. Consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Sesión de juego

Instancia de una partida multijugador que se ejecuta en un proceso de servidor de juegos. Representa una experiencia de juego única en la que varios jugadores pueden unirse y participar

juntos. Consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#) y [GameSession](#) en la referencia de la API.

Datos de la sesión de juego

Información asociada a una sesión de juego, incluidos los datos del jugador, el estado del juego, las propiedades de la sesión y otra información de tiempo de ejecución utilizada por los servidores y clientes del juego. Consulte [GameSession](#) en la referencia de la API.

Ubicación de sesiones de juego

El proceso de localizar los servidores de juegos disponibles para alojar nuevas sesiones de juego. Utiliza información en tiempo real sobre la disponibilidad de los servidores de juegos para tomar decisiones de ubicación óptimas en función de factores como el costo y la latencia. Consulte [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#) y [StartGameSessionPlacement](#) en la referencia de la API.

Protección de la sesión de juego

Configuración que impide a Amazon GameLift Servers finalizar los procesos del servidor de juegos que alojan sesiones de juego activas durante los eventos de escalado o las actualizaciones de la flota. Garantiza que las partidas en curso no se interrumpan. Consulte [Configurar manualmente la capacidad de una Amazon GameLift Servers flota](#).

Cola de sesiones de juego

Un mecanismo de colocación que utiliza algoritmos configurables para organizar las sesiones de juego y obtener resultados óptimos. Puede priorizar las ubicaciones en función de factores como el menor costo de alojamiento y la menor latencia de jugadores en varias ubicaciones geográficas. Consulte [Configuración de la ubicación de las sesiones de juego](#).

Instancia inactiva

Una instancia de flota que se está ejecutando pero que actualmente no aloja ninguna sesión de juego activa. Se utiliza como métrica para supervisar la eficiencia de la flota y las decisiones de escalado. Consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

Permisos de entrada

Reglas de acceso a la red que definen qué direcciones IP y rangos de puertos se pueden usar para conectarse a las instancias de la flota. Controla el acceso a los servidores del juego y garantiza la seguridad de las conexiones. Tanto las flotas de contenedores gestionadas como las gestionadas requieren una configuración de permisos de entrada. EC2

Consulte [IpPermission](#) y [Configuración de conexiones de red](#).

Política de latencia

Reglas en las colas de las sesiones de juego que definen los umbrales de latencia aceptables para las conexiones de los jugadores. Se utilizan para garantizar que los jugadores accedan a las sesiones de juego con un rendimiento de red aceptable. Consulte [Personalización de colas de sesiones de juego](#).

Flota administrada

Tipo de flota en la que Amazon GameLift Servers implementa y administra los recursos informáticos en la Nube de AWS. Proporciona escalado automático, supervisión del estado y administración de la infraestructura. Consulte [Amazon GameLift Servers EC2 flotas gestionadas](#).

Emparejamiento

El proceso de agrupar a los jugadores para jugar en modo multijugador en función de criterios como el nivel de habilidad, la latencia u otros factores específicos del juego. En Amazon GameLift Servers, gestionado principalmente por FlexMatch. Consulte [¿Qué es GameLift FlexMatch?](#)

Grupo de métricas

Un conjunto de métricas de rendimiento relacionadas que se utilizan para la supervisión y las decisiones de escalado automático. Agrupa métricas como las sesiones de juego activas, los servidores de juego disponibles y el uso de los jugadores. Consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).

Grupo de contenedores por instancia

Un tipo de grupo de contenedores opcional que ejecuta software adicional en cada instancia de flota. Útil para servicios en segundo plano o herramientas de monitoreo, con solo una copia implementada por instancia. Consulte [Cómo funcionan los contenedores en Amazon GameLift Servers](#).

Sesión de jugador

Ranura de conexión para un jugador en una sesión de juego. Se utiliza para reservar espacios para las sesiones de juego, validar a los jugadores cuando se conectan y hacer un seguimiento del estado de conexión y de las desconexiones de los jugadores. Consulte [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#) y [PlayerSession](#) en la referencia de la API.

Configuración de prioridad

Ajustes que personalizan el orden de prioridad de la cola de una sesión de juego al seleccionar los destinos para la ubicación de las sesiones de juego. Define el comportamiento general de

las colas, a diferencia de las listas de anulación de prioridades, que modifican la prioridad de las solicitudes de colocación individuales. Consulte [Priorización de la ubicación de las sesiones de juego](#).

Lista de anulaciones de prioridad

Una lista de ubicaciones de prioridad alternativa que se pueden incluir en una solicitud de ubicación de sesión de juego. Sustituye de forma efectiva la priorización configurada en la cola para las ubicaciones solo para esa solicitud, sin afectar a las demás solicitudes. Consulte [Priorización de la ubicación de las sesiones de juego](#).

Administrador de procesos

Componente que administra el ciclo de vida de varios procesos del servidor de juegos dentro de un contenedor o instancia. Maneja el inicio, la detención y la supervisión de los procesos del servidor de juegos individuales. Consulte [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#).

Guión en tiempo real

JavaScript código que define la lógica del juego para Amazon GameLift Servers Realtime Servers. Gestiona las conexiones de los jugadores, el enrutamiento de mensajes y la administración del estado del juego sin necesidad de un servidor de juego personalizado. Consulte [¿Qué es Amazon GameLift Realtime Servers?](#)

Estrategia de enrutamiento

Configuración que determina la forma en que los alias dirigen el tráfico a las flotas. Puede enrutar a una flota específica o distribuir el tráfico entre varias flotas según reglas definidas. Consulte [Abstracción de una designación de flota de Amazon GameLift Servers con un alias](#).

Configuración del tiempo de ejecución

Configuraciones que definen cómo se ejecutan los procesos del servidor de juegos en las instancias de la flota, incluida la cantidad de procesos simultáneos, los parámetros de lanzamiento y la asignación de recursos. Consulte [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#).

Escalado

Proceso de ajustar la capacidad de la flota añadiendo o eliminando recursos de alojamiento en función de la demanda de los jugadores. Puede ser automático (según las métricas de utilización) o manual (controlado por el administrador). Consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).

Política de escalado

Reglas que definen cuándo y cómo ajustar automáticamente la capacidad de la flota. Se pueden basar en objetivos (mantener un uso específico) o en reglas (escalar en función de los umbrales de las métricas). Consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).

SDK del servidor

Kit de desarrollo de software que proporciona las herramientas y bibliotecas necesarias para integrar el servidor de juegos con Amazon GameLift Servers. Permite la comunicación entre el servidor de juegos y el servicio para la administración de sesiones, los informes de estado y las actualizaciones de estado. Consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Multijugador basado en sesiones

Una arquitectura de juego multijugador en la que el juego se desarrolla en sesiones discretas y de tiempo limitado. Cada sesión tiene un inicio y un final definidos, y los jugadores se van y se van según sea necesario. Consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#).

Instancias de spot

Tipo de EC2 instancia de Amazon que utiliza AWS capacidad sobrante a un coste reducido. Las instancias puntuales se pueden interrumpir cuando es AWS necesario recuperar la capacidad, lo que las hace adecuadas para cargas de trabajo de juegos tolerantes a fallos. Consulte [Creación de una cola para instancias de spot](#).

Certificado TLS

Certificado digital que se utiliza para permitir la comunicación cifrada entre los clientes y los servidores del juego. Necesario para las flotas con TLS para garantizar una transmisión de datos segura. Consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#).

Introducción a Amazon GameLift Servers

Aproveche estos recursos de introducción para obtener más información sobre el servicio de Amazon GameLift Servers y sobre cómo puede empezar a desarrollar una solución de alojamiento personalizada para juegos multijugador basados en sesiones.

Antes de comenzar

- Crea una Cuenta de AWS (o designa una existente) para usarla Amazon GameLift Servers.
- Configura usuarios con permisos Amazon GameLift Servers y AWS servicios relacionados.
- Seleccione una Región de AWS con el que trabajar. Para el desarrollo, elija una región próxima a su ubicación. Puede cambiar de región en cualquier momento.

[Configurar una cuenta AWS de usuario](#)

Selección de la ruta

Deseo explorar Amazon GameLift Servers rápidamente

Ideal para: aprender, crear una prueba de concepto o crear prototipos con rapidez

Opciones de inicio rápido:

- [Encapsulador de servidor de juegos de Amazon GameLift Servers](#): consulte [Tutorial: Incorporación rápida con el encapsulador de Amazon GameLift Servers](#). Esta herramienta es la forma más rápida y sencilla de alojar su servidor de juegos y ejecutar sesiones de juego en Amazon GameLift Servers sin necesidad de cambiar el código del juego. El encapsulador de servidor de juegos ofrece una funcionalidad básica de administración de sesiones de juego y una implementación simplificada del servidor de juegos. Es ideal para hacer una evaluación práctica de Amazon GameLift Servers utilizando su propio proyecto de juego o un proyecto de muestra. Cuando lo tenga todo listo para crear una solución de alojamiento de juegos personalizada, puede realizar el cambio a una de las opciones de desarrollo personalizadas que ofrecen integración completa con el SDK del servidor de Amazon GameLift Servers. Si el juego no necesita una solución de alojamiento personalizada, puede seguir utilizando el encapsulador de servidor de juegos para implementar y alojar los servidores de juegos en producción.

- [Complemento de Amazon GameLift Servers para Unreal Engine o Unity](#): los complementos le proporcionan flujos de trabajo de la GUI y ejemplos de recursos para guiarle en los pasos iniciales y en la implementación de su servidor de juegos con una solución de alojamiento básica. Usa el complemento para configurar el alojamiento con flotas autogestionadas de Anywhere o despliega flotas o EC2 flotas de contenedores gestionadas y basadas en la nube. Cuando lo tenga todo listo para desarrollar una solución de alojamiento personalizada, puede compilar sus soluciones basadas en complementos.
- [Kit de inicio para contenedores administrados de Amazon GameLift Servers](#): este kit simplifica las tareas de integración del servidor de juegos, preparación de una imagen de contenedor del servidor de juegos e implementación de una flota de contenedores para el alojamiento. Para la integración, el kit añade características esenciales de administración de sesiones de juego al servidor de juegos. El kit utiliza plantillas preconfiguradas para crear una flota de contenedores y una canalización de implementación automatizada para el servidor de juegos. Cuando lo tenga todo listo para añadir características completas de administración de sesiones de juego, siga una de las hojas de ruta de desarrollo personalizadas para integrar el SDK del servidor de Amazon GameLift Servers.

Deseo crear una solución de alojamiento personalizada

Ideal para: crear una solución de nivel de producción para un servidor de juegos personalizado

Revise la página [opciones de alojamiento de juegos de Amazon GameLift Servers](#) y elija una de las opciones para la solución. Siga una de estas hojas de ruta de desarrollo para empezar a crear una solución de alojamiento personalizada con todas las características para su juego. Las hojas de ruta proporcionan directrices detalladas sobre cómo crear, probar y personalizar cada componente de la solución de alojamiento.

- [Hoja de ruta de desarrollo para el alojamiento con EC2 administrado por Amazon GameLift Servers](#)
- [Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers](#)
- [Hoja de ruta de desarrollo para el alojamiento con Amazon GameLift Servers Anywhere](#)
- [Hoja de ruta de desarrollo para el alojamiento híbrido con Amazon GameLift Servers](#)

Ruta de aprendizaje recomendada

1. Empezar poco a poco. Pruebe con el encapsulador de servidor de juegos o el complemento para Unreal.
2. Comprenda los conceptos. Revise la página [Cómo funciona el alojamiento con Amazon GameLift Servers](#).
3. Elija la arquitectura. Seleccione el modelo de alojamiento en función de los requisitos del juego.
4. Realice la compilación y pruébela. Siga la hoja de ruta de desarrollo que corresponda. Cree una versión básica de cada componente y, a continuación, itérela y personalícela.
5. Escale y optimice. Optimice la solución para el uso en producción. Añada características avanzadas como el emparejamiento.

Configurar una cuenta AWS de usuario

Tip

Utilice estos temas para obtener ayuda con estas tareas:

- Obtenga una nueva Cuenta de AWS para usarla con Amazon GameLift Servers.
- Cree un usuario o grupo con permisos para trabajar con los recursos de Amazon GameLift Servers.
- Configura las credenciales de seguridad (las necesitas para usar las AWS CLI herramientas y los Amazon GameLift Servers complementos de Unreal y Unity)
-

Al igual que con todos los AWS servicios, necesitas Cuenta de AWS usar el Amazon GameLift Servers servicio y las herramientas. An Cuenta de AWS cumple dos funciones principales: (1) le proporciona un contenedor para todos los AWS recursos que cree con la cuenta; y (2) le permite administrar la seguridad de sus AWS recursos, incluida la configuración de la autenticación de los usuarios y el control de los permisos de acceso de los usuarios. La creación de un Cuenta de AWS.

Explore Amazon GameLift Servers con o sin una AWS cuenta

No necesitas una AWS cuenta para:

- Descubre AWS herramientas para crear, ejecutar y desarrollar experiencias de juego en [AWS for Games](#). Leer el [blog](#) y explorar la [biblioteca de soluciones para videojuegos](#).
- Obtén más información Amazon GameLift Servers en la [descripción general del producto y en los recursos](#). FAQs Pedir a AWS que busque respuestas a sus preguntas relacionadas con el producto (pruebe con esta: “Estoy buscando opciones de bajo costo para alojar mi juego multijugador”).
- Para profundizar más, obtenga información sobre el funcionamiento de Amazon GameLift Servers en la [documentación técnica](#), incluidas las guías para desarrolladores sobre alojamiento y emparejamientos, y la guía de referencia de API de servicio.
- Consulte información sobre los [precios de Amazon GameLift Servers](#) y técnicas de optimización de costos. Pruebe la [Calculadora de precios](#) para averiguar cómo se calculan los costos de alojamiento en función del uso de jugadores simultáneos (CCU).
- Obtenga descargas y consulte los repositorios de código Amazon GameLift Servers SDKs, los complementos y los kits de herramientas. Consulte [Introducción a Amazon GameLift Servers](#). (Necesitas un hombre Cuenta de AWS para usarlos).

Necesitas una AWS cuenta para:

- Seguir los flujos de trabajo de incorporación con los complementos de Amazon GameLift Servers para Unreal y Unity, o usar el encapsulador de servidores de juegos.
- Crear y administrar recursos de AWS mediante la Consola de administración de AWS.
- Crear y administrar recursos de AWS mediante la AWS Command Line Interface.
- Usar Amazon Q con la documentación técnica de Amazon GameLift Servers para encontrar respuestas, directrices y recomendaciones.

Temas

- [Inscríbase en una Cuenta de AWS](#)
- [Creación de un usuario con acceso administrativo](#)
- [Configuración de permisos de usuario para Amazon GameLift Servers](#)
- [Para configurar el acceso mediante programación para usuarios, realice el siguiente procedimiento:](#)
- [Configuración de acceso mediante programación para su juego](#)
- [Ejemplo de permisos de IAM para Amazon GameLift Servers](#)

- [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#)

Inscríbase en una Cuenta de AWS

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abrir <https://portal.aws.amazon.com/billing/registro>.
2. Siga las instrucciones que se le indiquen.

Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

Cuando te registras en una Cuenta de AWS, Usuario raíz de la cuenta de AWS se crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar [Tareas que requieren acceso de usuario raíz](#).

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puede ver la actividad de su cuenta actual y administrarla accediendo a <https://aws.amazon.com/> y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión [Consola de administración de AWS](#) como propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Iniciar sesión como usuario raíz](#) en la Guía del usuario de AWS Sign-In .

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

Consulte las instrucciones en [Activar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center .

2. En IAM Identity Center, conceda acceso administrativo a un usuario.

Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte [Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la](#) Guía del AWS IAM Identity Center usuario.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte [Iniciar sesión en el portal de AWS acceso](#) en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

Para conocer las instrucciones, consulte [Create a permission set](#) en la Guía del usuario de AWS IAM Identity Center .

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

Para conocer las instrucciones, consulte [Add groups](#) en la Guía del usuario de AWS IAM Identity Center .

Configuración de permisos de usuario para Amazon GameLift Servers

Cree usuarios adicionales o amplíe los permisos de acceso a los usuarios existentes según sea necesario para sus recursos de Amazon GameLift Servers. Como práctica recomendada ([Prácticas de seguridad en IAM](#)), aplique permisos con privilegios mínimos a todos los usuarios. Para obtener información sobre la sintaxis de los permisos, consulte [Ejemplo de permisos de IAM para Amazon GameLift Servers](#).

Siga las siguientes instrucciones para configurar los permisos de usuario en función de la forma en que administre los usuarios de su AWS cuenta.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

Cree un conjunto de permisos. Siga las instrucciones de [Creación de un conjunto de permisos](#) en la Guía del usuario de AWS IAM Identity Center .

- Usuarios gestionados en IAM a través de un proveedor de identidades:

Cree un rol para la federación de identidades. Siga las instrucciones descritas en [Creación de un rol para un proveedor de identidad de terceros \(federación\)](#) en la Guía del usuario de IAM.

- Usuarios de IAM:
 - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en [Creación de un rol para un usuario de IAM](#) en la Guía del usuario de IAM.
 - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en [Adición de permisos a un usuario \(consola\)](#) de la Guía del usuario de IAM.

Cuando trabaje con usuarios de IAM, una práctica recomendada es adjuntar siempre los permisos a roles o grupos de usuarios, no a usuarios individuales.

Para configurar el acceso mediante programación para usuarios, realice el siguiente procedimiento:

Los usuarios necesitan acceso programático si quieren interactuar con personas AWS ajenas a. Consola de administración de AWS La forma de conceder el acceso programático depende del tipo de usuario que acceda. AWS

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(Recomendado) Utilice las credenciales de la consola como credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Iniciar sesión para el desarrollo AWS local en la Guía del AWS Command Line Interface usuario. • Para ello AWS SDKs, consulte Iniciar sesión para el desarrollo AWS local en la Guía de referencia de AWS SDKs and Tools.
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI, AWS SDKs, o AWS APIs.	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Configuración del AWS CLI uso AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario. • Para AWS SDKs ver las herramientas y AWS APIs, consulte la autenticación del Centro de Identidad de IAM en la Guía de referencia de herramientas AWS SDKs y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes	Siga las instrucciones de Uso de credenciales temporales

¿Qué usuario necesita acceso programático?	Para	Mediante
	programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	con AWS recursos de la Guía del usuario de IAM.
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas dirigidas al AWS CLI, AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Autenticación con credenciales de usuario de IAM en la Guía del AWS Command Line Interface usuario. • Para obtener AWS SDKs información sobre las herramientas, consulte Autenticarse con credenciales de larga duración en la Guía de referencia de herramientas AWS SDKs y herramientas. • Para ello AWS APIs, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

Si usa claves de acceso, consulte [Prácticas recomendadas para administrar las claves de AWS acceso](#).

Configuración de acceso mediante programación para su juego

La mayoría de los juegos utilizan servicios de backend para comunicarse Amazon GameLift Servers mediante el AWS SDKs. Utilice un servicio de backend (que actúe para un cliente de juegos) a fin de solicitar sesiones de juego, colocar a los jugadores en los juegos y otras tareas. Estos servicios

necesitan acceso mediante programación y credenciales de seguridad para autenticar las llamadas a las API del servicio de Amazon GameLift Servers.

Para ello Amazon GameLift Servers, este acceso se gestiona mediante la creación de un usuario jugador en AWS Identity and Access Management (IAM). Administre los permisos de usuario del jugador mediante una de las siguientes opciones:

- Cree un rol de IAM con permisos de usuario del jugador y permita que el usuario jugador asuma el rol cuando sea necesario. El servicio de backend debe incluir código para asumir ese rol antes de realizar solicitudes a Amazon GameLift Servers. De acuerdo con las prácticas recomendadas de seguridad, los roles proporcionan un acceso limitado y temporal. Puede usar roles para cargas de trabajo que se ejecuten en AWS recursos (roles de [IAM](#)) o fuera de ellos AWS (roles de [IAM](#) en cualquier lugar).
- Cree un grupo de usuarios de IAM con permisos de usuario jugador y añada su usuario jugador al grupo. Esta opción proporciona a su jugador credenciales de usuario a largo plazo, que el servicio de backend debe almacenar y utilizar al comunicarse con Amazon GameLift Servers.

Para ver la sintaxis de la política de permisos, consulte [Ejemplos de permisos de usuario de un jugador](#).

Para obtener más información sobre la administración de los permisos para su uso en una carga de trabajo, consulte [Identidades de IAM: credenciales temporales en IAM](#).

Ejemplo de permisos de IAM para Amazon GameLift Servers

Utilice la sintaxis de estos ejemplos para establecer los permisos AWS Identity and Access Management (de IAM) para los usuarios que necesitan acceder a Amazon GameLift Servers los recursos. Para obtener más información sobre la administración de permisos del usuario, consulte [Configuración de permisos de usuario para Amazon GameLift Servers](#). Cuando administre los permisos de los usuarios fuera del Centro de identidades de IAM, se recomienda adjuntar siempre los permisos a los roles o grupos de usuarios de IAM, no a los usuarios individuales.

Si los utiliza Amazon GameLift Servers FleetIQ como solución independiente, consulte [Configurar su Cuenta de AWS para](#). Amazon GameLift Servers FleetIQ

Ejemplos de permisos de administración

Estos ejemplos proporcionan al administrador del alojamiento o desarrollador acceso específico para administrar los recursos de alojamiento de juegos de Amazon GameLift Servers.

Example Sintaxis de los permisos de acceso completo a recursos de Amazon GameLift Servers

En el ejemplo siguiente se concede acceso completo a todos los recursos de Amazon GameLift Servers.

JSON

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

Example Sintaxis de los permisos de recursos de Amazon GameLift Servers con soporte para regiones que no están habilitadas de forma predeterminada

El siguiente ejemplo amplía el acceso a todos los Amazon GameLift Servers recursos y AWS regiones que no están habilitados de forma predeterminada. Para obtener más información sobre las regiones que no están habilitadas de forma predeterminada y cómo habilitarlas, consulte [Administración de Regiones de AWS](#) en la Referencia general de AWS.

JSON

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

Example Syntaxis para el acceso de un recurso de Amazon GameLift Servers a imágenes de contenedor en Amazon ECR

El siguiente ejemplo concede acceso a las acciones de Amazon Elastic Container Registry (Amazon ECR) que los usuarios de Amazon GameLift Servers necesitan cuando trabajan con flotas de contenedores administradas.

JSON

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

Example Syntaxis de los permisos **PassRole** y recursos de Amazon GameLift Servers

En el siguiente ejemplo se concede acceso a todos los recursos de Amazon GameLift Servers y se permite a un usuario pasar un rol de servicio de IAM a Amazon GameLift Servers. Un rol de servicio otorga a Amazon GameLift Servers una capacidad limitada para acceder a otros recursos y servicios en su nombre, tal y como se describe en [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#). Por ejemplo, al responder a una solicitud de CreateBuild, Amazon GameLift Servers necesita acceder a sus archivos de compilación en un bucket de Amazon S3. Para obtener más información sobre la PassRole acción, consulte [IAM: transferir una función de IAM a un AWS servicio específico](#) en la Guía del usuario de 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"
      }
    }
  }
]
```

Ejemplos de permisos de usuario de un jugador

Estos ejemplos permiten a un servicio de backend u otra entidad realizar llamadas a la API de Amazon GameLift Servers. Incluyen los escenarios más comunes para administrar las sesiones de juego, las sesiones de los jugadores y el emparejamiento. Para obtener más información, consulte [Configuración de acceso mediante programación para su juego](#).

Example Syntax de los permisos de colocación de las sesiones de juego

El siguiente ejemplo amplía el acceso a los Amazon GameLift Servers APIs que utilizan las colas de ubicación de las sesiones de juego para crear sesiones de juego y gestionar las sesiones de los jugadores.

JSON

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

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

Example Sintaxis de los permisos de emparejamiento

El siguiente ejemplo amplía el acceso a las áreas Amazon GameLift Servers APIs que gestionan las actividades de FlexMatch emparejamiento. FlexMatch empareja a los jugadores para sesiones de juego nuevas o existentes e inicia la ubicación de las sesiones de juego para los juegos alojados en Amazon GameLift Servers. Para obtener más información acerca de FlexMatch, consulte [¿Qué es 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": "*"
  }
}
```

Example Sintaxis de los permisos de ubicación manual de las sesiones de juego

El siguiente ejemplo amplía el acceso a las Amazon GameLift Servers APIs que crean manualmente sesiones de juego y sesiones de jugadores en flotas específicas. Este escenario es compatible

con los juegos que no utilizan colas de posicionamiento, como los juegos en los que los jugadores pueden unirse eligiendo entre una lista de sesiones de juego disponibles (el método «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": "*"
    }
  ]
}
```

Configuración de un rol de servicio de IAM para Amazon GameLift Servers

Algunas Amazon GameLift Servers funciones requieren que extienda el acceso limitado a otros AWS recursos de su propiedad. Para ello, puede crear un rol AWS Identity and Access Management (IAM). Un [rol de IAM](#) es una identidad de IAM que puede crear en su cuenta y que tiene permisos específicos. Un rol de IAM es similar al de un usuario de IAM en el sentido de que es una AWS identidad con políticas de permisos que determinan lo que la identidad puede y no puede hacer en él. AWS No obstante, en lugar de asociarse exclusivamente a una persona, la intención es que cualquier usuario pueda asumir un rol que necesite. Además, un rol no tiene asociadas credenciales a largo plazo estándar, como una contraseña o claves de acceso. En cambio, cuando asume un rol, le proporciona credenciales de seguridad temporales para su sesión de rol.

En este tema se explica cómo crear un rol que pueda utilizar con las flotas administradas de Amazon GameLift Servers. Si lo utilizas Amazon GameLift Servers FleetIQ para optimizar el alojamiento de juegos en tus instancias de Amazon Elastic Compute Cloud (Amazon EC2), [consulta Configurar](#) tu para. Cuenta de AWS Amazon GameLift Servers FleetIQ

En el siguiente procedimiento, puede crear un rol con una política de permisos y una política de confianza personalizadas que permita a Amazon GameLift Servers asumir el rol.

Creación de un rol de servicio de IAM para una flota de EC2 administrada por Amazon GameLift Servers

Paso 1: Creación de una política de permisos.

Utilice las instrucciones y los ejemplos de esta página para crear una política de permisos personalizada para el tipo de flota de Amazon GameLift Servers con la que trabaja.

Utilización del editor de política de JSON para la creación de una política

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en. <https://console.aws.amazon.com/iam/>
2. En el panel de navegación de la izquierda, elija Políticas.

Si es la primera vez que elige Políticas, aparecerá la página Welcome to Managed Policies (Bienvenido a políticas administradas). Elija Comenzar.

3. En la parte superior de la página, seleccione Crear política.
4. En la sección Editor de políticas, seleccione la opción JSON.
5. Introduzca o pegue un documento de política de JSON. Para obtener más información sobre el lenguaje de la política de IAM, consulte Referencia de [políticas JSON de IAM](#).
6. Resuelva las advertencias de seguridad, errores o advertencias generales que se generen durante la [validación de la política](#) y luego elija Siguiente.

Note

Puede alternar entre las opciones Visual y JSON del editor en todo momento. No obstante, si realiza cambios o selecciona Siguiente en la opción Visual del editor, es posible que IAM reestructure la política, con el fin de optimizarla para el editor visual. Para obtener más información, consulte [Reestructuración de política](#) en la Guía del usuario de IAM.

7. (Opcional) Al crear o editar una política en Consola de administración de AWS, puedes generar una plantilla de política JSON o YAML que puedes usar en CloudFormation las plantillas.

Para ello, en el editor de políticas, selecciona Acciones y, a continuación, selecciona Generar CloudFormation plantilla. Para obtener más información CloudFormation, consulte la [referencia sobre AWS Identity and Access Management los tipos de recursos](#) en la Guía del AWS CloudFormation usuario.

8. Cuando haya terminado de agregar permisos a la política, seleccione Siguiente.
9. En la página Revisar y crear, introduzca el Nombre de la política y la Descripción (opcional) para la política que está creando. Revise los Permisos definidos en esta política para ver los permisos que concede la política.
10. (Opcional) Agregar metadatos a la política al adjuntar las etiquetas como pares de clave-valor. Para obtener más información sobre el uso de etiquetas en IAM, consulte [Etiquetas para AWS Identity and Access Management recursos](#) en la Guía del usuario de IAM.
11. Elija Crear política para guardar la nueva política.

Paso 2: Cree un rol que Amazon GameLift Servers pueda asumir.

Cómo crear un rol de IAM

1. En el panel de navegación de la consola de IAM, seleccione Roles y, a continuación, elija Crear rol.
2. En la página Seleccionar entidad de confianza, elija la opción Política de confianza personalizada. Con esta selección se abre el editor de Política de confianza personalizada.
3. Sustituya la sintaxis JSON predeterminada por la siguiente y, a continuación, elija Siguiente para continuar.

JSON

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

```
]
}
```

4. En la página Añadir permisos, busque y seleccione la política de permisos creada en el paso 1. Elija Siguiente para continuar.
5. En la página Nombrar, revisar y crear, introduzca el Nombre de rol y una Descripción (opcional) para el rol que está creando. Revise las Entidades de confianza y los Permisos agregados.
6. Seleccione Crear rol para guardar el nuevo rol.

Creación de un rol de IAM para contenedores administrados por Amazon GameLift Servers

Si utiliza contenedores administrados por Amazon GameLift Servers, debe crear un rol de servicio de IAM para usarlo con una flota de contenedores. Este rol otorga los permisos limitados que Amazon GameLift Servers necesita para administrar los recursos de su flota de contenedores y tomar medidas en su nombre.

Creación de un rol de IAM para una flota de contenedores

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en. <https://console.aws.amazon.com/iam/>
2. En el panel de navegación de la consola de IAM, seleccione Roles y, a continuación, elija Crear rol.
3. En la página Seleccione una entidad de confianza, elija el AWS servicio y seleccione el caso de uso "GameLift». Elija Siguiente.
4. En Añadir permisos, elija la política administrada GameLiftContainerFleetPolicy. Elija Siguiente. Para obtener más información sobre esta política, consulte [AWS políticas gestionadas para Amazon GameLift Servers](#).
5. En Asignar nombre, revisar y crear, introduzca un nombre de rol y elija Crear rol para guardar el nuevo rol.

Sintaxis de la política de permisos

- Permisos para que Amazon GameLift Servers asuma el rol de servicio

JSON

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

- Permisos para acceder a AWS las regiones que no están habilitadas de forma predeterminada

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"
    }
  ]
}
```

Obtención de herramientas de desarrollo de Amazon GameLift Servers

Amazon GameLift Servers proporciona un conjunto SDKs y otras herramientas que le ayudarán a crear soluciones de alojamiento de juegos para sus juegos. SDKs Añaden funciones a los servidores de juegos, los clientes de juegos y los servicios de backend que les permiten interactuar con el Amazon GameLift Servers servicio. Para obtener la información más reciente sobre las versiones de los SDK de Amazon GameLift Servers y su compatibilidad, consulte [Notas de la versión de Amazon GameLift Servers](#).

Para servidores de juegos

Integre y compile servidores de juegos de 64 bits con el SDK del servidor de Amazon GameLift Servers. El servidor de juegos utiliza el SDK del servidor para comunicarse con el servicio Amazon GameLift Servers y administrar las sesiones de juego, lo que incluye iniciar, actualizar y detener las sesiones de juego. Si necesita ayuda para integrar el SDK del servidor en sus proyectos de juego, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#).

Admisión de entornos de desarrollo

- SO de desarrollo
 - Windows
 - Linux
- Lenguajes de programación

[Obtención del SDK de Amazon GameLift Servers](#). Para obtener más información e instrucciones de instalación para cada versión, consulte los archivos readme incluidos en cada paquete.

- [SDK del servidor de C++](#)
 - [Referencia del SDK del servidor](#)
 - [Cómo realizar la integración](#)
- [SDK del servidor de C#](#) (la compatibilidad para .NET 4, .NET 6 y .NET 8 varía según la versión; consulte [Versiones del SDK](#))
 - [Referencia del SDK de servidor](#)
 - [Cómo realizar la integración](#)
- [SDK del servidor de Go](#)

- [Referencia del SDK de servidor](#)
- [Cómo realizar la integración](#)
- Compatibilidad para motores de juegos

El complemento con todas las funciones Amazon GameLift Servers incluye flujos de trabajo de interfaz de usuario y activos de muestra, así como versiones integradas del AWS SDK y el SDK del servidor. Los flujos de trabajo le guían sobre cómo configurar e implementar su servidor de juegos para alojarlo con flotas de EC2 administradas, flotas de contenedores administradas o flotas de Anywhere autoadministradas. Si no necesitas los flujos de trabajo guiados, puedes obtener únicamente el SDK de servidor para tu motor de juego desde los mismos GitHub repositorios.

Si utiliza otro motor de juegos o entorno de desarrollo no compatibles con el complemento, obtenga el SDK del servidor correspondiente a su lenguaje de programación y añádalo a su proyecto de juego.

Para obtener más información e instrucciones de instalación para cada versión, consulte los archivos readme incluidos en cada paquete.

- [Complemento para Unreal Engine](#): integrado para su uso con Unreal Engine versiones 5.0, 5.1, 5.2, 5.3, 5.4, 5.5 y 5.6. Consulte los archivos readme específicos de cada versión para comprobar la compatibilidad con Unreal.
 - [Guía del complemento para Unreal Engine](#)
 - [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
- [Complemento para Unity](#): integrado para su uso con versiones LTS del editor de Unity 6.0, 2022.3 o 2021.3. Es compatible con los perfiles .NET Framework y .NET Standard de Unity, con .NET Standard 2.1 y .NET 4.x. Consulte los archivos readme específicos de cada versión para comprobar la compatibilidad con Unity.
 - [Guía del complemento para Unity](#)
 - [Referencia del SDK del servidor de C#](#)
- [SDK del servidor para Unreal](#)
 - [Referencia del SDK para servidores](#)
 - [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#)
- [SDK del servidor para Unity](#)
 - [Referencia del SDK de servidor](#)

- [Integración de Amazon GameLift Servers en un proyecto de Unity](#)

Compatibilidad del tiempo de ejecución

Para una solución de alojamiento gestionado, crea tu servidor de juegos para que se ejecute en una de las siguientes imágenes de máquinas de Amazon (AMIs). Consulte [Versiones de AMI de Amazon GameLift Servers](#) de Amazon GameLift Servers para obtener información más detallada.

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

Note

Amazon Linux 2 (AL2) finalizará el soporte el 30 de junio de 2026. Consulte más detalles en [Amazon Linux 2 FAQs](#). En el caso de los servidores de juegos que están alojados en el SDK de Amazon GameLift Servers servidor 4.x AL2 y lo utilizan, primero actualice la compilación del servidor de juegos a la versión 5.x del servidor y, a continuación, impleméntela en las AL2023 instancias. Consulte [Migración al SDK 5.x del servidor de Amazon GameLift Servers](#).

Herramientas adicionales

[Encapsulador de servidor de juegos para Amazon GameLift Servers](#)

Esta herramienta le ayuda a implementar un servidor de juegos para el alojamiento con un conjunto de funciones básicas de administración de las sesiones de juego. Con esta herramienta, no es necesario realizar ningún cambio en el código del juego ni integrar el SDK del servidor de Amazon GameLift Servers. Utilice el encapsulador de servidor de juegos para empaquetar el servidor de juegos e implementarlo para el alojamiento de juegos utilizando cualquiera de las tres soluciones de alojamiento de Amazon GameLift Servers (Anywhere, EC2 administrado o contenedores administrados). Esta herramienta es más útil para realizar una evaluación temprana o crear prototipos con su propio juego o con un juego de muestra, ya que no admite la personalización del servidor de juegos. Si el juego no necesita características personalizadas, puede implementar el servidor de juegos con el encapsulador de servidores de juegos para el alojamiento en producción.

[Amazon GameLift ServersKit de herramientas](#)

El kit de herramientas de Amazon GameLift Servers es una colección de scripts y otras herramientas que hemos desarrollado para ayudar a los desarrolladores a resolver situaciones y problemas habituales. Los materiales del kit de herramientas incluyen scripts, código de muestra y archivos readme.

- [Kit de inicio para contenedores](#): utilice esta herramienta para simplificar las tareas de configuración de las compilaciones de servidores de juegos para su alojamiento con contenedores administrados por Amazon GameLift Servers. El kit integra características esenciales de administración de sesiones de juego en un servidor de juegos, y utiliza plantillas preconfiguradas para crear una flota de contenedores y una canalización de implementación automatizada para el servidor de juegos. Tras la implementación, puede supervisar el rendimiento de la flota, administrar las sesiones de juego y analizar las métricas utilizando la consola y las herramientas de la API de Amazon GameLift Servers. El kit se integra con Amazon Simple Storage Service AWS CodeBuild para el almacenamiento y para la implementación de infraestructuras AWS CloudFormation para la automatización de la construcción.
- [Fast Build Update Tool](#): utilice esta herramienta para modificar la compilación de un servidor de juegos que ya esté implementada en una flota de EC2 administrada. La herramienta está diseñada para ayudarle a intercambiar rápidamente los archivos de compilación del juego sin necesidad de configurar o crear nuevas flotas de EC2 con cada cambio. Puede actualizar instancias individuales o todas las instancias de la flota. Las opciones le permiten reemplazar archivos de compilación específicos o una compilación completa, y le permiten administrar cómo reiniciar los servidores de juegos después de las actualizaciones.

Para clientes de juegos

Integre su cliente de juego con el SDK del cliente para Amazon GameLift Servers. El SDK del cliente proporciona utilidades para que los clientes de juegos utilicen Amazon GameLift Servers funciones como la puerta de enlace del jugador y las balizas de ping UDP.

Obtén el SDK del cliente

- [SDK de cliente para C++](#): incluye utilidades de pasarela de reproductor y balizas de ping UDP con algoritmos de selección de puntos finales integrados, además de una aplicación de muestra que demuestra la integración. Requiere C++17 y una biblioteca de subprocesos.
- [SDK de cliente para Unreal Engine](#): incluye una pasarela para jugadores (con o sin modificaciones en el motor) y utilidades de balizas de ping UDP con algoritmos de selección de puntos finales

integrados. Requiere la versión original de Unreal Engine 5.1 o posterior. El acceso requiere ser miembro de la organización de [Epic Games](#) a partir de. GitHub

Para ver las guías de integración, consulta la documentación de cada GitHub repositorio.

Para ver los servicios de back-end de juegos

Crea un servicio de backend de 64 bits para tu juego e intégralo con las funciones del AWS SDK, que incluye la API de servicio para. Amazon GameLift Servers Usa el servicio de backend para gestionar las interacciones con el Amazon GameLift Servers servicio, como iniciar o encontrar sesiones de juego y unir a los jugadores a las partidas.

[Obtención del SDK de AWS](#)

Para obtener más información sobre cómo usar el AWS SDK con Amazon GameLift Servers, consulta los siguientes recursos:

- [Referencia de la API de Amazon GameLift Servers](#)
- Integración de la funcionalidad de cliente de juego
 - [Integración del servicio del lado del cliente para juegos con compilaciones de servidor de juegos personalizadas](#)
 - [Integración del servicio del lado del cliente para juegos con servidores Realtime](#)
- [Diseñe un servicio de back-end](#)

Para la administración de recursos de Amazon GameLift Servers

Utilice las siguientes herramientas para crear, actualizar y supervisar sus recursos de alojamiento administrados por Amazon GameLift Servers.

- [Consola de administración de AWS](#)— La AWS consola es una aplicación basada en la web que proporciona acceso centralizado a todas las consolas de AWS servicio individuales, incluidas Amazon GameLift Servers las siguientes. Usa la consola para crear o iniciar sesión en una AWS cuenta y abre la Amazon GameLift Servers consola para que funcione con tus recursos de alojamiento de juegos. Puede configurar e implementar flotas de alojamiento y otros recursos, ver métricas de uso y de rendimiento, monitorizar los recursos en el panel, además de otras tareas. [Ve a la Amazon GameLift Servers consola.](#)

- [API de servicio de Amazon GameLift Servers](#): esta API le brinda acceso programático a todos sus recursos de Amazon GameLift Servers. Forma parte del AWS SDK, que puedes descargar para usarlo con los lenguajes de programación más populares. [Obtén el AWS SDK](#).
- [AWS interfaz de línea de comandos \(CLI\)](#): la AWS CLI le permite interactuar con AWS los servicios mediante un shell de línea de comandos. Las herramientas proporcionan acceso directo al público APIs para los AWS servicios, así como comandos personalizados que están disponibles para un servicio. [Obtenga la AWS CLI](#).
- [AWS CloudFormation](#) para Amazon GameLift Servers: el CloudFormation servicio le ayuda a modelar y configurar AWS los recursos para optimizar la implementación y la administración de la infraestructura. Cree una CloudFormation plantilla para describir los Amazon GameLift Servers recursos de su solución de alojamiento y, a continuación, utilícela para crear recursos adicionales o actualizar las configuraciones. Consulte la [Referencia de tipos de recurso de Amazon GameLift Servers](#).

Para Amazon GameLift Servers Realtime

Configure e implementa servidores de Realtime para alojar sus juegos multijugador. Para permitir que sus clientes de juego se conecten a servidores de Realtime, utilice el SDK de cliente de Amazon GameLift Servers Realtime. Para comenzar, [descargue el SDK de cliente de Realtime](#). Para obtener información sobre la configuración, consulte [Integración de un cliente de juego para Amazon GameLift Servers Realtime](#).

Compatibilidad con SDK

El SDK del cliente de Realtime contiene código fuente para los siguientes lenguajes:

- C# (.NET)

Entornos de desarrollo

Compile el SDK a partir del código adecuado para los siguientes sistemas operativos de desarrollo y motores de videojuegos compatibles:

- Sistemas operativos: Windows, Linux, Android e iOS
- Motores de juegos: Unity y motores que admiten bibliotecas de C#

Sistemas operativos de servidor de juegos

Puede implementar servidores de Realtime en recursos de alojamiento que se ejecuten en las siguientes plataformas:

- [Amazon Linux 2](#)

 Note

AL2 se acerca al final del soporte. Consulte más detalles en [Amazon Linux 2 FAQs](#).

- [Amazon Linux 2023](#)

Tutorial: Incorporación rápida con el encapsulador de Amazon GameLift Servers

Le damos la bienvenida al tutorial de incorporación de a Amazon GameLift Servers. En este tutorial, implementará rápidamente su servidor de juegos para que se aloje en una flota de recursos de computación basados en la nube. Utilice este tutorial para evitar la tarea de integrar el SDK del servidor de Amazon GameLift Servers en el código del juego y, en su lugar, impleméntelo con la funcionalidad mínima necesaria para comunicarse con el servicio Amazon GameLift Servers y ejecutar sesiones de juego. Configuraré una solución de alojamiento básica y la utilizaré para disfrutar de todo el abanico de características, como el escalado automático y el emparejamiento. También es ideal para alojar un prototipo del juego como parte de una demostración en directo o a modo de prueba.

Ventajas clave de este método de incorporación:

- Implemente rápidamente el servidor de juegos para un alojamiento rápido.
- No hay cambios en el código del juego ni se requieren modificaciones.
- Utilice este método con cualquier ejecutable del juego, independientemente del motor de juegos.
- Explore todas las herramientas de administración de Amazon GameLift Servers, incluida la supervisión de la actividad de las sesiones de juego y el estado del alojamiento.

 Note

El encapsulador está diseñado para la evaluación y el uso básico en producción. Las características avanzadas, como la administración detallada de las sesiones de jugador, requieren una integración completa del SDK del servidor.

Requisitos previos

Antes de comenzar, asegúrese de que dispone de lo siguiente:

- Y Cuenta de AWS con los permisos adecuados
- AWS CLI instalado
- Go 1.18+
- Un archivo ejecutable de servidor de juegos multijugador
- Make (Linux/Mac)
- Git instalado con una cuenta activa

Descripción general de

En este tutorial, podrá:

1. Obtener y crear el encapsulador
2. Preparar la compilación del juego
3. Configurar el encapsulador
4. Cargar la compilación del servidor de juegos
5. Cree la EC2 flota gestionada
6. Crear una sesión de juego y conectarse a ella
7. Supervisar y administrar los servidores de juegos
8. Escalar los servidores de juegos

Paso 1: Obtener y crear el encapsulador de servidor de juegos

Utilice los siguientes comandos para obtener la fuente del encapsulador de servidor de juegos y compilarlo. Estos comandos usan SSH, pero también puede ir directamente al repositorio de 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 y Linux

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

Si la compilación se realiza correctamente, se añade un directorio “out” a `amazon-gamelift-servers-game-server-wrapper`. En este directorio hay tres carpetas, una para cada opción de flota de alojamiento compatible, que contienen un conjunto de artefactos de compilación. Para este tutorial, va a realizar la implementación en EC2 flotas gestionadas, por lo que utilizará la carpeta `amazon-gamelift-servers-managed-ec2`.

Paso 2: Preparar la compilación del servidor de juegos

En este paso, deberá preparar los archivos de compilación de servidor de juegos para cargarlos en Amazon GameLift Servers.

Creación del directorio del juego

Ahora, prepare un directorio para el juego en su máquina local. Este directorio debe contener todos los archivos que necesita para ejecutar el servidor de juegos con Amazon GameLift Servers. Esto incluye el contenedor del servidor de juegos, la versión de tu servidor de juegos y el archivo `config.yaml` que hace que el contenedor funcione con tu servidor de juegos.

Utilice los siguientes pasos:

1. **EC2 Flota gestionada.** En la carpeta de salida del contenedor del servidor del juego, busca el artefacto de construcción para desplegarlo en una flota gestionada EC2 . La compilación

se escribirá en un directorio externo como se indica aquí: `out\linux\amd64\gamelift-servers-managed-ec2`.

2. Copie el ejecutable de su servidor de juegos y todos los archivos asociados que necesite para ejecutarse en la carpeta `gamelift-servers-managed-ec2`. Puede tener directorios anidados según sea necesario.

Un ejemplo de estructura de directorios sería el siguiente:

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

Paso 3: Configurar la contenedor para la flota

Amazon GameLift Servers administra el ciclo de vida de las instancias de computación de una flota, activando nuevas instancias con la versión de servidor instalada y reciclando las instancias según sea necesario. El servicio administra el ciclo de vida de los procesos del servidor de juegos que se ejecuta en cada instancia. Una EC2 flota gestionada puede tener instancias en varios lugares para ayudar a los jugadores estén donde estén.

Edita el archivo `config.yaml` para configurar el contenedor para el registro, la configuración de los puertos y la inicialización del servidor.

1. Configure los ajustes de registro. El contenedor del servidor de juegos genera mensajes de registro para cada proceso del servidor de juegos. De forma predeterminada, el nivel registro está establecido en Depuración para obtener la máxima verbosidad. Esto resulta muy útil durante la configuración y la solución de problemas y determina el nivel de detalle de los mensajes de registro (en este caso, los más detallados). Las opciones incluyen depuración, información, advertencia y error (las menos detalladas).
2. Especifica la ruta al directorio de registro del servidor del juego. La ruta por defecto para los registros del servidor de juegos es `./game-server-logs`. Este directorio contiene todos los registros generados por el servidor de juegos y cada instancia los contiene. Los registros se

cargan automáticamente en Amazon GameLift Servers, donde se accede a ellos desde la pestaña Eventos. Consulte la sección de solución de problemas para obtener más detalles.

3. Defina la configuración de los puertos de red. Configure el puerto del juego según su preferencia. Para este tutorial, especifique un solo puerto, ya que creará una flota que ejecutará solo un proceso de servidor de juegos simultáneo por instancia. Si decide ejecutar varios procesos a la vez, tendrás que configurar suficientes puertos para cada proceso simultáneo. El valor predeterminado es 37016, como se muestra en el archivo de configuración, pero en general, para las flotas que utilizan compilaciones de Linux, utilice los puertos 22 y 1026-60000. Para las flotas que utilizan compilaciones de Windows, utilice los puertos del 1026 al 60 000.
4. Establezca la ruta al archivo ejecutable del servidor de juegos. Para `./MyGame/my-server-executable.exe`, personalice la ruta al ejecutable de su servidor de juegos con su nombre y ubicación reales. Este es el punto de entrada para lanzar el servidor de juegos.
5. Configure los argumentos del servidor del juego. Como mínimo, especifica un argumento `-port` y usa el mismo valor de puerto de juego que definió anteriormente. El valor "pos" 0 indica que este es el primer argumento. Añada otros argumentos según corresponda. Estos argumentos se pasan al servidor de juegos cuando se lanza, lo que le permite configurar su comportamiento en tiempo de ejecución.
 - a. Argumento: `--port`
 - b. Valor: `"{{.port number here}}"`
 - c. Posición: 0 (primer argumento de la lista)

Ejemplo de configuración:

```
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
```

Paso 4: Cargar la compilación del servidor de juegos

Ya ha completado todos los elementos necesarios de la compilación de su servidor de juegos (el encapsulador de servidor de juegos, config.yaml y los archivos de servidor de juegos) y está listo para cargar la compilación del juego en Amazon GameLift Servers para su alojamiento. La forma más rápida de cargar la versión del juego es mediante el AWS CLI comando que `upload-build` se muestra en el siguiente ejemplo.

Carga de una compilación de juego con 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_2022 \  
  --server-sdk-version 5.3.0 \  
  --region us-west-2
```

Note

Para compilaciones de Mac y Linux, use `--operating-system AMAZON_LINUX_2023`

Al crear la compilación, registre el ID de compilación de la respuesta de la API para usarlo en la creación de la flota.

Paso 5: Crea la flota gestionada EC2

Los siguientes pasos describen una configuración mínima de la flota para que pueda empezar a funcionar lo antes posible.

Para crear la flota:

1. Inicie sesión en Consola de administración de AWS y navegue hasta Amazon GameLift Servers.
2. En la barra de menús de la parte superior de la ventana de la consola, compruebe en qué región se encuentra la compilación. Apúntalo porque tu flota debe estar en la misma región o no podrás encontrar ni elegir tu construcción.

3. En el panel de navegación de la EC2 sección Administrado, elija Compilaciones.
4. Seleccione la compilación que cargó anteriormente para mostrar la página de detalles de la compilación.
5. En la sección Flotas, seleccione Crear flota, que mostrará la página de detalles sobre EC2 cómo definir la flota gestionada, desde la que podrá realizar un seguimiento del estado de su flota y ver los eventos de creación de la flota en la pestaña Eventos.
6. Rellene el nombre y la descripción y seleccione Siguiente.
7. En la página Definir detalles de la instancia, se muestra la región de la compilación de forma predeterminada. Elija cualquier región adicional que desee agregar.
8. En Tipo de flota, elija Bajo demanda.
9. En Tipos de instancia, elija c5.large y Siguiente.
10. En Configuración de tiempo de ejecución, dado que la compilación del juego que ha cargado usa el encapsulador, debe apuntar al ejecutable del encapsulador. Para los servidores de juegos de Windows, es `C:\game\amazon-gamelift-servers-game-server-wrapper.exe`. Para los servidores de juegos de Linux, esto es `/local/game/amazon-gamelift-servers-game-server-wrapper`.

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

Además, configura los valores de los puertos del juego con un rango que tenga en cuenta el puerto que se estableció en los `config.yaml` parámetros de inicio de la configuración del tiempo de ejecución. `config.yaml` No es necesario que el puerto de entrada coincida con el especificado en la configuración de tiempo de ejecución de la flota, pero en tiempo de ejecución, si la configuración especifica un puerto diferente, ese valor anula el `config.yaml` que hay en. Los parámetros de lanzamiento introducidos en la configuración del tiempo de ejecución también anulan los que hay en `config.yaml`.

11. En la página Revisar y crear, comprueba todas las configuraciones y, a continuación, selecciona Enviar para crear tu flota. El estado de la flota cambiará a medida que aumente su capacidad para alojar el servidor del juego y, muy pronto, aparecerá como Activa. Una vez completada la activación y desplegada la flota, el servicio lanza el contenedor, que está listo para recibir una solicitud de sesión de juego.

Paso 6: Crea una sesión de juego y conéctate a ella

Si el estado de tu flota es Activa, significa que los servidores del juego están listos para albergar una sesión de juego. Para iniciar una sesión de juego, realiza una solicitud de sesión de juego al Amazon GameLift Servers servicio. Aquí, utilizará la AWS CLI para realizar esta solicitud.

Note

Ten en cuenta que crear una sesión de juego con el AWS CLI es útil para probar y familiarizarte con el proceso. En algún momento, añadirás llamadas programáticas al AWS SDK al servicio de backend del juego como parte del sistema de asignación de partidas o ubicación de las sesiones de juego.

Para crear una sesión de juego, utilice lo siguiente:

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

También puede pasar las propiedades personalizadas del juego al ejecutable del servidor. Consulta los argumentos del servidor de juegos en el archivo README para obtener más información. Cuando reciba la create-game-session llamada, Amazon GameLift Servers avisa al contenedor de que inicie el ejecutable del servidor del juego e inicie una sesión de juego. Lo que está dentro config.yaml afecta a la configuración del servidor del juego, y los parámetros de inicio establecidos en la consola determinan la configuración de la sesión de juego en sí.

Ejemplo de formato para añadir propiedades del juego:

```
defaultArgs:  
  - arg: "--port"  
    val: "{{.GamePort}}"  
    pos: 0  
  - arg: "--ipAddress"  
    val: "{{.IpAddress}}"
```

```
pos: 1
- arg: "--gameSessionId"
  val: "{{.GameSessionId}}"
pos: 2
```

Paso 7: Gestiona y monitorea tu flota

Ahora que tu flota de servidores de juegos está configurada y ha empezado una sesión de juego, puede administrarla y monitorizarla desde la consola de Amazon GameLift Servers. La mejor forma de hacerlo es desde la página de detalles de la flota, donde puede editar los detalles de la flota o cambiar la escala y la capacidad de la flota en la pestaña Escalado. Consulta la siguiente sección sobre cómo escalar los servidores de tus juegos.

Selecciona la pestaña Métricas para ver los gráficos que muestran la supervisión de las métricas de actividad y hardware. Para obtener más información sobre los gráficos de métricas, selecciona el enlace Información situado junto al identificador de tu flota. Además, si bien puedes supervisar de cerca tus servidores de juegos desde la pestaña Métricas, también puedes añadir alarmas a estas métricas en el CloudWatch panel de control.

Para navegar al CloudWatch panel de control desde la consola:

1. Escribe "CloudWatch" en la barra de búsqueda y selecciónalo en la lista de resultados de búsqueda para ver la CloudWatch descripción general.
2. Desplázate hacia abajo y selecciona Ver GameLift panel para ver los gráficos basados en las métricas clave de tus flotas y sesiones de juego.

Paso 8: Escala tus servidores de juegos

Para el siguiente paso, deberá configurar el escalado automático. Utilice el escalado automático para escalar de forma dinámica la capacidad de la flota en respuesta a la actividad del servidor de juegos. A medida que los jugadores lleguen e inicien sesiones de juego, el escalado automático añadirá más instancias, y a medida que la demanda de los jugadores disminuya, el escalado automático liberará instancias no utilizadas. Esta es una forma eficaz de minimizar sus recursos y costos de alojamiento y, a la vez, facilitar al jugador una experiencia rápida y sin interrupciones.

Durante la preparación para el lanzamiento del juego, tendrá que configurar el escalado automático para sus flotas. El escalado automático es una práctica recomendada eficaz para minimizar los

recursos y los costos del alojamiento y, a la vez, facilitar al jugador una experiencia rápida y sin interrupciones.

Para configurar manualmente la capacidad de la flota

Para configurar manualmente la capacidad de la flota

1. Ve a la pestaña Escalamiento en la página de detalles de tu flota.
2. Seleccione una ubicación y elija Editar.
3. Cambie el valor de las instancias deseadas y cambie la configuración de tamaño mínimo y máximo para escalar más allá de sus valores actuales, y seleccione Confirmar.

Note

Utilice la configuración del número máximo de instancias como medida provisional para evitar un escalado y unos costos excesivos.

Para usar el escalado automático basado en objetivos

Para usar el escalado automático basado en objetivos

El ajuste de escala automático basado en objetivos (seguimiento de objetivos) vincula el escalado de la flota con el porcentaje de sesiones de juego disponibles. A medida que los jugadores se lanzan a jugar a tu juego y disminuyen las sesiones de juego disponibles, la aplicación responde añadiendo automáticamente más instancias a la flota.

1. En Política de escalado automático basada en objetivos, selecciona Añadir política y establece la capacidad de la flota para que cambie automáticamente cuando alcance el límite del porcentaje de sesiones de juego disponibles que hayas establecido. Un búfer más grande puede gestionar mejor las oleadas de jugadores, haciendo que los nuevos jugadores se metan rápidamente en las partidas, pero también puede suponer un aumento de los costos de alojamiento.
2. Seleccione Confirmar para aceptar los cambios.

El escalado automático basado en reglas le brinda un control más detallado, como la capacidad de vincular el escalado con otras métricas de la flota y establecer umbrales personalizados y respuestas de escalado. Ofrece opciones potentes, pero también requiere el uso de la CLI y pruebas

importantes para comprender cómo se comportan las reglas personalizadas en acción. Este tutorial se centra primero en configurar el enfoque basado en objetivos.

Solución de problemas comunes

Los siguientes son problemas comunes que pueden surgir al ejecutar los servidores y las sesiones de juego. Si el servidor o la sesión de juego no funcionan correctamente, el primer paso consiste en comprobar los registros, ya que pueden revelar alguno de los problemas que se describen a continuación en relación con nuevas implementaciones o juegos en producción.

En los registros suele aparecer lo siguiente:

- El proceso del servidor del juego no se puede iniciar. Puede que se deba a un error en la configuración del contenedor. Compruebe que el archivo tenga la ruta de inicio correcta y que los parámetros y argumentos de lanzamiento sean correctos.
- La compilación del servidor de juegos no se puede ejecutar. Es probable que se trate de un error en el código del juego.
- Los jugadores no se pueden conectar a las sesiones de juego. Es probable que se trate de un error de configuración del puerto.
- Conexiones retrasadas o lentas. Revise las políticas y los umbrales de escalado.
- Sin conexión. Verifique las reglas y la configuración de los puertos de su flota.

Para ver los registros de eventos de su Amazon GameLift Servers flota

Para ver los registros de eventos de su Amazon GameLift Servers flota

1. Abra la consola de Amazon GameLift Servers.
2. En la página de detalles de la flota, selecciona la pestaña Eventos y descarga el registro. También puede supervisar las métricas de actividad y hardware para comprobar el estado del servidor de juegos y las activaciones de las sesiones de juego desde la pestaña Métricas.

Para ver los registros de las sesiones de juego

Para ver los registros de las sesiones de juego

1. Desde la consola, abre tu flota y abre la pestaña Sesiones de juego.

2. Elija un ID de sesión de juego de la lista para mostrar la página de Información general de la sesión de juego.
3. Selecciona Descargar registros para descargar un archivo de registro de forma local.

Para ver los registros de las sesiones del juego con la CLI, utilice la API de `GetGameSessionLogURL`. Amazon GameLift Servers almacena automáticamente los registros durante 14 días.

También puedes configurar CloudWatch los registros de Amazon para tu flota. Esto proporciona capacidades de registro adicionales e integración con otros servicios AWS de monitoreo.

Para acceder a los registros en tiempo real o prolongar los períodos de retención mediante CloudWatch:

1. En la parte superior del panel de control de la Amazon GameLift Servers consola, escribe CloudWatch "" en la barra de búsqueda y selecciónalo en el menú desplegable de resultados.
2. Ve a los grupos de CloudWatch registro y busca sesiones específicas. El método más sencillo es hacer clic en Buscar todo y filtrar mediante `gameSessionId` o `ClientID`.

Siguientes pasos

- [Cree una flota con múltiples ubicaciones para agregar alojamiento en ubicaciones adicionales](#)
- [Añade una cola de sesiones de juego para conseguir las mejores ubicaciones posibles para las sesiones de juego en varios lugares](#)
- [Experimenta con el FlexMatch emparejamiento creando un emparejador y un conjunto de reglas para tu juego](#)
- [Empieza a trabajar en la funcionalidad de los componentes del cliente y del servicio de backend del juego, de forma que los jugadores puedan solicitar unirse y conectarse directamente a las sesiones de juego](#)
- [Cuando lo tenga todo listo, haga el cambio a una solución totalmente integrada](#)

Ejemplos de Amazon GameLift Servers

Si está pensando en utilizar Amazon GameLift Servers para administrar un servidor de juegos personalizado, o si está interesado en usar Amazon GameLift Servers Realtime, le recomendamos

que pruebe los siguientes ejemplos antes de usar el servicio para su propio juego. El ejemplo del servidor de juegos personalizado le proporciona experiencia con el alojamiento de juegos en la consola de Amazon GameLift Servers. El ejemplo de Amazon GameLift Servers Realtime muestra cómo preparar un juego para su alojamiento con servidores Realtime.

Ejemplo de servidor de juegos personalizado

En este ejemplo, se muestra el proceso de implementación de un servidor de juegos de muestra en una EC2 flota Amazon GameLift Servers gestionada para su alojamiento. Utilice el cliente de juego de muestra para conectarse a una sesión de juego en directo. Descubra cómo utilizar las herramientas de Amazon GameLift Servers, como la consola y la CLI de AWS , para supervisar el rendimiento y el uso del alojamiento de la flota.

En el ejemplo se le guiará por los siguientes pasos:

- Cargar la compilación del servidor de juegos de ejemplo.
- Crear una flota para ejecutar la compilación del servidor de juegos.
- Obtener el cliente de juego de muestra y usarlo para la conexión a un servidor de juegos y para unirse a una sesión de juego.
- Revisar las métricas de la flota y las sesiones de juego.

Inicie varios clientes de juego y empiece a jugar para generar datos de alojamiento. Utilice la consola de Amazon GameLift Servers para ver los recursos de alojamiento, hacer un seguimiento de las métricas y explorar las opciones para escalar la capacidad de alojamiento de la flota.

Para empezar, inicie sesión en la [consola de Amazon GameLift Servers](#). En la barra de navegación de la izquierda, vaya a Recursos y seleccione Probar un juego de muestra.

Creación de una solución de alojamiento de juegos con Amazon GameLift Servers

En esta sección se incluyen directrices para crear todos los componentes de una solución de alojamiento de juegos de Amazon GameLift Servers, incluidas prácticas recomendadas de diseño. Tanto si está empezando desde cero como si se dispone a migrar un juego existente, encontrará información exhaustiva que le ayudará a diseñar, implementar y optimizar su infraestructura de alojamiento de juegos con Amazon GameLift Servers.

Componentes de la solución de alojamiento

La solución de alojamiento debe tener alguna versión de los siguientes componentes. Para obtener información más detallada sobre estos componentes y sobre cómo funcionan en conjunto, consulte [Cómo funciona el alojamiento con Amazon GameLift Servers](#).

- El cliente del juego se ejecuta en los dispositivos de los jugadores.
- El servicio de backend coordina la comunicación entre los clientes de juego y Amazon GameLift Servers.
- El software de servidor de juegos administra el juego multijugador.
- El sistema de ubicación empareja a los jugadores con servidores de juegos.
- Las flotas de alojamiento de juegos proporcionan recursos informáticos en ubicaciones geográficas.
- El sistema de administración de alojamiento de juegos supervisa el estado del alojamiento de juegos y administra la capacidad.

Temas

- [Hojas de ruta de desarrollo](#)
- [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#)
- [Creación de un servicio de backend para Amazon GameLift Servers](#)
- [Implementación de flotas de alojamiento para Amazon GameLift Servers](#)
- [Configuración de la ubicación de las sesiones de juego](#)
- [Personalización de su solución de alojamiento de juegos](#)

Hojas de ruta de desarrollo

Las hojas de ruta de desarrollo proporcionan una step-by-step guía estructurada para crear diferentes tipos de soluciones de alojamiento de juegos. Amazon GameLift Servers Cada hoja de ruta está adaptada a un método de alojamiento específico y le guía en las tareas esenciales, desde la configuración inicial hasta la implementación y las pruebas.

Elija la hoja de ruta que mejor se adapte a sus requisitos de alojamiento y limitaciones técnicas. También puede combinar métodos y seguir varias hojas de ruta para crear una solución híbrida que responda a sus necesidades concretas.

Temas

- [Hoja de ruta de desarrollo para el alojamiento con EC2 administrado por Amazon GameLift Servers](#)
- [Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers](#)
- [Hoja de ruta de desarrollo para el alojamiento con Amazon GameLift Servers Anywhere](#)
- [Hoja de ruta de desarrollo para el alojamiento híbrido con Amazon GameLift Servers](#)

Hoja de ruta de desarrollo para el alojamiento con EC2 administrado por Amazon GameLift Servers

En esta hoja de ruta se indica cómo desarrollar una solución de alojamiento de EC2 administrada por Amazon GameLift Servers para su juego multijugador. Amazon GameLift Servers ofrece varias opciones de alojamiento de juegos. Para obtener más información sobre estas opciones, consulte [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

Con el alojamiento Amazon GameLift Servers gestionado, su servidor de juegos se aloja en recursos informáticos virtuales Nube de AWS basados en los que Amazon GameLift Servers se alojan y funcionan en función de su configuración. Usted obtiene la seguridad, la fiabilidad y la disponibilidad global de las instancias de Amazon Elastic Compute Cloud (Amazon EC2), que están aún más optimizadas para su uso con el alojamiento de juegos multijugador. Amazon GameLift Servers simplifica la administración del alojamiento con herramientas como la implementación automática de servidores, la gestión del ciclo de vida y el escalado automático de la capacidad.

Una solución administrada de Amazon GameLift Servers cuenta con los siguientes componentes:

- Una compilación de servidor de juegos, integrada con el SDK del servidor de Amazon GameLift Servers, para la implementación en todas las flotas.
- Un cliente de juego y un servicio de backend, integrados con el AWS SDK, para interactuar con el Amazon GameLift Servers servicio y solicitar sesiones de juego.
- Una cola de Amazon GameLift Servers u otro mecanismo de ubicación para ubicar las nuevas sesiones de juego con servidores de juego disponibles en todas las flotas.
- (Opcional) Un emparejador de FlexMatch para crear partidas multijugador y configurar sesiones de juego para ellas.
- Una o más flotas administradas por Amazon GameLift Servers, con instancias de Amazon Elastic Compute Cloud (Amazon EC2) optimizadas para el alojamiento de juegos multijugador.

Este plan presenta una ruta simplificada para que el juego multijugador funcione correctamente con el alojamiento de EC2 administrado por Amazon GameLift Servers. Cuando tenga los componentes necesarios, podrá continuar con la iteración del desarrollo del juego y personalizar la solución de alojamiento. A medida que se acerque el lanzamiento, consulte [Preparativos para el lanzamiento con alojamiento en Amazon GameLift Servers](#): le resultará de ayuda a fin de preparar la solución de alojamiento para el uso en producción.

Iniciación rápida al uso del complemento de Amazon GameLift Servers para Unreal Engine y Unity

Para agilizar la implementación, pruebe el [complemento de Amazon GameLift Servers](#) para Unreal Engine y Unity. Proporciona flujos de trabajo de la IU guiados que le permitirán implementar rápidamente el servidor de juegos con una configuración mínima, para así poder probar los componentes del juego en acción. Entonces podrá tomarlos como base para crear una solución de alojamiento personalizada para su juego. Para obtener más información, consulte [Preparación de un juego de Unreal o Unity con el complemento de Amazon GameLift Servers](#).

Paso 1: preparar el servidor de juegos para que funcione con Amazon GameLift Servers

Añada funciones al servidor de juegos para que pueda comunicarse con el servicio Amazon GameLift Servers cuando se implemente para el alojamiento.

- Obtenga el SDK del servidor de Amazon GameLift Servers (versión 5.x) para su proyecto de juego. El SDK del servidor está disponible en C++, C# y Go. [Descarga un SDK Amazon GameLift Servers de servidor](#).
- Modifique el código del servidor de juegos para añadir la funcionalidad del SDK del servidor. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Agregue código para inicializar el Amazon GameLift Servers SDK y establecer una WebSocket conexión con el Amazon GameLift Servers servicio. Use la acción `InitSdk()` del SDK del servidor.
 - Añada código para indicar al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Use la acción `ProcessReady()` del SDK del servidor.
 - Implemente las funciones de devolución de llamada `OnStartGameSession()` y `OnProcessTerminate()` necesarias. Con estas funciones, los procesos del servidor de juegos pueden mantener una conexión con el servicio Amazon GameLift Servers, iniciar una sesión de juego cuando Amazon GameLift Servers lo solicite y responder a una petición para finalizar el proceso del servidor de juegos.
 - Añada código para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté finalizando una sesión de juego. Use la acción `ProcessEnding()` del SDK del servidor.
- Empaquete la compilación del servidor de juegos. Cree un script de instalación con los archivos de compilación, las dependencias y el software de soporte. Consulte [Empaquetado de los archivos de compilación del juego](#). Es recomendable utilizar un bucket de Amazon Simple Storage Service (Amazon S3) para almacenar las versiones de la compilación del juego.
- Pruebe la integración del servidor de juegos. Para esta tarea, es recomendable configurar una flota de Amazon GameLift Servers Anywhere para una estación de trabajo local, tal y como se describe en [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#). Para este paso, instale manualmente la compilación del servidor de juegos en el dispositivo de prueba e inicie un proceso de servidor. Utilice la CLI de AWS para solicitar una nueva sesión de juego y compruebe que el servicio Amazon GameLift Servers le pida correctamente al proceso del servidor que inicie una sesión de juego.

Paso 2: preparar el cliente de juego para unirse a sesiones de juego alojadas

Cree un método para que el cliente de juego solicite unirse a una sesión de juego, obtenga la información de conexión y, a continuación, se conecte directamente a una sesión de juego alojada. El método más común consiste en configurar una funcionalidad de servicio de backend que sirva

de intermediario entre el cliente de juego y el servicio Amazon GameLift Servers. Esto protege los recursos de alojamiento y le aporta un mayor control sobre la ubicación de los jugadores en las sesiones de juego.

- Desarrolle una funcionalidad de servicio de backend para el alojamiento. El servicio de backend se comunica con el servicio Amazon GameLift Servers y le envía datos de conexión a un cliente de juego. Esta funcionalidad incluye iniciar sesiones de juego, colocar a los jugadores en juegos y recuperar la información de las sesiones de juego. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Obtenga el AWS SDK Amazon GameLift Servers y agréguelo a su proyecto de servicio de backend. Consulte [Recursos del SDK de Amazon GameLift Servers para servicios de cliente](#).
 - Añada código para inicializar un cliente de Amazon GameLift Servers y almacenar la configuración clave. Consulte [Configuración de la API de Amazon GameLift Servers](#).
 - Añade funciones para activar la acción del AWS SDK `CreateGameSession()` y proporcionar información sobre la conexión de la sesión de juego a un cliente de juego. Consulte [Create a game session on a specific fleet](#).

Llamar a `CreateGameSession()` es un punto de partida muy práctico para solicitar nuevas sesiones de juego. Cuando disponga de un sistema de ubicación de sesiones de juego (consulte el paso 3), sustituirá ese código por una llamada a `StartGameSessionPlacement()` (o a `StartMatchmaking()` si utiliza `FlexMatch`).

Para obtener orientación sobre el diseño de un servicio de backend, consulte [Creación de un servicio de backend para Amazon GameLift Servers](#).

- Añada una función en el cliente de juego que permita a los jugadores unirse a una sesión de juego alojada. El cliente de juego hace solicitudes al servicio de backend, no directamente a Amazon GameLift Servers. Cuando el servicio de backend proporciona la información de conexión de la sesión de juego, el cliente de juego se conecta directamente con la sesión de juego para empezar a jugar.
- Pruebe la integración del cliente de juego. Puede utilizar la misma flota de Amazon GameLift Servers Anywhere con una estación de trabajo local para hacer pruebas.

Para un desarrollo iterativo rápido, o cuando trabaje con equipos de varias personas, le recomendamos que configure un [entorno de pruebas basado en la nube](#). Esta solución del kit de herramientas de Amazon GameLift Servers imita el comportamiento de una flota administrada

por Amazon GameLift Servers, pero le permite actualizar las compilaciones de los servidores de juegos con un tiempo de entrega mínimo.

Paso 3: configurar la ubicación de las sesiones de juego

Personalice cómo desea que Amazon GameLift Servers procese las solicitudes de nuevas sesiones de juego y ubique los servidores de juego disponibles para alojarlos. Amazon GameLift Servers monitoriza automáticamente la disponibilidad de todos los servidores de juego en todas las flotas. Cuando un cliente de juego envía una solicitud para unirse a una sesión de juego, Amazon GameLift Servers busca la mejor ubicación posible en función de un conjunto de prioridades definidas, como el costo, la disponibilidad y un nivel mínimo de latencia.

- Cree una cola de sesiones de juego para colocar una nueva sesión de juego con los servidores de juego disponibles. Las colas son el mecanismo principal para la ubicación de las sesiones de juego. Para obtener instrucciones, consulte [Creación de una cola de sesión de juego](#).
- Como mínimo, añada las flotas de Anywhere como destinos en la cola. Todos los demás ajustes son personalizaciones opcionales.
- En el código del servicio de backend, convierta la llamada a **CreateGameSession()** en **StartGameSessionPlacement()**. Consulte [Create a game session in a multi-location queue](#).
- Cree un mecanismo para notificar a un cliente del juego cuando una sesión de juego ya se pueda unir. Durante el desarrollo, puedes sondear el estado de la sesión del juego mediante una llamada a DescribeGameSessionPlacement. Sin embargo, antes de utilizar una cola para procesar grandes volúmenes, deberá activar las notificaciones de eventos. Consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).
- (Opcional) Añada componentes de emparejamiento de FlexMatch. Para obtener instrucciones, consulte la [guía para desarrolladores de Amazon GameLift Servers FlexMatch](#).

Paso 4: crear flotas administradas y basadas en la nube

Hasta ahora, ha trabajado con flotas de Anywhere autoadministradas para probar los componentes del juego y hacer iteraciones sobre ellos, y también ha refinado la ubicación de las sesiones de juego. La última parte de la solución consiste en configurar el tipo de recursos de alojamiento que necesitará para un sistema de producción. Para empezar a planificar y configurar la producción, es recomendable que pase a trabajar con una flota administrada por Amazon GameLift Servers.

- Empaquete la compilación del servidor de juegos y cárguela en Amazon GameLift Servers. Cree un script de instalación con los archivos de compilación, las dependencias y el software de

soporte. Consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#). Puede cargar su compilación Amazon GameLift Servers mediante la consola o la AWS CLI.

Antes de subir su versión, decida en Región de AWS qué quiere crear una flota. Debe cargar la compilación en la misma región. Para obtener más información sobre cómo elegir la ubicación de una flota, consulte [Ubicaciones geográficas](#).

- Cree una flota de EC2 administrada. Cuando crea una flota, Amazon GameLift Servers comienza inmediatamente a implementar la compilación del servidor de juegos para el alojamiento. Puede configurar muchos elementos de una flota administrada. Para obtener instrucciones, consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#). Como mínimo, haga lo siguiente:
 - Póngale un nombre a la flota y especifique qué compilación del juego cargado desea implementar.
 - Elija instancias bajo demanda para la flota y seleccione un tipo de instancia que esté disponible en la ubicación de la flota. Las flotas de spot son una opción valiosa, pero requieren un nivel adicional de diseño y configuración.
 - Cree una configuración de tiempo de ejecución para la flota. Como mínimo, especifique la ruta de lanzamiento del ejecutable para el servidor de juegos.
 - Especifique la configuración del puerto para permitir que el tráfico entrante acceda a los servidores de juegos.
- Añada las flotas administradas a la cola. En la cola de sesiones de juego, sustituya las flotas de Anywhere por las flotas administradas.
- Pruebe el alojamiento de juegos con las flotas administradas. En este momento, debería poder probar todo el ciclo de alojamiento, con un cliente de juego que solicite una sesión de juego, obtenga los datos de conexión y se conecte correctamente a una sesión de juego.

Paso 5: personalizar las flotas administradas

Durante la preparación para el lanzamiento del juego, tendrá que afinar los recursos de alojamiento administrado. Estos son algunos de los factores en los que deberá pensar:

- Valore la posibilidad de añadir flotas de spot para ahorrar costos. Consulte [Reducción de los costos de alojamiento de juegos con flotas de spot](#).
- Si tu servidor de juegos necesita comunicar otros AWS recursos, configura las funciones de IAM para gestionar el acceso. Consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

- Decida dónde desea ubicar geográficamente los servidores de juegos. Añada ubicaciones remotas a las flotas administradas. Consulte [Personalizaciones de recursos de alojamiento](#).
- Optimice el rendimiento de la flota seleccionando un tipo y tamaño de instancia y configurando el tiempo de ejecución para ejecutar varios procesos de servidor. Consulte [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#).
- Experimente con las opciones de ubicación de sesiones de juego para las flotas administradas, lo que incluye personalizar los ajustes de priorización. Consulte [Personalización de colas de sesiones de juego](#).
- Configure el escalado automático de la capacidad para satisfacer la demanda prevista de jugadores. Consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).
- Configure flotas en espera en otras Regiones de AWS y modifique las colas y el escalado automático para gestionar las conmutaciones por error si es necesario.
- Configure las herramientas de observabilidad del alojamiento, como los análisis y el registro. Consulte [Supervisión de Amazon GameLift Servers](#).
- Automatice la implementación mediante la [infraestructura como código \(IaC\)](#). Consulte [Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation](#).

Amazon GameLift Servers admite el uso de AWS CloudFormation plantillas para cualquier configuración específica de la implementación. También puede utilizarlas AWS Cloud Development Kit (AWS CDK) para definir sus recursos. Amazon GameLift Servers Para obtener más información sobre el AWS CDK, consulte la [Guía para AWS Cloud Development Kit \(AWS CDK\) desarrolladores](#).

Para gestionar el despliegue de sus CloudFormation pilas, le recomendamos que utilice herramientas y servicios de integración y entrega continuas (CI/CD), como: AWS CodePipeline. Estas herramientas le ayudarán con la implementación automática y la aprobación cada vez que cree un archivo binario para el servidor de juegos. Con una CI/CD herramienta o un servicio, el despliegue de recursos para una nueva versión del servidor de juegos puede tener el siguiente aspecto:

- Compilar y probar el archivo binario del servidor de juegos.
- Cargar el archivo binario en Amazon GameLift Servers.
- Implementar nuevas flotas con la nueva compilación.

- Añadir las nuevas flotas a la lista de sesiones de juego y eliminar las flotas de la versión de compilación anterior.
- Cuando las flotas de la versión anterior ya no alojen sesiones de juego activas, elimina las CloudFormation pilas de esas flotas.

Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers

Esta hoja de ruta describe cómo desarrollar una solución de alojamiento administrada por Amazon GameLift Servers para servidores de juegos en contenedor. Los contenedores administrados son solo una de las soluciones de alojamiento que ofrece Amazon GameLift Servers. Para obtener más información acerca de las opciones de alojamiento, consulte [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

Una solución de contenedores administrados por Amazon GameLift Servers tiene los siguientes componentes:

- Una imagen de contenedor con la compilación del servidor de juegos, cargada en el repositorio privado de Amazon Elastic Container Registry (Amazon ECR). La compilación del servidor de juegos está integrada con el SDK del servidor de Amazon GameLift Servers y diseñada para ejecutarse en Linux.
- Un servicio de backend que interactúa con el servicio Amazon GameLift Servers en nombre de los clientes de juegos. El servicio de backend utiliza la funcionalidad de la API de servicio de Amazon GameLift Servers, que forma parte del SDK de AWS .
- Una cola de sesiones de juego de Amazon GameLift Servers u otro mecanismo de ubicación que procese las solicitudes de nuevas sesiones de juego, busque los servidores de juegos disponibles en todas las flotas y solicite al servidor de juegos que inicie una sesión de juego.
- (Opcional) Un emparejador de FlexMatch para crear partidas multijugador y configurar sesiones de juego para ellas.
- Una o más flotas de contenedores, que utilizan instancias de Amazon Elastic Compute Cloud (Amazon EC2) y están optimizadas para el alojamiento de juegos multijugador.

Esta hoja de ruta presenta una ruta simplificada para que el juego multijugador funcione correctamente con contenedores administrados por Amazon GameLift Servers. Cuando tenga los componentes necesarios, podrá continuar con la iteración del desarrollo del juego y personalizar

la solución de alojamiento. A medida que se acerque el lanzamiento, consulte [Preparativos para el lanzamiento con alojamiento en Amazon GameLift Servers](#): le resultará de ayuda a fin de preparar la solución de alojamiento para el uso en producción.

 Acelere la incorporación con estas herramientas para contenedores administrados:

- El [kit básico para contenedores](#) agiliza la integración y la configuración de flotas. Añade características esenciales de administración de sesiones de juego al servidor de juegos y utiliza plantillas preconfiguradas para crear una flota de contenedores y una canalización de implementación automatizada para el servidor de juegos. Tras la implementación, utilice la consola de Amazon GameLift Servers y las herramientas de la API para supervisar el rendimiento de la flota, administrar las sesiones de juego y analizar las métricas.
- Los desarrolladores de Unreal Engine y Unity pueden utilizar los [complementos de Amazon GameLift Servers](#) para integrar el servidor de juegos y crear una flota de contenedores desde el entorno de desarrollo del motor de juegos. Los flujos de trabajo guiados del complemento ayudan a crear una solución rápida y sencilla con alojamiento en la nube mediante contenedores administrados. Puede tomarlos como base para crear una solución de alojamiento personalizada para su juego.

Paso 1: preparar el servidor de juegos para que funcione con Amazon GameLift Servers

Añada funciones al servidor de juegos para que pueda comunicarse con el servicio Amazon GameLift Servers cuando se implemente para el alojamiento.

- Obtenga el SDK del servidor de Amazon GameLift Servers (versión 5.2 o superior) para su proyecto de juego. El SDK del servidor está disponible en C++, C# y Go. [Descargue el SDK del servidor de Amazon GameLift Servers](#). El SDK del servidor está disponible en C++, C# y Go.
- Modifique el código del servidor de juegos para añadir la funcionalidad del SDK del servidor. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Agregue código para inicializar el Amazon GameLift Servers SDK y establecer una WebSocket conexión con el Amazon GameLift Servers servicio. Use la acción `InitSdk()` del SDK del servidor.
 - Añada código para indicar al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Use la acción `ProcessReady()` del SDK del servidor.

- Implemente las funciones de devolución de llamada `OnProcessTerminate()` y `OnStartGameSession()` necesarias. Con estas funciones, los procesos del servidor de juegos pueden mantener una conexión con el servicio Amazon GameLift Servers, iniciar una sesión de juego cuando Amazon GameLift Servers lo solicite y responder a una petición para finalizar el proceso del servidor de juegos.
- Añada código para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté finalizando una sesión de juego. Use la acción `ProcessEnding()` del SDK del servidor.
- Empaquete la compilación del servidor de juegos. Compile el servidor de juegos para que se ejecute en Linux. Prepare la compilación y otros archivos necesarios para ejecutar el servidor de juegos. Si está desarrollando en Windows, este paso puede implicar la configuración de un espacio de trabajo de Linux independiente o el uso de una herramienta como el subsistema de Windows para Linux (WSL). Necesitará un entorno Linux para probar la compilación del servidor de juegos, así como para crear y probar las imágenes de contenedor.
- Pruebe la integración del servidor de juegos. Verifique que el servidor de juegos integrado pueda conectarse al servicio Amazon GameLift Servers y responder a las peticiones. Recomendamos configurar una flota sencilla de Amazon GameLift Servers Anywhere con una estación de trabajo local como host de pruebas, tal y como se describe en [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#). Instale manualmente la compilación del servidor de juegos en el host de prueba e inicie un proceso del servidor. Utilice la AWS CLI para solicitar una nueva sesión de juego y compruebe que el Amazon GameLift Servers servicio solicite correctamente al proceso del servidor que inicie una sesión de juego.

Paso 2: Crear una imagen de contenedor de servidor de juegos

Una vez que haya integrado correctamente el servidor de juegos, cree una imagen de contenedor con el ejecutable del servidor de juegos. Almacénela en un repositorio de Amazon Elastic Container Registry (Amazon ECR) para usarla con Amazon GameLift Servers. Para obtener instrucciones detalladas, consulte [Creación de una imagen de contenedor para Amazon GameLift Servers](#).

- Obtenga la plantilla de Dockerfile para un contenedor de servidor de juegos (proporcionada por Amazon GameLift Servers). Modifique el archivo general de los archivos de compilación del servidor de juegos.
- Creación de una imagen de contenedor del servidor de juegos. Si trabaja en un entorno Linux, utilice la herramienta Docker para crear la imagen.

- Inserción de la imagen de contenedor en Amazon ECR. Cree un repositorio público o privado en Amazon ECR, utilizándolo Cuenta de AWS y en el Región de AWS lugar en el que vaya a desplegar su flota de contenedores. Inserte en él la imagen de contenedor.
- Pruebe sus imágenes de contenedor con su flota de Anywhere (opcional). Es posible que desee probar sus imágenes de contenedor de forma local antes de implementarlas en una flota de contenedores alojada en la nube. Puede utilizar su flota de Amazon GameLift Servers Anywhere existente con una estación de trabajo local para hacer pruebas. Instale y ejecute el contenedor del servidor de juegos y compruebe que: (1) el servicio Amazon GameLift Servers solicita correctamente al proceso del servidor que inicie una sesión de juego; y que (2) un cliente de juego pueda conectarse a la sesión de juego.

Paso 3: Crear una flota de contenedores basada en la nube

Hasta ahora, ha trabajado con una flota de Anywhere autoadministrada para probar los componentes del juego y hacer iteraciones sobre ellos. Cuando disponga de una compilación operativa del servidor de juegos integrada con Amazon GameLift Servers, podrá empezar a configurar los recursos de alojamiento de flotas de contenedores administradas por Amazon GameLift Servers en la nube que necesitará para un entorno de producción.

- Creación de definiciones de grupos de contenedores. Las definiciones de grupo de contenedores describen la arquitectura de contenedores de una flota e identifican qué imágenes de contenedor se deben implementar. Consulte [Creación de una definición de grupo de contenedores para una flota de contenedores de Amazon GameLift Servers](#). Cree su definición de grupo de contenedores en el mismo Región de AWS lugar donde se almacenan las imágenes del contenedor. Para obtener más información sobre cómo elegir la ubicación de una flota, consulte [Ubicaciones geográficas](#). Como mínimo, haga lo siguiente:
 - Cree una definición de grupo de contenedores de servidor de juegos.
 - Añada una definición de contenedor con una imagen de contenedor a su compilación del servidor de juegos.
 - Configure un rango de puertos para los procesos del servidor de juegos del contenedor.
- Cree una flota de contenedores administrada. Cuando crea una flota, Amazon GameLift Servers comienza inmediatamente a implementar la compilación del servidor de juegos para el alojamiento. Puede configurar muchos elementos de una flota administrada. Para obtener instrucciones, consulte [Creación de una flota de contenedores administrada por Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:

- Configure un rol de servicio AWS Identity and Access Management (IAM) para la flota de contenedores. Consulte [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).
- Especifique la definición de grupo de contenedores de servidor de juegos que desea implementar en las instancias de la flota.
- Utilice los valores predeterminados cuando estén disponibles para todos los demás parámetros. Amazon GameLift Servers calcula algunos parámetros para una configuración óptima.
- Añada las flotas de contenedores a la cola. En la cola de sesiones de juego, sustituya la flota de prueba de Anywhere por su flota de contenedores administrada.
- Pruebe el alojamiento de juegos con las flotas de contenedores. En este punto, debería poder probar toda la solución. Inicie un cliente de juego y solicite una sesión de juego a través del servicio de backend. Obtenga información de conexión y conéctese a una sesión de juego en la flota de contenedores.
- Itere las implementaciones de flota. Puede actualizar las definiciones de grupos de contenedores y las configuraciones de flota y, a continuación, implementar las actualizaciones en las flotas existentes.

Paso 4: Prepara tu cliente de juego para unirte a las sesiones de juego alojadas

Cree un método para que el cliente de juego solicite unirse a una sesión de juego, obtenga la información de conexión y, a continuación, se conecte directamente a una sesión de juego alojada. El método más común consiste en configurar una funcionalidad de servicio de backend que sirva de intermediario entre el cliente de juego y el servicio Amazon GameLift Servers. Esto protege los recursos de alojamiento y le aporta un mayor control sobre la ubicación de los jugadores en las sesiones de juego.

- Desarrolle una funcionalidad de servicio de backend para el alojamiento. El servicio de backend se comunica con el servicio Amazon GameLift Servers y le envía datos de conexión a un cliente de juego. Esta funcionalidad incluye iniciar sesiones de juego, colocar a los jugadores en juegos y recuperar la información de las sesiones de juego. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Obtén el AWS SDK Amazon GameLift Servers y agrégalo a tu proyecto de servicio de backend. Consulte [Recursos del SDK de Amazon GameLift Servers para servicios de cliente](#).

- Añada código para inicializar un cliente de Amazon GameLift Servers y almacenar la configuración clave. Consulte [Configuración de la API de Amazon GameLift Servers](#).
- Añade funciones para activar la acción del AWS SDK `CreateGameSession()` y proporcionar información de conexión de la sesión de juego a un cliente de juego. Consulte [Create a game session on a specific fleet](#).

Llamar a `CreateGameSession()` es un punto de partida muy práctico para solicitar nuevas sesiones de juego. Cuando disponga de un sistema de ubicación de sesiones de juego (consulte el paso 3), sustituirá ese código por una llamada a `StartGameSessionPlacement()` (o a `StartMatchmaking()` si utiliza FlexMatch).

Para obtener orientación sobre el diseño de un servicio de backend, consulte [Creación de un servicio de backend para Amazon GameLift Servers](#).

- Añada una función en el cliente de juego que permita a los jugadores unirse a una sesión de juego alojada. El cliente de juego hace solicitudes al servicio de backend, no directamente a Amazon GameLift Servers. Cuando el servicio de backend proporciona la información de conexión de la sesión de juego, el cliente de juego se conecta directamente con la sesión de juego para empezar a jugar.
- Pruebe la integración del cliente de juego. Puede utilizar su flota de Amazon GameLift Servers Anywhere existente con una estación de trabajo local para hacer pruebas. Utilice el nuevo servicio de backend para solicitar una nueva sesión de juego y compruebe que: (1) el servicio Amazon GameLift Servers le pide correctamente al proceso del servidor que inicie una sesión de juego y (2) que un cliente de juego pueda conectarse a la sesión de juego.

Paso 5: Configura la ubicación de las sesiones de juego

Personalice cómo desea que Amazon GameLift Servers procese las solicitudes de nuevas sesiones de juego y ubique los servidores de juego disponibles para alojarlos. Amazon GameLift Servers monitoriza automáticamente la disponibilidad de todos los servidores de juego en todas las flotas. Cuando un cliente de juego envía una solicitud para unirse a una sesión de juego, Amazon GameLift Servers busca la mejor ubicación posible en función de un conjunto de prioridades definidas, como el costo, la disponibilidad y un nivel mínimo de latencia.

- Cree una cola de sesiones de juego para colocar una nueva sesión de juego con los servidores de juego disponibles. Las colas son el mecanismo principal para la ubicación de las sesiones de juego. Para obtener instrucciones, consulte [Creación de una cola de sesión de juego](#).

- Como mínimo, añada las flotas de Anywhere como destinos en la cola. Todos los demás ajustes son personalizaciones opcionales.
- En el código del servicio de backend, convierta la llamada a **CreateGameSession()** en **StartGameSessionPlacement()**. Consulte [Create a game session in a multi-location queue](#).
- Cree un mecanismo para notificar a un cliente del juego cuando una sesión de juego ya se pueda unir. Durante el desarrollo, puede sondear el estado de las sesiones de juego con una llamada a `DescribeGameSessionPlacement`. Sin embargo, antes de utilizar una cola para procesar grandes volúmenes, deberá activar las notificaciones de eventos. Consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).
- Añada el emparejamiento de FlexMatch (opcional). Cree un conjunto de reglas de emparejamiento y una configuración de emparejamiento que se adapte a la cola de sesiones de juego. Para obtener información sobre cómo configurar un sistema de emparejamiento, consulte la [Guía para desarrolladores de Amazon GameLift Servers FlexMatch](#).
- Pruebe el sistema de ubicación. Puede utilizar su flota de Amazon GameLift Servers Anywhere existente con una estación de trabajo local para hacer pruebas. Utilice el servicio de backend para solicitar una nueva sesión de juego y compruebe que el servicio Amazon GameLift Servers solicite correctamente al proceso del servidor que inicie una sesión de juego.

Paso 6: Personalizar la solución de contenedores administrados

Durante la preparación para el lanzamiento del juego, tendrá que afinar los recursos de alojamiento administrado. Estos son algunos de los factores en los que deberá pensar:

- Optimice la configuración de su flota de contenedores. Consulte [Personalización de una flota de contenedores de Amazon GameLift Servers](#).
- Valore la posibilidad de añadir flotas de spot para ahorrar costos. Consulte [Reducción de los costos de alojamiento de juegos con flotas de spot](#).
- Si tu servidor de juegos necesita comunicar otros AWS recursos, configura las funciones de IAM para gestionar el acceso. Consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).
- Decida dónde desea ubicar geográficamente los servidores de juegos. Añada ubicaciones remotas a las flotas administradas. Consulte [Personalizaciones de recursos de alojamiento](#).
- Experimente con las opciones de ubicación de sesiones de juego para las flotas administradas, lo que incluye personalizar los ajustes de priorización. Consulte [Personalización de colas de sesiones de juego](#).

- Configure el escalado automático de la capacidad para satisfacer la demanda prevista de jugadores. Consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).
- Cree flotas en otras Regiones de AWS y modifique las colas y el escalado automático para gestionar las conmutaciones por error según sea necesario.
- Configure las herramientas de observabilidad del alojamiento, como los análisis y el registro. Consulte [Supervisión de Amazon GameLift Servers](#).
- Automatice las implementaciones de flota mediante la [infraestructura como código \(IaC\)](#). Consulte [Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation](#).

Amazon GameLift Servers admite el uso de AWS CloudFormation plantillas para cualquier configuración específica de la implementación. También puede utilizarlas AWS Cloud Development Kit (AWS CDK) para definir sus recursos. Amazon GameLift Servers Para obtener más información sobre el AWS CDK, consulte la [Guía para AWS Cloud Development Kit \(AWS CDK\) desarrolladores](#).

Para gestionar el despliegue de sus CloudFormation pilas, le recomendamos que utilice herramientas y servicios de integración y entrega continuas (CI/CD), como: AWS CodePipeline. Estas herramientas le ayudarán con la implementación automática y la aprobación cada vez que cree un archivo binario para el servidor de juegos. Con una CI/CD herramienta o un servicio, el despliegue de recursos para una nueva versión del servidor de juegos puede tener el siguiente aspecto:

- Compilar y probar el archivo binario del servidor de juegos.
- Cargar el archivo binario en Amazon GameLift Servers.
- Implementar nuevas flotas con la nueva compilación.
- Añadir las nuevas flotas a la lista de sesiones de juego y eliminar las flotas de la versión de compilación anterior.
- Cuando las flotas de la versión anterior ya no alojen sesiones de juego activas, elimina las CloudFormation pilas de esas flotas.

Hoja de ruta de desarrollo para el alojamiento con Amazon GameLift Servers Anywhere

Este plan le muestra cómo desarrollar una solución de alojamiento para un juego multijugador, de modo que pueda usarla con sus propios recursos (hardware en las instalaciones o máquinas

virtuales). Amazon GameLift Servers ofrece varias opciones de alojamiento de juegos. Para obtener más información sobre estas opciones, consulte [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

Con el alojamiento de Amazon GameLift Servers Anywhere, el servidor de juegos se aloja en los recursos de computación que usted gestiona y suministra. Puede crear una flota de Anywhere con las configuraciones que necesite y con una ubicación geográfica que coincida con la de los jugadores. Amazon GameLift Servers ofrece las siguientes características para las flotas de Anywhere:

- Se encarga automáticamente del proceso de ubicación de las sesiones de juego, con base en su configuración; esto incluye lo siguiente:
 - Hace un seguimiento de la disponibilidad de los servidores de juegos en todas las flotas de Anywhere.
 - Procesa las solicitudes de juego desde el servicio del cliente de juegos y empareja las solicitudes de juego con los servidores disponibles.
 - Le pide a los servidores de juegos de las flotas de Anywhere que inicien las sesiones de juego.
 - Le devuelve los detalles de la conexión a los clientes de juego.
- Recopila métricas de rendimiento para el proceso de ubicación de las sesiones, así como métricas de uso para los jugadores y las sesiones de juego.
- Es compatible con el conjunto completo de características de emparejamiento de FlexMatch, así que puede crear un emparejador e integrarlo en el sistema de ubicación de las sesiones de juego.
- Dispone del agente de Amazon GameLift Servers para gestionar tareas clave de administración de host en una flota de Anywhere.
- Admite la combinación con flotas administradas de Amazon GameLift Servers, lo que permite disfrutar de una solución híbrida y flexible.

Una solución de Amazon GameLift Servers Anywhere cuenta con los siguientes componentes:

- Una compilación de servidor de juegos, integrada con el SDK del servidor de Amazon GameLift Servers, para la implementación en todas las flotas.
- Un cliente de juego y un servicio de backend, integrados con el AWS SDK, para interactuar con el Amazon GameLift Servers servicio y solicitar sesiones de juego.
- Una cola de Amazon GameLift Servers u otro mecanismo de ubicación para ubicar las nuevas sesiones de juego con servidores de juego disponibles en todas las flotas.

- (Opcional) Un emparejador de FlexMatch para crear partidas multijugador y configurar sesiones de juego para ellas.
- Una o más flotas de Amazon GameLift Servers Anywhere con sus recursos de alojamiento (en las instalaciones o de otro tipo), administrados con las herramientas de implementación y administración de la configuración existentes (Opcionalmente, puedes usar el AWS Systems Manager.)

Este plan presenta una ruta simplificada para que el juego multijugador funcione correctamente con el alojamiento de Amazon GameLift Servers Anywhere. Cuando tenga los componentes necesarios, podrá continuar con la iteración del desarrollo del juego y personalizar la solución de alojamiento. A medida que se acerque el lanzamiento, consulte [Preparativos para el lanzamiento con alojamiento en Amazon GameLift Servers](#): le resultará de ayuda a fin de preparar la solución de alojamiento para el uso en producción.

Iniciación rápida al uso del complemento de Amazon GameLift Servers para Unreal Engine y Unity

Para agilizar la implementación, pruebe el [complemento de Amazon GameLift Servers](#) para Unreal Engine y Unity. Proporciona flujos de trabajo de la IU guiados que le permitirán implementar rápidamente el servidor de juegos con una configuración mínima, para así poder probar los componentes del juego en acción. Entonces podrá tomarlos como base para crear una solución de alojamiento personalizada para su juego. Para obtener más información, consulte [Preparación de un juego de Unreal o Unity con el complemento de Amazon GameLift Servers](#).

Paso 1: preparar el servidor de juegos para que funcione con Amazon GameLift Servers

Añada funciones al servidor de juegos para que pueda comunicarse con el servicio Amazon GameLift Servers cuando se implemente para el alojamiento.

- Obtenga el SDK del servidor de Amazon GameLift Servers (versión 5.x) para su proyecto de juego. El SDK del servidor está disponible en C++, C# y Go. [Descargue un SDK Amazon GameLift Servers de servidor](#).
- Modifique el código del servidor de juegos para añadir la funcionalidad del SDK del servidor. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:

- Agregue código para inicializar el Amazon GameLift Servers SDK y establecer una WebSocket conexión con el Amazon GameLift Servers servicio. Utilice la acción `InitSdk()` del SDK del servidor e incluya los parámetros del servidor, que son necesarios para una flota de Anywhere.
- Añada código para indicar al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Use la acción `ProcessReady()` del SDK del servidor.
- Implemente las funciones de devolución de llamada `OnStartGameSession()` y `OnProcessTerminate()` necesarias. Con estas funciones, los procesos del servidor de juegos pueden mantener una conexión con el servicio Amazon GameLift Servers, iniciar una sesión de juego cuando Amazon GameLift Servers lo solicite y responder a una petición para finalizar el proceso del servidor de juegos.
- Añada código para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté finalizando una sesión de juego. Use la acción `ProcessEnding()` del SDK del servidor.
- Empaquete la compilación del servidor de juegos. Cree un script de instalación con los archivos de compilación, las dependencias y el software de soporte. Consulte [Empaquetado de los archivos de compilación del juego](#). Es recomendable utilizar un bucket de Amazon Simple Storage Service (Amazon S3) para almacenar las versiones de la compilación del juego.
- Pruebe la integración del servidor de juegos. Para esta tarea, es recomendable configurar una flota de Amazon GameLift Servers Anywhere para una estación de trabajo local, tal y como se describe en [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#). Para este paso, instale manualmente la compilación del servidor de juegos en el dispositivo de prueba e inicie un proceso de servidor. Utilice la CLI de AWS para solicitar una nueva sesión de juego y compruebe que el servicio Amazon GameLift Servers le pida correctamente al proceso del servidor que inicie una sesión de juego.

Paso 2: preparar el cliente de juego para unirse a sesiones de juego alojadas

Cree un método para que el cliente de juego solicite unirse a una sesión de juego, obtenga la información de conexión y, a continuación, se conecte directamente a una sesión de juego alojada. El método más común consiste en configurar una funcionalidad de servicio de backend que sirva de intermediario entre el cliente de juego y el servicio Amazon GameLift Servers. Esto protege los recursos de alojamiento y le aporta un mayor control sobre la ubicación de los jugadores en las sesiones de juego.

- Desarrolle una funcionalidad de servicio de backend para el alojamiento. El servicio de backend se comunica con el servicio Amazon GameLift Servers y le envía datos de conexión a un cliente de juego. Esta funcionalidad incluye iniciar sesiones de juego, colocar a los jugadores en juegos

y recuperar la información de las sesiones de juego. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:

- Obtenga el AWS SDK Amazon GameLift Servers y agréguelo a su proyecto de servicio de backend. Consulte [Recursos del SDK de Amazon GameLift Servers para servicios de cliente](#).
- Añada código para inicializar un cliente de Amazon GameLift Servers y almacenar la configuración clave. Consulte [Configuración de la API de Amazon GameLift Servers](#).
- Añade funciones para activar la acción del AWS SDK `CreateGameSession()` y proporcionar información de conexión de la sesión de juego a un cliente de juego. Consulte [Create a game session on a specific fleet](#).

Llamar a `CreateGameSession()` es un punto de partida muy práctico para solicitar nuevas sesiones de juego. Cuando disponga de un sistema de ubicación de sesiones de juego (consulte el paso 3), sustituirá ese código por una llamada a `StartGameSessionPlacement()` (o a `StartMatchmaking()` si utiliza FlexMatch).

Para obtener orientación sobre el diseño de un servicio de backend, consulte [Creación de un servicio de backend para Amazon GameLift Servers](#).

- Añada una función en el cliente de juego que permita a los jugadores unirse a una sesión de juego alojada. El cliente de juego hace solicitudes al servicio de backend, no directamente a Amazon GameLift Servers. Cuando el servicio de backend proporciona la información de conexión de la sesión de juego, el cliente de juego se conecta directamente con la sesión de juego para empezar a jugar.
- Pruebe la integración del cliente de juego. Puede utilizar la misma flota de Amazon GameLift Servers Anywhere con estaciones de trabajo locales para hacer pruebas.

Paso 3: configurar la ubicación de las sesiones de juego

Personalice cómo desea que Amazon GameLift Servers procese las solicitudes de nuevas sesiones de juego y ubique los servidores de juego disponibles para alojarlos. Amazon GameLift Servers monitoriza automáticamente la disponibilidad de todos los servidores de juego en todas las flotas. Cuando un cliente de juego envía una solicitud para unirse a una sesión de juego, Amazon GameLift Servers busca la mejor ubicación posible en función de un conjunto de prioridades definidas, como el costo, la disponibilidad y un nivel mínimo de latencia.

- Cree una cola de sesiones de juego para colocar una nueva sesión de juego con los servidores de juego disponibles. Las colas son el mecanismo principal para la ubicación de las sesiones de juego. Para obtener instrucciones, consulte [Creación de una cola de sesión de juego](#).
- Como mínimo, añada las flotas de Anywhere como destinos en la cola. Todos los demás ajustes son opcionales.
- En el código del servicio de backend, convierta la llamada a **CreateGameSession()** en **StartGameSessionPlacement()**. Consulte [Create a game session in a multi-location queue](#).
- Cree un mecanismo para notificar a un cliente del juego cuando una sesión de juego ya se pueda unir. Durante el desarrollo, puedes sondear el estado de la sesión del juego mediante una llamada a DescribeGameSessionPlacement. Sin embargo, antes de utilizar una cola para procesar grandes volúmenes, deberá activar las notificaciones de eventos. Consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).
- (Opcional) Añada componentes de emparejamiento de FlexMatch. Para obtener instrucciones, consulte la [guía para desarrolladores de Amazon GameLift Servers FlexMatch](#).

Paso 4: configurar una flota de Anywhere con el agente de Amazon GameLift Servers

Hasta ahora, ha estado trabajando con dispositivos locales (registrados como recursos de computación de flotas de Anywhere) para probar los componentes del juego y hacer iteraciones con ellos. El siguiente paso consiste en configurar el tipo de flota que necesitará para un sistema de producción. Para estos recursos, utilice el agente de Amazon GameLift Servers para administrar algunas tareas importantes de administración del alojamiento en recursos de computación. Para obtener más información, consulte [Uso el agente de Amazon GameLift Servers](#).

- Obtenga el agente de Amazon GameLift Servers y añádalo al paquete de instalación del servidor de juegos. Obtenga y compile el código fuente del agente, que está disponible en el [repositorio de GitHub del agente de Amazon GameLift Servers](#). Ponga el archivo ejecutable JAR resultante en el mismo directorio que el ejecutable de la compilación del juego.
- Modifique el script de arranque del agente según sea necesario. Asegúrese de que el ejecutable del agente se inicie en cuanto el recurso de computación comience a ejecutarse. Consulte el archivo readme del repositorio del agente para obtener ayuda sobre la instalación y ejecución del agente en los recursos de computación de alojamiento. El comando de lanzamiento debería incluir opciones para especificar, como mínimo, la Región de AWS y el ID de flota de Anywhere, una ubicación personalizada y un nombre de recurso de computación.

El agente se encarga automáticamente de las siguientes tareas, así que ya puede eliminarlas si ha estado gestionándolas con scripts:

- Llama a `RegisterCompute()` para añadir el recurso de computación a una flota de Anywhere.
- Llama a `GetComputeAuthToken()` para autenticar los servidores de juego cuando se conectan al servicio Amazon GameLift Servers. El agente se encarga de obtener y actualizar el token de autenticación; todos los procesos del servidor de juegos que se ejecutan en el recurso de computación podrán usarlo.
- Inicia nuevos procesos de servidor en el recurso de computación, en función de un conjunto de instrucciones de tiempo de ejecución.
- Cree una configuración de tiempo de ejecución para recursos de computación en la flota de Anywhere. Puede usar la Amazon GameLift Servers consola o la AWS CLI para crear o modificar instrucciones de tiempo de ejecución para la flota. El agente sigue estas instrucciones y solicita actualizaciones de forma periódica al servicio Amazon GameLift Servers.
- Configure o modifique la cola de sesiones de juego según sea necesario. Cree una nueva cola (o actualice una existente) para el uso de las flotas de Anywhere implementadas con el agente de Amazon GameLift Servers.
- Pruebe la integración del agente con las flotas de Anywhere. Compruebe que el agente esté iniciando correctamente los procesos del servidor, de acuerdo con la configuración del tiempo de ejecución.

Paso 5: personalizar las flotas de Anywhere

Durante la preparación para el lanzamiento del juego, tendrá que afinar los recursos de alojamiento administrado. Estos son algunos de los factores en los que deberá pensar:

- Automatice el proceso de inicio y apagado de los recursos de computación según sea necesario, lo que incluye la instalación y la ejecución del software del servidor de juegos. Reciclar los recursos de computación resulta útil para garantizar que estos se actualicen periódicamente, y el apagado de dichos recursos cuando no son necesarios puede ahorrar costos.
- Si tu servidor de juegos necesita comunicar otros AWS recursos, configura las funciones de IAM para administrar el acceso. Consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).
- Decida dónde desea ubicar geográficamente los servidores de juegos. Añada ubicaciones remotas a las flotas administradas. Consulte [Personalizaciones de recursos de alojamiento](#).

- Optimice el rendimiento de las flotas; para ello, seleccione las configuraciones de los recursos de computación y, a continuación, configure las instrucciones de tiempo de ejecución de tal modo que se ejecute un número óptimo de procesos de servidor por cada recurso de computación.
- Experimente con las opciones de ubicación de sesiones de juego para las flotas administradas, lo que incluye personalizar los ajustes de priorización. Consulte [Personalización de colas de sesiones de juego](#).
- Cree mecanismos para gestionar el escalado automático o manual de la capacidad, a fin de satisfacer la demanda prevista de jugadores. Piense en cuáles deberían ser los factores que determinen si el sistema debe aumentar o disminuir la cantidad de recursos de computación disponible para alojar sesiones de juego.
- Diseñe e implemente medidas de conmutación por error a otros recursos cuando sea necesario.
- Configure las herramientas de observabilidad del alojamiento, como los análisis y el registro. Consulte [Supervisión de Amazon GameLift Servers](#).

Hoja de ruta de desarrollo para el alojamiento híbrido con Amazon GameLift Servers

En esta hoja de ruta se indica cómo desarrollar una solución de alojamiento para juegos multijugador. Amazon GameLift Servers ofrece varias opciones de alojamiento de juegos; para obtener más información sobre estas opciones, consulte [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

Una solución híbrida utiliza una combinación de recursos de alojamiento, como los recursos basados en la nube y administrados por Amazon GameLift Servers o sus propios recursos de alojamiento autoadministrados. Para obtener información más detallada sobre el alojamiento híbrido, consulte el artículo [Alojamiento de servidor de juegos híbrido con Amazon GameLift Servers Anywhere](#). Con Amazon GameLift Servers, puede configurar una solución híbrida que utilice componentes y procesos comunes, de forma que pueda administrar de forma centralizada una flota global y mover cargas con facilidad entre todos los tipos de recursos.

Una arquitectura híbrida consta de los siguientes componentes:

- Una sola compilación de servidor de juegos, integrada con el SDK del servidor de Amazon GameLift Servers, para la implementación en todas las flotas.
- Un único cliente de juegos y servicio de backend, integrado con el SDK de AWS , para interactuar con el servicio Amazon GameLift Servers y solicitar sesiones de juego.

- Una cola de Amazon GameLift Servers compartida para colocar nuevas sesiones de juego con servidores de juego disponibles y equilibrio de cargas en todas las flotas.
- (Opcional) Un agente de administración de procesos, que se implementa con una flota de Anywhere, a fin de simplificar las tareas de administración de los procesos del servidor en todos los recursos de computación de todas las flotas.
- (Opcional) Un emparejador de FlexMatch para crear partidas multijugador y configurar sesiones de juego para ellas.
- Una o más flotas administradas por Amazon GameLift Servers, con instancias de Amazon Elastic Compute Cloud (Amazon EC2) optimizadas para el alojamiento de juegos multijugador.
- Una o más flotas de Amazon GameLift Servers Anywhere con sus recursos de alojamiento existentes (en las instalaciones o de otro tipo), lo que incluye las herramientas de implementación y administración de la configuración (si lo prefiere, puede utilizar AWS Systems Manager).

Este plan presenta una ruta simplificada para que el juego multijugador funcione correctamente con una solución de alojamiento híbrida de Amazon GameLift Servers. Cuando tenga los componentes necesarios, podrá continuar con la iteración del desarrollo del juego y personalizar la solución de alojamiento. A medida que se acerque el lanzamiento, consulte [Preparativos para el lanzamiento con alojamiento en Amazon GameLift Servers](#): le resultará de ayuda a fin de preparar la solución de alojamiento para el uso en producción.

Iniciación rápida al uso del complemento de Amazon GameLift Servers

Si está desarrollando proyectos con Unreal Engine o Unity, comience a configurar el juego para el alojamiento con el complemento de Amazon GameLift Servers. Con el complemento, puedes añadirlo Amazon GameLift Servers SDKs a tu proyecto de juego y utilizar los flujos de trabajo guiados para crear una versión sencilla y funcional de una solución de alojamiento híbrida con una flota de Anywhere y una flota Amazon GameLift Servers gestionada. Luego, sobre la base de esos elementos fundamentales, podrá seguir con el desarrollo y personalizar lo que sea necesario.

Paso 1: preparar el servidor de juegos para que funcione con Amazon GameLift Servers

Añada funciones al servidor de juegos para que pueda comunicarse con el servicio Amazon GameLift Servers cuando se implemente para el alojamiento. La funcionalidad necesaria es la

misma, con independencia de si el servidor del juego se ejecuta en una flota administrada por Amazon GameLift Servers como si se ejecuta en una flota de Anywhere.

- Obtenga el SDK del servidor de Amazon GameLift Servers (versión 5.x) para su proyecto de juego. El SDK del servidor está disponible en C++, C# y Go. [Descargue el SDK del servidor de Amazon GameLift Servers](#).
- Modifique el código del servidor de juegos para añadir la funcionalidad del SDK del servidor. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Agregue código para inicializar el Amazon GameLift Servers SDK y establecer una WebSocket conexión con el Amazon GameLift Servers servicio. Use la acción `InitSdk()` del SDK del servidor. Incluya código para especificar los parámetros del servidor cuando se ejecute en el recurso de computación de una flota de Anywhere.
 - Añada código para indicar al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Use la acción `ProcessReady()` del SDK del servidor.
 - Implemente las funciones de devolución de llamada `OnStartGameSession()` y `OnProcessTerminate()` necesarias. Con estas funciones, los procesos del servidor de juegos pueden mantener una conexión con el servicio Amazon GameLift Servers, iniciar una sesión de juego cuando Amazon GameLift Servers lo solicite y responder a una petición para finalizar el proceso del servidor de juegos.
 - Añada código para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté finalizando una sesión de juego. Use la acción `ProcessEnding()` del SDK del servidor.
- Empaquete la compilación del servidor de juegos. Cree un script de instalación con los archivos de compilación, las dependencias y el software de soporte. Consulte [Empaquetado de los archivos de compilación del juego](#). Es recomendable utilizar un bucket de Amazon Simple Storage Service (Amazon S3) para almacenar las versiones de la compilación del juego.
- Pruebe la integración del servidor de juegos. Para esta tarea, es recomendable configurar una flota de Amazon GameLift Servers Anywhere con una estación de trabajo local, tal y como se describe en [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#). Para este paso, instale manualmente la compilación del servidor de juegos en el dispositivo de prueba e inicie un proceso de servidor. Utilice la CLI de AWS para solicitar una nueva sesión de juego y compruebe que el servicio Amazon GameLift Servers le pida correctamente al proceso del servidor que inicie una sesión de juego.

Paso 2: preparar el cliente de juego para unirse a sesiones de juego alojadas

Cree un método para que el cliente de juego solicite unirse a una sesión de juego, obtenga la información de conexión y, a continuación, se conecte directamente a una sesión de juego alojada. El método más común consiste en configurar una funcionalidad de servicio de backend que sirva de intermediario entre el cliente de juego y el servicio Amazon GameLift Servers. Esto protege los recursos de alojamiento y le aporta un mayor control sobre la ubicación de los jugadores en las sesiones de juego.

- Desarrolle una funcionalidad de servicio de backend para el alojamiento. El servicio de backend se comunica con el servicio Amazon GameLift Servers y le envía datos de conexión a un cliente de juego. Esta funcionalidad incluye iniciar sesiones de juego, colocar a los jugadores en juegos y recuperar la información de las sesiones de juego. Para obtener instrucciones, consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#). Como mínimo, haga lo siguiente:
 - Obtenga el AWS SDK Amazon GameLift Servers y agréguelo a su proyecto de servicio de backend. Consulte [Recursos del SDK de Amazon GameLift Servers para servicios de cliente](#).
 - Añada código para inicializar un cliente de Amazon GameLift Servers y almacenar la configuración clave. Consulte [Configuración de la API de Amazon GameLift Servers](#).
 - Añade funciones para activar la acción del AWS SDK `CreateGameSession()` y proporcionar información sobre la conexión de la sesión de juego a un cliente de juego. Consulte [Create a game session on a specific fleet](#).

Llamar a `CreateGameSession()` es un punto de partida muy práctico para solicitar nuevas sesiones de juego. Cuando disponga de un sistema de ubicación de sesiones de juego (consulte el paso 3), sustituirá ese código por una llamada a `StartGameSessionPlacement()` (o a `StartMatchmaking()` si utiliza FlexMatch).

Para obtener orientación sobre el diseño de un servicio de backend, consulte [Creación de un servicio de backend para Amazon GameLift Servers](#).

- Añada una función en el cliente de juego que permita a los jugadores unirse a una sesión de juego alojada. El cliente de juego hace solicitudes al servicio de backend, no directamente a Amazon GameLift Servers. Cuando el servicio de backend proporciona la información de conexión de la sesión de juego, el cliente de juego se conecta directamente con la sesión de juego para empezar a jugar.
- Pruebe la integración del cliente de juego. Puede utilizar la misma flota de Amazon GameLift Servers Anywhere con una estación de trabajo local para hacer pruebas.

Durante la fase de desarrollo, si quiere probar el comportamiento de la compilación del juego en una flota administrada por Amazon GameLift Servers, es recomendable que configure también un [entorno de pruebas basado en la nube](#). Esta solución del kit de herramientas de Amazon GameLift Servers imita el comportamiento de una flota administrada, pero le permite actualizar las compilaciones de los servidores de juegos con un tiempo de entrega mínimo.

Paso 3: configurar la ubicación de las sesiones de juego

Personalice cómo desea que Amazon GameLift Servers procese las solicitudes de nuevas sesiones de juego y ubique los servidores de juego disponibles para alojarlos. Amazon GameLift Servers monitoriza automáticamente la disponibilidad de todos los servidores de juego en todas las flotas. Cuando un cliente de juego envía una solicitud para unirse a una sesión de juego, Amazon GameLift Servers busca la mejor ubicación posible en función de un conjunto de prioridades definidas, como el costo, la disponibilidad y un nivel mínimo de latencia.

- Cree una cola de sesiones de juego para colocar una nueva sesión de juego con los servidores de juego disponibles. Las colas son el mecanismo principal para la ubicación de las sesiones de juego. Para obtener instrucciones, consulte [Creación de una cola de sesión de juego](#).
- Como mínimo, añada las flotas de Anywhere como destinos en la cola. Todos los demás ajustes son personalizaciones opcionales.
- En el código del servicio de backend, convierta la llamada a **CreateGameSession()** en **StartGameSessionPlacement()**. Consulte [Create a game session in a multi-location queue](#).
- Cree un mecanismo para notificar a un cliente del juego cuando una sesión de juego ya se pueda unir. Durante el desarrollo, puede sondear el estado de las sesiones de juego con una llamada a `DescribeGameSessionPlacement`. Sin embargo, antes de utilizar una cola para procesar grandes volúmenes, deberá activar las notificaciones de eventos. Consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).
- (Opcional) Añada componentes de emparejamiento de FlexMatch. Para obtener instrucciones, consulte la [guía para desarrolladores de Amazon GameLift Servers FlexMatch](#).

Paso 4: configurar una flota de Anywhere con el agente de Amazon GameLift Servers

Hasta ahora, ha estado trabajando con dispositivos locales (registrados como recursos de computación de flotas de Anywhere) para probar los componentes del juego y hacer iteraciones con ellos. El siguiente paso es configurar el tipo de flotas que necesitará para un sistema de producción. Comience con una flota de Anywhere y añada el agente de Amazon GameLift Servers para gestionar

algunas tareas clave de administración de hosts en el entorno informático. Para obtener más información, consulte [Uso el agente de Amazon GameLift Servers](#).

- Obtenga el agente de Amazon GameLift Servers y añádalo al paquete de instalación del servidor de juegos. Obtenga y compile el código fuente del agente, que está disponible en el [repositorio de GitHub del agente de Amazon GameLift Servers](#). Ponga el archivo ejecutable JAR resultante en el mismo directorio que el ejecutable de la compilación del juego.
- Modifique el script de arranque del agente según sea necesario. Asegúrese de que el ejecutable del agente se inicie en cuanto el recurso de computación comience a ejecutarse. Consulte el archivo readme del repositorio del agente para obtener ayuda sobre la instalación y ejecución del agente en los recursos de computación de alojamiento. El comando de lanzamiento debe incluir opciones para especificar, como mínimo, el identificador de flota de Anywhere Región de AWS, una ubicación personalizada y un nombre de cálculo.

El agente se encarga automáticamente de las siguientes tareas, así que ya puede eliminarlas si ha estado gestionándolas con scripts:

- Llama a `RegisterCompute()` para añadir el recurso de computación a una flota de Anywhere.
- Llama a `GetComputeAuthToken()` para autenticar los servidores de juego cuando se conectan al servicio Amazon GameLift Servers. El agente se encarga de obtener y actualizar el token de autenticación; todos los procesos del servidor de juegos que se ejecutan en el recurso de computación podrán usarlo.
- Inicia nuevos procesos de servidor en el recurso de computación, en función de un conjunto de instrucciones de tiempo de ejecución.
- Cree una configuración de tiempo de ejecución para recursos de computación en la flota de Anywhere. Como mínimo, especifique la ruta de lanzamiento del ejecutable para el servidor de juegos. Puede usar la Amazon GameLift Servers consola o la AWS CLI para crear o modificar instrucciones de tiempo de ejecución para la flota. El agente sigue estas instrucciones y solicita actualizaciones de forma periódica al servicio Amazon GameLift Servers.
- Configure o modifique la cola de sesiones de juego según sea necesario. Cree una nueva cola (o actualice una existente) y designe un destino para la flota de Anywhere implementada con el agente de Amazon GameLift Servers.
- Pruebe la integración del agente con las flotas de Anywhere. Compruebe que el agente esté iniciando correctamente los procesos del servidor, de acuerdo con la configuración del tiempo de ejecución.

Paso 5: crear una flota administrada basada en la nube

Cree una flota de EC2 administrada por Amazon GameLift Servers para complementar la flota de Anywhere. Si ha configurado un entorno de pruebas basado en la nube en el paso 2 para acelerar el desarrollo, planea crear una flota administrada tras completar la mayor parte del desarrollo y las pruebas del juego. Necesita una flota totalmente administrada para configurar y probar ajustes adicionales, como el escalado automático de la capacidad.

- Empaquete la compilación del servidor de juegos y cárguela en Amazon GameLift Servers. Cree un script de instalación con los archivos de compilación, las dependencias y el software de soporte. Puede usar el mismo software de compilación en la flota de Anywhere y en la flota administrada. Consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#). Puede cargar su compilación Amazon GameLift Servers mediante la consola o la AWS CLI.

Antes de cargar la compilación, decida en qué Región de AWS quiere crear la flota administrada. Debe cargar la compilación en la misma región. Para obtener más información sobre cómo elegir la ubicación de una flota, consulte [Ubicaciones geográficas](#).

- Cree una flota de EC2 administrada. Puede usar la Amazon GameLift Servers consola o la AWS CLI para crear una flota gestionada. Cuando crea una flota, Amazon GameLift Servers comienza inmediatamente a implementar la compilación del servidor de juegos para el alojamiento. Puede configurar muchos elementos de una flota administrada. Para obtener instrucciones, consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#). Como mínimo, haga lo siguiente:
 - Póngale un nombre a la flota y especifique qué compilación del juego cargado desea implementar.
 - Elija instancias bajo demanda para la flota y seleccione un tipo de instancia que esté disponible en la ubicación de la flota. Las flotas de spot son una opción valiosa, pero requieren un nivel adicional de diseño y configuración.
 - Cree una configuración de tiempo de ejecución con ajustes similares a los que ha usado con la flota de Anywhere. Como mínimo, especifique la ruta de lanzamiento del ejecutable para el servidor de juegos.
 - Especifique la configuración del puerto para permitir que el tráfico entrante acceda a los servidores de juegos.
- Añada la flota administrada a la cola de sesiones de juego compartida. Actualice la cola del paso 4 para que incluya los destinos de la flota administrada y de la flota de Anywhere implementada con el agente de Amazon GameLift Servers.

- Prueba el alojamiento de juegos con las flotas administradas. En este momento, debería poder probar todo el ciclo de alojamiento, con un cliente de juego que solicite una sesión de juego, obtenga los datos de conexión y se conecte correctamente a una sesión de juego.

Paso 6: personalizar las flotas

Durante la preparación para el lanzamiento del juego, tendrá que afinar las soluciones de alojamiento. Estos son algunos de los factores en los que deberá pensar:

- Para las flotas de Anywhere, automatice el proceso de inicio y apagado de los recursos de computación según sea necesario, lo que incluye la instalación y la ejecución del software del servidor de juegos. Reciclar los recursos de computación resulta útil para garantizar que estos se actualicen periódicamente, y el apagado de dichos recursos cuando no son necesarios puede ahorrar costos.
- Si el servidor de juegos necesita comunicarse con otros recursos de AWS, configure los roles de IAM para administrar el acceso. Consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).
- Decida dónde desea ubicar geográficamente los servidores de juegos. Añada ubicaciones remotas a las flotas administradas. Consulte [Personalizaciones de recursos de alojamiento](#).
- En el caso de las flotas administradas, valore la opción de usar flotas de Spot para ahorrar costos. Consulte [Reducción de los costos de alojamiento de juegos con flotas de spot](#).
- Optimice el rendimiento de las flotas; para ello, seleccione las configuraciones de los recursos de computación y, a continuación, configure las instrucciones de tiempo de ejecución de tal modo que se ejecute el número óptimo de procesos de servidor por cada recurso de computación. Haga esto tanto para las flotas de Anywhere como para las flotas administradas. Consulte [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#).
- Experimente con las opciones de ubicación de sesiones de juego para las flotas administradas, lo que incluye personalizar los ajustes de priorización. Consulte [Personalización de colas de sesiones de juego](#).
- En el caso de las flotas administradas, configure el escalado automático de la capacidad para satisfacer la demanda prevista de jugadores. Consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).
- Para las flotas de Anywhere, cree mecanismos para gestionar el escalado automático o manual de la capacidad, a fin de satisfacer la demanda prevista de jugadores.

- Diseñe e implemente medidas de conmutación por error a otros recursos cuando sea necesario. Configure flotas en espera en otras Regiones de AWS y modifique las colas y el escalado automático para la gestión de conmutaciones por error en caso de que sea necesario.
- Configure las herramientas de observabilidad del alojamiento, como los análisis y el registro. Consulte [Supervisión de Amazon GameLift Servers](#). Cree grupos de métricas a fin de agregar análisis para todos los recursos de alojamiento.
- Automatice la implementación mediante la [infraestructura como código \(IaC\)](#). Consulte [Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation](#).

Amazon GameLift Servers admite el uso de AWS CloudFormation plantillas para cualquier configuración específica de la implementación. También puede utilizarlas AWS Cloud Development Kit (AWS CDK) para definir sus recursos. Amazon GameLift Servers Para obtener más información sobre el AWS CDK, consulte la [Guía para AWS Cloud Development Kit \(AWS CDK\) desarrolladores](#).

Para gestionar el despliegue de sus CloudFormation pilas, le recomendamos que utilice herramientas y servicios de integración y entrega continuas (CI/CD), como: AWS CodePipeline. Estas herramientas le ayudarán con la implementación automática y la aprobación cada vez que cree un archivo binario para el servidor de juegos. Con una CI/CD herramienta o un servicio, el despliegue de recursos para una nueva versión del servidor de juegos puede tener el siguiente aspecto:

- Compilar y probar el archivo binario del servidor de juegos.
- Cargar el archivo binario en Amazon GameLift Servers.
- Implementar nuevas flotas con la nueva compilación.
- Añadir las nuevas flotas a la lista de sesiones de juego y eliminar las flotas de la versión de compilación anterior.
- Cuando las flotas de la versión anterior ya no alojen sesiones de juego activas, eliminar las pilas de CloudFormation de esas flotas.

Preparación de un juego para su alojamiento con Amazon GameLift Servers

Para poder implementar el juego para alojarlo en Amazon GameLift Servers, primero debe preparar los componentes del servidor de juegos y del cliente de juego. Esto implica integrar el servidor de

juegos con el SDK del servidor de Amazon GameLift Servers y crear un servicio de backend que se encargue de la autenticación de los jugadores y de la administración de las sesiones de juego.

La integración del SDK del servidor permite al servidor de juegos comunicarse con el servicio de Amazon GameLift Servers, informar sobre su estado y administrar las sesiones de juego. El servicio de backend actúa como intermediario entre los clientes de juego y el servicio de Amazon GameLift Servers, administrando las solicitudes de los jugadores y coordinando la ubicación de las sesiones de juego.

Temas

- [Preparación de un juego de Unreal o Unity con el complemento de Amazon GameLift Servers](#)
- [Integración de un servidor de juegos con Amazon GameLift Servers](#)
- [Empaquetado de una compilación de servidor de juegos para su implementación](#)
- [client/server Interacciones del juego con Amazon GameLift Servers](#)

Preparación de un juego de Unreal o Unity con el complemento de Amazon GameLift Servers

El complemento de Amazon GameLift Servers es un complemento repleto de características para motores de juegos Unreal o Unity. Describe los pasos básicos necesarios para preparar el juego para el alojamiento en Amazon GameLift Servers. Con el conjunto de herramientas y los flujos de trabajo del complemento, puede trabajar dentro del entorno de desarrollo de los motores de juego a fin de preparar un servidor de juegos para el alojamiento, configurar el alojamiento en un equipo local para realizar pruebas, crear un servicio de backend sencillo e implementar un servidor de juegos en un alojamiento basado en la nube y administrado.

Utilice el complemento para trabajar con Amazon GameLift Servers y disponer rápidamente de una solución de alojamiento de juegos. Puede trabajar con ejemplos de activos de juego o con su propio proyecto de juego. El complemento automatiza una serie de pasos para que pueda crear con rapidez una solución funcional sencilla. Cuando complete los flujos de trabajo guiados del complemento, podrá conectar un cliente de juego a sesiones de juego alojadas en directo a través de Amazon GameLift Servers. Después de usar el complemento para crear una solución de alojamiento sencilla, podrá personalizar la solución para adaptarla a las necesidades de su juego.

El complemento está disponible para los siguientes motores de juego:

- Unreal Engine

- Unity

El complemento incluye los siguientes componentes para cada motor de juego:

- Módulos de complementos para el editor del motor de juegos. Cuando instale el complemento, un nuevo botón del menú principal le proporcionará acceso a la funcionalidad de Amazon GameLift Servers.
- Bibliotecas para la API del servicio de Amazon GameLift Servers con funcionalidad en el cliente.
- Bibliotecas para el SDK del servidor de Amazon GameLift Servers (versión 5).
- Ejemplos de activos para usarlos al probar la integración de un servidor.
- Configuraciones editables, en forma de CloudFormation plantillas, que definen su solución de servidor de juegos.

Temas

- [Flujo de trabajo del complemento](#)
- [Complemento de Amazon GameLift Servers para Unreal Engine](#)
- [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#)
- [Complemento de Amazon GameLift Servers para Unity \(SDK 4 del servidor\)](#)

Flujo de trabajo del complemento

Los siguientes pasos describen una ruta típica para preparar e implementar un proyecto de juego en Amazon GameLift Servers. Para completar estos pasos, debe trabajar en el editor de motores de juego y en el código del juego.

1. Cree un perfil de usuario que se vincule al usuario de su AWS cuenta y proporcione credenciales de acceso con permisos de usoAmazon GameLift Servers.
2. Configura AWS los recursos relacionados que el complemento utiliza en la solución de alojamiento (lo que se denomina «bootstrapping»).
3. Añada el código de servidor al proyecto para establecer la comunicación entre un servidor de juegos en ejecución y el servicio de Amazon GameLift Servers.
4. Añada un código de cliente a su proyecto que permita a los clientes del juego enviar solicitudes a Amazon GameLift Servers para iniciar nuevas sesiones de juego y, a continuación, conectarse a ellas.

5. Utilice el flujo de trabajo de Anywhere para configurar la estación de trabajo local como un recurso de computación de Anywhere y para alojar el servidor de juegos. Inicie el servidor y el cliente de juegos de forma local a través del complemento, conéctese a una sesión de juego y pruebe la integración.
6. Usa el EC2 flujo de trabajo administrado para cargar tu servidor de juegos Amazon GameLift Servers e implementar una solución de alojamiento en la nube simple pero completa. Inicie el cliente de juegos de forma local a través del complemento, solicite una sesión de juego, conéctese a ella y empiece a jugar.

Cuando trabajes en el plugin, crearás y utilizarás AWS recursos. Estas acciones pueden conllevar cargos a la AWS cuenta que utilices. Si es la primera vez que lo AWS usas, es posible que estas acciones estén incluidas en la [capa AWS gratuita](#).

Complemento de Amazon GameLift Servers para Unreal Engine

Este complemento agrega las herramientas y el SDK del servidor de C++ de Amazon GameLift Servers al editor de UE. Utilice los flujos de trabajo guiados de la interfaz de usuario para integrar la funcionalidad del SDK del servidor en su proyecto de juego e implementar una solución de alojamiento de Amazon GameLift Servers para su servidor de juegos.

Con el complemento, puede crear rápidamente una solución de alojamiento básica y, a continuación, optimizarla y personalizarla según sea necesario. Configure una flota de Amazon GameLift Servers Anywhere con su estación de trabajo local como host. Para el alojamiento en la nube con flotas de EC2 administradas o flotas de contenedores administradas, implemente el servidor de juegos con una solución completa para administrar las solicitudes de sesiones de juego y las conexiones de los clientes.

Temas

- [Instalación del complemento para su proyecto de juego de Unreal](#)
- [Próximos pasos: Personalizar la solución de alojamiento de juegos](#)
- [Plugin para Unreal: configura un perfil AWS de usuario](#)
- [Complemento para Unreal: integración del código del juego](#)
- [Complemento para Unreal: alojamiento local del juego con Amazon GameLift Servers Anywhere](#)
- [Complemento para Unreal: implementación de un juego en flotas de EC2 administradas](#)
- [Complemento para Unreal: implementación del juego en una flota de contenedores administrada](#)

Instalación del complemento para su proyecto de juego de Unreal

[Obtén el Amazon GameLift Servers plugin para Unreal Engine en GitHub](#)

Consulta el GitHub repositorio readme para obtener información sobre cómo instalar el complemento en tu Unreal Editor para un proyecto de juego.

El complemento incluye los siguientes componentes:

- Módulos de complementos para el editor de UE. Cuando instale el complemento, un nuevo botón del menú principal le proporcionará acceso a la funcionalidad de Amazon GameLift Servers.
- Bibliotecas de C++ para la API de servicio de Amazon GameLift Servers. Usa la funcionalidad de la API en un servicio de backend del lado del cliente para ayudar a los clientes del juego a solicitar sesiones de juego e información sobre las sesiones de juego. send/retrieve
- Bibliotecas de Unreal para el SDK del servidor de Amazon GameLift Servers (versión 5). Utilice el SDK del servidor en el código del servidor de juegos para administrar la comunicación entre los procesos del servidor de juegos alojado y el servicio Amazon GameLift Servers.
- Contenido para realizar pruebas, que incluye un mapa del juego inicial y dos mapas de prueba con planos básicos y elementos de la interfaz de usuario para utilizarlos cuando se pruebe la integración de un servidor.
- Configuraciones editables, en forma de CloudFormation plantillas, que el complemento utiliza al implementar tu servidor de juegos como alojamiento.

Este complemento usa plantillas de AWS CloudFormation para implementar soluciones de alojamiento en escenarios de juego comunes. Use estas soluciones tal y como se proporcionan o personalícelas según sea necesario para los juegos.

Próximos pasos: Personalizar la solución de alojamiento de juegos

Utilizar los flujos de trabajo guiados del complemento es una buena forma de empezar a utilizar rápidamente una solución de alojamiento de Amazon GameLift Servers. Con el complemento, puede configurar versiones básicas de cada uno de los componentes de la solución.

Cuando lo tenga todo listo, puede tomar esta solución como base, personalizar cada componente y refinar su solución para preparar el lanzamiento del juego. Considere estas opciones:

- Modifique las flotas y configuraciones de flota. Consulte [Personalizaciones de recursos de alojamiento](#).

- Personalice la configuración de la cola de sesiones de juego. Consulte [Personalización de colas de sesiones de juego](#):
- Añada funcionalidad al servidor y el cliente de juego. Consulte [Integración de un servidor de juegos con Amazon GameLift Servers](#) y [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#).
- Personalice el servicio de backend. Consulte [Creación de un servicio de backend para Amazon GameLift Servers](#).
- Configure el escalado automático de la capacidad para satisfacer la demanda prevista de jugadores. Consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).
- Configure las herramientas de observabilidad del alojamiento, como los análisis y el registro. Consulte [Supervisión de Amazon GameLift Servers](#).
- Automatice la implementación mediante la [infraestructura como código \(IaC\)](#). Los flujos de trabajo guiados del complemento para soluciones administradas utilizan plantillas de AWS CloudFormation . Esto se puede personalizar según sea necesario. Consulte [Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation](#).

Temas

- [Plugin para Unreal: configura un perfil AWS de usuario](#)
- [Complemento para Unreal: integración del código del juego](#)
- [Complemento para Unreal: alojamiento local del juego con Amazon GameLift Servers Anywhere](#)
- [Complemento para Unreal: implementación de un juego en flotas de EC2 administradas](#)
- [Complemento para Unreal: implementación del juego en una flota de contenedores administrada](#)

Plugin para Unreal: configura un perfil AWS de usuario

Tras instalar el complemento, configure un perfil de usuario con una AWS cuenta válida. Puede mantener varios perfiles en el complemento, pero solo puede tener un perfil seleccionado a la vez. Siempre que trabaje con el complemento, seleccione el perfil que quiera utilizar. Cada página del flujo de trabajo muestra el perfil seleccionado.

El mantenimiento de varios perfiles le ofrece la posibilidad de cambiar entre diferentes implementaciones de alojamiento. Por ejemplo, puedes configurar perfiles que usen la misma AWS cuenta pero que se desplieguen en distintas AWS regiones. O bien, puede configurar perfiles con AWS cuentas o usuarios y conjuntos de permisos diferentes.

 Note

Si ha instalado la AWS CLI en su estación de trabajo y ya tiene un perfil configurado, el Amazon GameLift Servers complemento lo detectará y lo incluirá como un perfil existente. El complemento selecciona automáticamente cualquier perfil nombrado como [default]. Puede utilizar un perfil existente o crear uno nuevo.

Todos los perfiles deben estar inicializados para configurar algunos de los AWS recursos necesarios para el usuario de su cuenta.

Para administrar sus perfiles AWS

1. En la barra de herramientas principal del editor de Unreal, elija el menú de Amazon GameLift Servers y seleccione Credenciales de acceso a AWS . Esta acción abre el complemento de Amazon GameLift Servers a la página Configuración del perfil de usuario.
2. Utilice los botones para crear una AWS cuenta nueva o configurar un perfil de usuario para una AWS cuenta que ya tenga.
3. Si aún no tiene un perfil de usuario, se le solicitará que introduzca los detalles del perfil y cree uno nuevo. Proporcione la información siguiente:
 - Una AWS cuenta. Si crea una AWS cuenta nueva, utilice el enlace a la Consola de administración de AWS y siga las instrucciones. Consulte [Creación de una cuenta de AWS](#) para obtener más información.
 - Un AWS usuario con permisos de uso Amazon GameLift Servers y otros AWS servicios necesarios. Consulte [Configurar una cuenta AWS de usuario](#) para obtener instrucciones sobre cómo configurar un usuario AWS Identity and Access Management (IAM) con Amazon GameLift Servers permisos y acceso programático con credenciales de larga duración.
 - Credenciales de su usuario. AWS Estas credenciales constan de un identificador de clave de AWS acceso y una clave AWS secreta. Consulte [Obtención de claves de acceso](#) para obtener más información.
 - AWS región. Se trata de la ubicación geográfica en la que desea crear sus recursos de AWS de alojamiento. Durante el desarrollo, le recomendamos que utilice una región cercana a su ubicación física. Elija una región de la lista de [AWS regiones compatibles](#).
4. Si el complemento detecta perfiles existentes, muestra una lista de perfiles disponibles. Seleccione un perfil existente de la lista o elija Añadir otro perfil para crear uno nuevo.

Arranque de un perfil de usuario

Todos los perfiles deben estar arrancados para poder utilizarlos con el complemento de Amazon GameLift Servers. Con el arranque, se crea un bucket de Amazon S3 específico para el perfil. Se usa para almacenar configuraciones de proyectos, artefactos de compilación y otras dependencias. Los buckets no se comparten entre otros perfiles.

La puesta en marcha implica la creación de nuevos AWS recursos y puede implicar costes.

Para arrancar el perfil, tenga en cuenta la siguiente información:

1. En la página Credenciales de acceso a AWS , compruebe el estado de arranque del perfil de usuario que desee utilizar. Si el estado de arranque del perfil es “Inactivo” y no aparece ningún bucket de S3 en la lista, debe arrancar el perfil.
2. Seleccione el perfil de seguridad que desea utilizar y elija Arrancar perfil.
3. Espere a que el estado del arranque cambie a “Activo”. Este proceso puede tardar unos minutos.

Complemento para Unreal: integración del código del juego

Antes de poder implementar el servidor de juegos en una flota, debe realizar una serie de actualizaciones en el código del juego y empaquetar los componentes del juego para usarlos con el servicio Amazon GameLift Servers.

En este tema se explican los pasos para realizar una integración mínima. Para la integración del servidor, utilice el ejemplo de código proporcionado para actualizar el modo de juego de su proyecto.

- [Configuración de los destinos de compilación y las reglas de los módulos](#)
- [Actualización del código del servidor de juegos](#)
- [Integración del mapa de juego del cliente](#)
- [Empaquetado de los componentes del juego](#)

Configuración de los destinos de compilación y las reglas de los módulos

Modifique los archivos del proyecto del juego para generar correctamente los componentes de compilación para usarlos con Amazon GameLift Servers.

Para añadir destinos de compilación del cliente y el servidor, siga estos pasos:

1. Abra los archivos de código del proyecto de juego y localice el archivo `.../Games/[your application name]Source/[your application name]Target.cs`. Ejemplo: `.../Source/GameLiftUnrealAppTarget.cs`. Si utiliza Visual Studio, abra el archivo `.sln` del proyecto.
2. Copie este archivo para crear dos nuevos archivos de destino en el directorio `Source/`.
 - Destino del cliente: cambie el nombre del nuevo archivo a `[your application name]Client.Target.cs`. Edite el contenido para actualizar el nombre de clase y los valores de tipo de destino, como se muestra en el siguiente ejemplo de código:

```
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" );
    }
}
```

- Destino del servidor: cambie el nombre del nuevo archivo a `[your application name]Server.Target.cs`. Edite el contenido para actualizar el nombre de clase y los valores de tipo de destino, como se muestra en el siguiente ejemplo de código:

```
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. Regenera los archivos del proyecto. Si utiliza Visual Studio, puede hacer clic con el botón derecho en el archivo .uproject del proyecto de juego y seleccionar Generar archivos de proyecto de Visual Studio.

Para actualizar las reglas del módulo del proyecto de juego:

Actualice las reglas del módulo del proyecto de juego para que dependan del complemento.

1. Abra los archivos de código del proyecto de juego y localice el archivo .../Games/[*your application name*]Source/[*your application name*].Build.cs. Ejemplo: .../Source/GameLiftUnrealApp.Build.cs. Si utiliza Visual Studio, abra el archivo .sln del proyecto.
2. Localice la clase ModuleRules y actualícela como se ilustra en el siguiente código de ejemplo:

```

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. Tras crear los nuevos archivos de destino y modificar las reglas del módulo, recompile su proyecto de juego.

Actualización del código del servidor de juegos

Actualice el código del servidor de juegos para permitir la comunicación entre un proceso del servidor de juegos y el servicio Amazon GameLift Servers. El servidor de juegos debe poder responder a las solicitudes de Amazon GameLift Servers, por ejemplo, para iniciar y detener nuevas sesiones de juego.

Para añadir el código de servidor para Amazon GameLift Servers

1. En el editor de código, abra el archivo de la solución (.sln) del proyecto de juego, que normalmente se encuentra en la carpeta raíz del proyecto. Por ejemplo: `GameLiftUnrealApp.sln`.
2. Con la solución abierta, busque el archivo de cabecera del modo de juego del proyecto: archivo `[project-name]GameMode.h`. Por ejemplo: `GameLiftUnrealAppGameMode.h`.
3. Cambie el archivo de encabezado para alinearlos con el siguiente código. Asegúrese de sustituir "GameLiftServer" por el nombre de su propio proyecto. Estas actualizaciones son específicas del servidor de juegos; le recomendamos que realice una copia de seguridad de los archivos originales del modo de juego para utilizarla con el cliente.

Código gameMode.h de ejemplo

```
// 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;
};
```

- Abra el archivo [project-name]GameMode.cpp del archivo de origen relacionado (por ejemplo, GameLiftUnrealAppGameMode.cpp). Cambie el código para alinearlo con el siguiente código de ejemplo. Asegúrese de reemplazar «GameLiftUnrealApp» por el nombre de su propio proyecto. Estas actualizaciones son específicas del servidor de juegos; le recomendamos que realice una copia de seguridad del archivo original para utilizarla con el cliente.

El siguiente código de ejemplo muestra cómo añadir los elementos mínimos necesarios para la integración del servidor con Amazon GameLift Servers:

- Inicializar un cliente de API de Amazon GameLift Servers. La llamada InitSDK() con los parámetros del servidor es obligatoria en el caso de las flotas de Amazon GameLift Servers Anywhere. Cuando se conecta a una flota de Anywhere, el complemento almacena los parámetros del servidor como argumentos de la consola. El código de ejemplo puede acceder a los valores en tiempo de ejecución.
- Implemente las funciones de devolución de llamada necesarias para responder a las solicitudes del servicio Amazon GameLift Servers, incluidas OnStartGameSession, OnProcessTerminate y onHealthCheck.
- Llame a ProcessReady() con un puerto designado para informar al servicio de Amazon GameLift Servers cuando el proceso esté listo para alojar sesiones de juego.

Código de servidor de juegos de ejemplo

```
// 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();

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
```

```

{
#if 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  
}
```

Integración del mapa de juego del cliente

El mapa inicial del juego contiene una lógica de esquema y elementos de interfaz de usuario que ya incluyen un código básico para solicitar sesiones de juego y utilizar la información de la conexión para conectarse a una sesión de juego. Puede utilizar el mapa tal como está o modificarlo según sea necesario. Utilice el mapa inicial del juego con otros recursos del juego, como el proyecto de plantilla Tercera persona proporcionado por Unreal Engine. Estos recursos están disponibles en el navegador de contenido. Puede utilizarlos para probar los flujos de trabajo de implementación del complemento, o como guía para crear un servicio de backend personalizado para el juego.

El mapa inicial tiene las siguientes características:

- Incluye la lógica tanto para una flota de Anywhere como para una flota de EC2 administrada. Al ejecutar el cliente, puede elegir conectarse a cualquiera de las dos flotas.
- La funcionalidad del cliente incluye buscar una sesión de juego (`SearchGameSessions()`), crear una nueva sesión de juego (`CreateGameSession()`) y unirse a una sesión de juego directamente.
- Obtiene un ID de jugador único del grupo de usuarios de Amazon Cognito del proyecto (forma parte de una solución de Anywhere implementada).

Uso del mapa inicial del juego

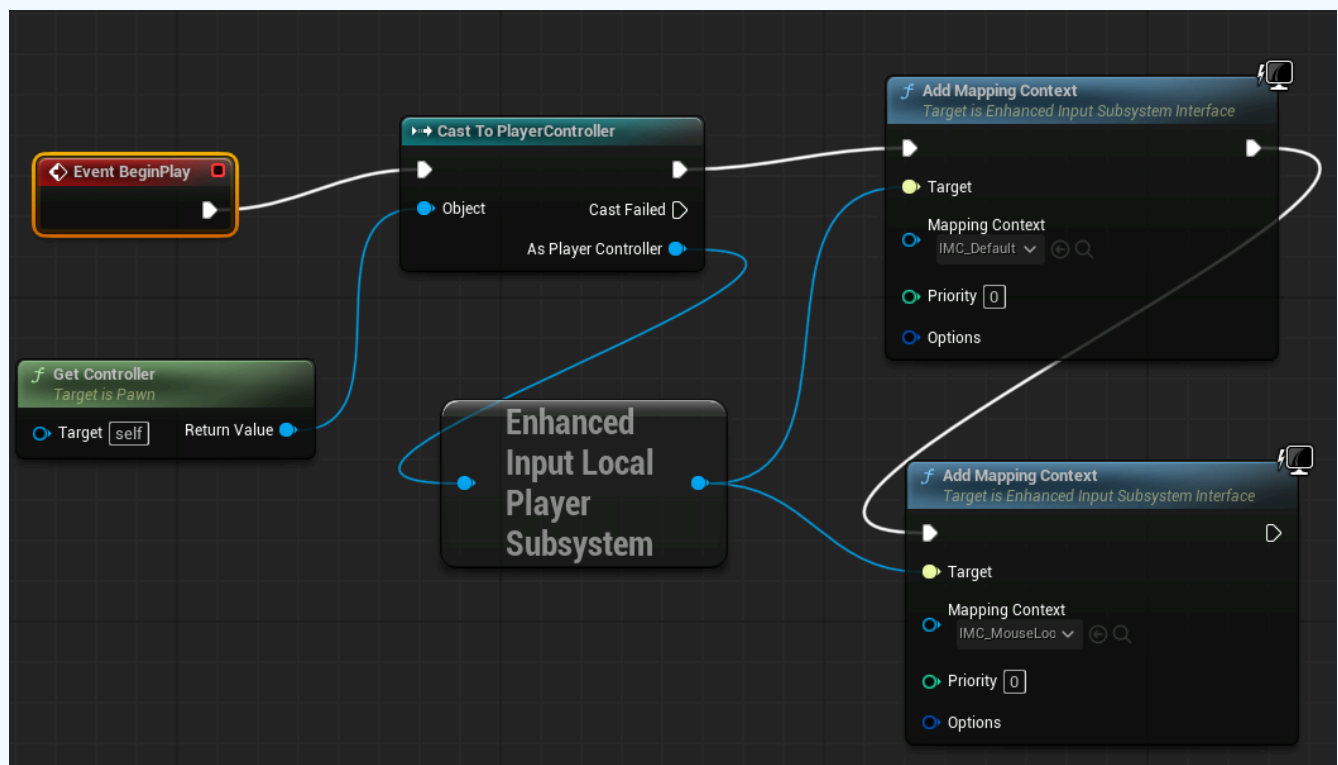
1. En el editor de UE, abra la página Configuración, mapas y modos del proyecto y expanda la sección Mapas predeterminados.
2. En Editor Startup Map, selecciona StartupMap "» en la lista desplegable. Puede que necesite buscar el archivo, que se encuentra en ... > Unreal Projects/[project-name]/Plugins/Amazon GameLift Servers Plugin Content/Maps.
3. En el mapa predeterminado del juego, selecciona el mismo "StartupMap" en la lista desplegable.
4. Para el mapa predeterminado del servidor, selecciona «Nivel 1_ThirdPerson» para Unreal Engine 5.6 o posterior, o «ThirdPersonMap» para versiones anteriores. Este es un mapa predeterminado incluido en el proyecto de juego. Este mapa está diseñado para dos jugadores del juego.

5. Abra el panel de detalles del mapa predeterminado del servidor. Establece la opción GameMode Anular como «Ninguna».
6. Expanda la sección Modos predeterminados y establezca Modo de juego del servidor predeterminado global en el modo de juego que actualizó para la integración del servidor.

Tras hacer esos cambios en el proyecto, estará listo para crear los componentes del juego.

Note

En el caso de Unreal Engine 5.6 o versiones posteriores, si no puedes mover al personaje tras conectarte al servidor del juego, actualiza el ThirdPersonCharacter plano BP_ para añadir el contexto de mapeo de entrada y tal IMC_Default y IMC_MouseLook como se muestra a continuación:



Empaquetado de los componentes del juego

Empaquetado de las compilaciones del servidor y el cliente del juego

1. Abra el proyecto de juego en una versión original del editor de Unreal Engine.

2. Si utilizas Unreal Engine 5.6 o una versión posterior, ve a Edición, Configuración del proyecto y Empaquetado. Busca Cook Everything en el directorio de contenido del proyecto y actívalo.
3. Utilice el editor para empaquetar las compilaciones de servidor y cliente de juego.
 - a. Elija un destino. Diríjase a Plataformas, Windows y seleccione una de las siguientes opciones:
 - Servidor: [your-application-name]Server
 - Cliente: [your-application-name]Client
 - b. Iniciar la compilación. Diríjase a Plataforma, Windows, Proyecto de paquete.

Cada proceso de empaquetado genera un archivo ejecutable: [your-application-name]Client.exe o [your-application-name]Server.exe.

En el complemento, establezca las rutas a los archivos ejecutables de compilación del cliente y del servidor en su estación de trabajo local.

Complemento para Unreal: alojamiento local del juego con Amazon GameLift Servers Anywhere

Utilice este flujo de trabajo para configurar su estación de trabajo local como host de servidor de juegos utilizando una flota de Anywhere. Puede usarlo para probar la integración del servidor de juegos antes de implementarlo en una flota administrada basada en la nube. También puede resultar útil para realizar pruebas locales durante el desarrollo iterativo del juego.

Para iniciar el flujo de trabajo de Amazon GameLift Servers Anywhere:

- En la barra de herramientas principal del editor de Unreal, elija el menú de Amazon GameLift Servers y seleccione Alojamiento con Anywhere. Con esta acción se abre la página del complemento Implementar Anywhere, que presenta un proceso de seis pasos para integrar, crear y lanzar los componentes del juego.

Paso 1: Configuración del perfil

Elija el perfil que desee utilizar al seguir este flujo de trabajo. El perfil que seleccione afectará a todos los pasos del flujo de trabajo. Todos los recursos que cree están asociados a la AWS cuenta del perfil y se colocan en la AWS región predeterminada del perfil. Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

Configuración de un perfil de usuario

1. Seleccione un perfil de la lista desplegable de perfiles disponibles. Si aún no tienes un perfil o quieres crear uno nuevo, ve al GameLift menú de Amazon y selecciona Establecer perfiles AWS de usuario.
2. Si el estado del arranque no es “Active”, seleccione Bootstrap profile y espere a que el estado cambie a “Active”.

Paso 2: Configuración del código de juego

En este paso, prepare compilaciones de servidor y cliente de juego que sean compatibles con Amazon GameLift Servers. Si todavía no ha integrado el código del juego, consulte [Complemento para Unreal: integración del código del juego](#). Introduzca las rutas a los archivos ejecutables del juego en su estación de trabajo local.

- Servidor de juegos: integre su servidor de juegos con el SDK del servidor para Amazon GameLift Servers y empaquete su versión de servidor de juegos. Para obtener instrucciones, consulte [Complemento para Unreal: integración del código del juego](#). El servidor de juegos debe estar integrado con el SDK del servidor para poder establecer comunicación con el servicio Amazon GameLift Servers y responder a las peticiones para iniciar nuevas sesiones de juego y aceptar conexiones con los clientes del juego.
- Cliente de juego: como mínimo, necesitará un cliente de juego que pueda conectarse a su servidor de juegos con la información de la dirección IP y el puerto. Si aún no has configurado los componentes del cliente del juego Amazon GameLift Servers, puedes usar la AWS CLI herramienta para solicitar nuevas sesiones de juego de forma manual, obtener información de conexión y usar esa información para conectar el cliente del juego.

En algún momento, necesitará disponer de un servicio de backend para enviar las solicitudes de nuevas sesiones de juego al servicio Amazon GameLift Servers y retransmitir la información de conexión al cliente del juego. Utilice los mapas de prueba incluidos con el complemento para añadir la funcionalidad de Amazon GameLift Servers de cliente a su proyecto de juego. Si necesita ayuda para compilar una solución personalizada, consulte [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#).

Paso 3: Conexión a una flota de Anywhere

En este paso, tendrá que designar la flota de Anywhere que desee utilizar. Una flota de Anywhere define un conjunto de recursos informáticos, que se pueden ubicar en cualquier lugar para el alojamiento de servidores de juegos.

- Si la AWS cuenta que utilizas actualmente tiene flotas de Anywhere, abre el campo desplegable del nombre de la flota y elige una flota. Este menú desplegable solo muestra las flotas de Anywhere de la AWS región para el perfil de usuario activo actualmente.
- Si no hay ninguna flota existente, o si desea crear una nueva, seleccione **Create new Anywhere fleet** e introduzca un nombre para la flota.

Una vez que haya elegido una flota de Anywhere para su proyecto, Amazon GameLift Servers verifica que el estado de la flota esté activo y muestra el ID de la flota. Puede hacer un seguimiento del progreso de esta solicitud en el registro de salida del editor de Unreal.

Paso 4: Registro de su estación de trabajo

En este paso, se registrará su estación de trabajo local como recurso informático en la nueva flota de Anywhere.

Registro de la estación de trabajo como recurso de computación de Anywhere

1. Especifique un nombre de equipo para la máquina local. Si añade más de un recurso informático a la flota, los nombres deben ser únicos.
2. Proporcione una dirección IP para su máquina local. El valor predeterminado de este campo es la dirección IP pública del equipo. También puede utilizar localhost (127.0.0.1), siempre que ejecute el cliente y el servidor de juegos en el mismo equipo.
3. Seleccione **Register compute**. Puede hacer un seguimiento del progreso de esta solicitud en el registro de salida del editor de Unreal.

Como respuesta a esta acción, Amazon GameLift Servers comprueba que se puede conectar al recurso informático y devuelve información sobre el recurso de computación recién registrado. También crea los argumentos de consola que los archivos ejecutables de los juegos necesitan para inicializar la comunicación con el servicio de Amazon GameLift Servers.

Paso 5: Generación del token de autenticación

Los procesos del servidor de juegos que se ejecutan en el recurso informático de Anywhere necesitan un token de autenticación para realizar llamadas al servicio Amazon GameLift Servers. El complemento genera y almacena automáticamente un token de autenticación para la flota de Anywhere cada vez que inicia el servidor de juegos desde el complemento. El valor del token de autenticación se almacena como un argumento de línea de comandos, que el código del servidor puede recuperar en tiempo de ejecución.

Los ejemplos de código proporcionados anteriormente también le permiten utilizar [AWS Signature Version 4 \(SigV4\) para solicitudes de API](#). SigV4 es el protocolo de AWS firma para añadir información de autenticación a las solicitudes de API.

No tiene que realizar ninguna acción en este paso.

Paso 6: Lanzamiento del juego

Ya ha completado todas las tareas necesarias para iniciar el juego multijugador en una estación de trabajo local mediante Amazon GameLift Servers.

Participación en el juego alojado

1. Lance el servidor de juegos. El servidor de juegos informará a Amazon GameLift Servers cuando esté listo para alojar sesiones de juego.
2. Lance el cliente de juegos y utilice la nueva funcionalidad para iniciar una nueva sesión de juego. Esta solicitud se envía a Amazon GameLift Servers a través del nuevo servicio de backend. Como respuesta, Amazon GameLift Servers llama al servidor de juegos, que se ejecuta en la máquina local, para iniciar una nueva sesión de juego. Cuando la sesión de juego esté lista para aceptar jugadores, Amazon GameLift Servers proporcionará la información de la conexión para que el cliente de juego se una a la sesión de juego.

Complemento para Unreal: implementación de un juego en flotas de EC2 administradas

En este flujo de trabajo, implemente el juego para el alojamiento en recursos de computación basados en la nube administrados por Amazon GameLift Servers. Cargue la compilación del servidor de juegos integrado al servicio Amazon GameLift Servers para su implementación. Si todavía no ha integrado el código del juego, consulte [Complemento para Unreal: integración del código del juego](#). Cuando se complete este flujo de trabajo, dispondrá de un cliente de juego en funcionamiento que podrá conectarse a los servidores de juegos en la nube.

Para iniciar el flujo de trabajo de Amazon EC2 administrado por Amazon GameLift Servers:

- En la barra de herramientas principal del editor de Unreal, elija el menú de Amazon GameLift Servers y seleccione Alojamiento con EC2 administrado. Con esta acción, se abrirá la página del complemento Implementar flota de Amazon EC2, que presenta un proceso de seis pasos para integrar, compilar y lanzar los componentes del juego.

Paso 1: Configuración del perfil

Elija el perfil que desee utilizar al seguir este flujo de trabajo. El perfil que seleccione afectará a todos los pasos del flujo de trabajo. Todos los recursos que cree están asociados a la AWS cuenta del perfil y se colocan en la AWS región predeterminada del perfil. Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

Configuración de un perfil de usuario

1. Seleccione un perfil de la lista desplegable de perfiles disponibles. Si aún no tienes un perfil o quieres crear uno nuevo, ve al GameLift menú de Amazon y selecciona Establecer perfiles AWS de usuario.
2. Si el estado del arranque no es “Active”, seleccione Bootstrap profile y espere a que el estado cambie a “Active”.

Paso 2: Configuración del código de juego

En este paso, prepare las compilaciones del servidor y del cliente del juego para que funcionen con el SDK del servidor C++ de Amazon GameLift Servers para Unreal. Si aún no ha integrado el código del juego ni ha creado los ejecutables del cliente y servidor del juego, consulte [Complemento para Unreal: integración del código del juego](#). Introduzca las rutas a los archivos ejecutables del juego en su estación de trabajo local.

En este paso del flujo de trabajo, el complemento proporciona enlaces a instrucciones y código fuente para configurar una versión compilada en origen del editor de Unreal. Debe utilizar la versión de la compilación de origen al compilar sus componentes de cliente y servidor.

Después de compilar un servidor de juegos que esté integrado con el SDK del servidor, realice las siguientes tareas a fin de prepararlo para cargarlo en Amazon GameLift Servers para el alojamiento.

Preparación de la compilación del servidor para la implementación en la nube (Windows)

En la carpeta `WindowsServer`, donde el editor de Unreal almacena los archivos de compilación del servidor de forma predeterminada, realice las adiciones siguientes:

1. Copie el script de instalación de la compilación del servidor en la raíz de la carpeta **WindowsServer**. El script de instalación se incluye en la descarga del complemento. Busque el archivo `[project-name]/Plugins/Resources/CloudFormation/extra_server_resources/install.bat`. Amazon GameLift Servers utiliza este archivo para instalar la compilación del servidor en los recursos de computación de alojamiento.
2. Copie el archivo **VC_redist.x64.exe** en el directorio raíz de la carpeta **WindowsServer**. Puede omitir este paso si utiliza la versión 5.6 o posterior de Unreal Engine. Este archivo se incluye en la instalación de Visual Studio. Por lo general, se encuentra en `C:/Program Files (x86)/Microsoft Visual Studio/2019/Professional/VC/Redist/MSVC/v142`.
3. Añada los archivos de la biblioteca de OpenSSL a la compilación del servidor de juegos. Puede saltarse este paso si el servidor de juegos está integrado con el SDK del servidor 5.3 o posterior. Esta versión se incluye en el complemento Amazon GameLift Servers para Unreal, versión 3.0 o posterior.

Busque y copie manualmente las bibliotecas OpenSSL en el directorio de paquetes de compilación del juego en `<YourGame>/Binaries/Win64`. Debe usar la misma versión de OpenSSL que usa su versión de Unreal Engine 5. Las compilaciones de juegos que se implementen con bibliotecas OpenSSL incorrectas no podrán comunicarse con el servicio Amazon GameLift Servers.

Busque las bibliotecas OpenSSL en el origen de su motor de juego. La ubicación varía en función del entorno de desarrollo:

En 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`

En 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

Preparación de la compilación del servidor para la implementación en la nube (Linux)

Para obtener instrucciones más detalladas sobre cómo preparar un servidor de juegos compilado para Linux, consulte [Cómo crear el SDK del servidor de Amazon GameLift Servers para Unreal Engine 5 en Amazon Linux](#).

1. Diseñe un directorio de trabajo para organizar sus archivos de compilación. La estructura del directorio de trabajo se implementa tal cual en cada recurso de computación de alojamiento. Añada su servidor de juegos compilado en Linux y todos los archivos dependientes.
2. Cree un script de instalación de la compilación del servidor en la raíz de su directorio de trabajo. Si es necesario, cree un archivo `install.sh` y añada los comandos necesarios para instalar correctamente la compilación del servidor de juegos. Amazon GameLift Servers utiliza este archivo para instalar la compilación del servidor en cada recurso de alojamiento de EC2.
3. Añada los archivos de la biblioteca OpenSSL a la compilación del servidor de juegos. Puede saltarse este paso si el servidor de juegos está integrado con el SDK del servidor 5.3 o posterior.

Localice y copie las bibliotecas manualmente. Debe usar la misma versión de OpenSSL que usa su versión de Unreal Engine 5. Las compilaciones de juegos que se implementen con bibliotecas OpenSSL incorrectas no podrán comunicarse con el servicio Amazon GameLift Servers.

- a. Busque las bibliotecas OpenSSL en el origen de su motor de juego. La ubicación varía en función del entorno de desarrollo:

En 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

En 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. Cuando encuentre las bibliotecas OpenSSL, cópielas en el directorio de paquetes de compilación del juego en <YourGame>/Binaries/Linux.

Paso 3: Selección del escenario de implementación

En este paso, tendrá que elegir la solución de alojamiento de juegos que desee implementar en ese momento. Puede disponer de varias implementaciones del juego mediante cualquiera de los escenarios.

- Flota de una sola región: despliega tu servidor de juegos en una sola flota de recursos de alojamiento en la región predeterminada AWS del perfil activo. Este escenario es un buen punto de partida para probar la integración del servidor con AWS y la configuración de compilación del servidor. Permite implementar los siguientes recursos:
 - La flota de AWS (bajo demanda) con la compilación del servidor de juegos instalada y en ejecución.
 - Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
 - Autorizador de pasarela de API que vincula el grupo de usuarios con. APIs
 - Web ACI para limitar las llamadas excesivas de los jugadores a la puerta de enlace API.
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten una ranura de juego. Esta función llama a `CreateGameSession()` si no hay ninguna disponible.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- FlexMatch flota: despliega tu servidor de juego en un conjunto de flotas y configura un FlexMatch emparejador con reglas para crear partidas de jugadores. En este escenario, se utiliza un alojamiento de spot de bajo costo con una estructura con múltiples flotas y con múltiples ubicaciones para garantizar la disponibilidad. Este método resultará útil cuando esté listo para empezar a diseñar un componente del emparejador para la solución de alojamiento. En este escenario, se crearán los recursos básicos para esta solución, que podrá personalizar después según sea necesario. Permite implementar los siguientes recursos:
 - FlexMatch Configuración y reglas de emparejamiento establecidas para aceptar las solicitudes de los jugadores y formar partidas.
 - Tres flotas de AWS con la compilación del servidor de juegos instalada y en ejecución en varios lugares. Incluye dos flotas de spot y una flota bajo demanda como respaldo.

- Cola de ubicación de sesión de juego de AWS que responde a las solicitudes de emparejamientos propuestos mediante la búsqueda del mejor recurso de alojamiento posible (en función de la viabilidad, el costo, la latencia de los jugadores, etc.) y el inicio de una sesión de juego.
- Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
- Autorizador de pasarela API que vincula el grupo de usuarios con. APIs
- Web API para limitar las llamadas excesivas de los jugadores a la puerta de enlace API.
- Puerta de enlace de la API + función de Lambda para que los jugadores soliciten una ranura de juego. Esta función llama a `StartMatchmaking()`.
- Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- Tablas de Amazon DynamoDB para almacenar tickets de emparejamiento para jugadores e información sobre las sesiones de juego.
- Tema de SNS más función Lambda para `GameSessionQueue` gestionar eventos.

Paso 4: Configuración de los parámetros del juego

En este paso, debes describir el juego para subirlo a; AWS

- Nombre de la compilación del servidor: proporcione un nombre significativo para la compilación del servidor de juegos. AWS utiliza ese nombre para hacer referencia a la copia de la compilación del servidor que se carga y se usa para las implementaciones.
- SO de compilación del servidor: especifique el sistema operativo para el que se ha diseñado el servidor. Esto indica a AWS qué tipo de recursos informáticos utilizar para alojar el juego.
- Carpeta del servidor de juegos: permite identificar la ruta a la carpeta de compilación del servidor local.
- Compilación del servidor de juegos: permite identificar la ruta al archivo ejecutable del servidor de juegos.
- Ruta del cliente de juego: permite identificar la ruta al archivo ejecutable del cliente de juego.
- Resultado de la configuración del cliente: este campo debe apuntar a una carpeta de la compilación del cliente que contenga la AWS configuración. Búsquelo en la siguiente ubicación: `[client-build]/[project-name]/Content/CloudFormation`.

Paso 5: Implementación del escenario

En este paso deberá implementar el juego en una solución de alojamiento en la nube en función del escenario de implementación que elija. Este proceso puede tardar varios minutos, mientras AWS valida la compilación del servidor, realiza un aprovisionamiento de los recursos de alojamiento, instala el servidor de juegos y lanza los procesos del servidor y los prepara para alojar sesiones de juego.

Para iniciar la implementación, elija Implementar CloudFormation. Puede realizar el seguimiento del estado del alojamiento de su juego aquí. Para obtener información más detallada, puede iniciar sesión en la consola AWS de administración AWS y ver las notificaciones de eventos. Asegúrate de iniciar sesión con la misma cuenta, usuario y AWS región que el perfil de usuario activo del complemento.

Cuando se complete la implementación, tendrá el servidor de juegos instalado en una instancia de EC2 de AWS. Hay al menos un proceso del servidor en ejecución y listo para iniciar una sesión de juego.

Paso 6: Lanzamiento del cliente

Ya ha completado todas las tareas necesarias para iniciar el juego multijugador alojado con Amazon GameLift Servers. Para jugar al juego, inicie una instancia del cliente de juego.

Si ha implementado el escenario de flota única, puede abrir una instancia de cliente única con un jugador, especificar el mapa del servidor y desplazarse. Abra instancias adicionales del cliente de juego para añadir un segundo jugador al mismo mapa de juego del servidor.

Si has implementado FlexMatch este escenario, la solución espera a que al menos dos clientes estén en cola para situarlos en la sesión de juego antes de que los jugadores puedan entrar en el mapa del servidor.

Complemento para Unreal: implementación del juego en una flota de contenedores administrada

Utilice este flujo de trabajo guiado del complemento para crear una imagen de contenedor para el servidor de juegos e implementarla en una solución de alojamiento basada en contenedores. Si todavía no ha integrado el código del juego, consulte [Complemento para Unreal: integración del código del juego](#). Cuando haya completado correctamente este flujo de trabajo, el servidor de juegos en contenedores se ejecutará en la nube, y podrá usar el complemento para iniciar un cliente de juego, conectarse a una sesión de juego y jugar.

Antes de comenzar

Este flujo de trabajo presupone que ha completado las siguientes tareas:

- Integre el código del servidor de juegos con el SDK del servidor de Amazon GameLift Servers. El servidor de juegos alojado debe poder comunicarse con el servicio Amazon GameLift Servers para poder responder a las solicitudes de inicio de nuevas sesiones de juego e informar sobre el estado de las sesiones de juego. Si no ha completado esta tarea, le recomendamos que siga primero el flujo de trabajo del complemento Alojamiento con Anywhere. Para obtener instrucciones sobre cómo preparar el código del servidor de juegos, consulte [Actualización del código del servidor de juegos](#). Para las flotas de contenedores administradas, debe integrar el juego con la versión 5.2 o superior del SDK del servidor.

Note

Si utiliza el mapa del juego inicial, esta tarea ya estará hecha.

- Empaquetado del archivo ejecutable del servidor de juegos para su ejecución en Linux. Si va a desarrollar en Windows e integrar la versión 5.2.x o anterior del SDK del servidor C++, tendrá que trabajar con el [kit de herramientas de compilación cruzada de Unreal](#). Como alternativa, puede configurar un espacio de trabajo de Linux independiente o utilizar una herramienta como el subsistema de Windows para Linux (WSL).
- Recopilación de los archivos para implementarlos con la compilación del servidor de juegos. En la máquina local, cree un directorio de trabajo para organizar los archivos que se incluirán en la imagen de contenedor del servidor de juegos. Estos pueden incluir las dependencias del juego, un script para lanzar los servidores de juegos y otros procesos al iniciar un contenedor, etc.
- Añada los archivos de la biblioteca OpenSSL para la compilación del servidor de juegos. Puede saltarse este paso si el servidor de juegos está integrado con el SDK del servidor 5.3 o posterior.

Busque y copie manualmente las bibliotecas OpenSSL en el directorio de paquetes de compilación del juego en <YourGame>/Binaries/Win64. Debe usar la misma versión de OpenSSL que usa su versión de Unreal Engine 5. Las compilaciones de juegos que se implementen con bibliotecas OpenSSL incorrectas no podrán comunicarse con el servicio Amazon GameLift Servers.

Para encontrar las bibliotecas OpenSSL, busque en el origen de su motor de juego. La ubicación varía en función del entorno de desarrollo:

En 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

En 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
- Integración del código de cliente de juego con Amazon GameLift Servers. Una forma de completar esta tarea consiste en añadir un recurso de muestra (incluido con el complemento) que ya esté integrado. Para obtener instrucciones sobre cómo preparar el código del cliente de juego, consulte [Integración del mapa de juego del cliente](#).
- Instalación de Docker en la máquina local. Necesita tener esta herramienta instalada si desea que el complemento cree imágenes de contenedor por usted y las envíe a un repositorio de ECR. Como alternativa, puede realizar estas tareas manualmente e indicar al complemento que utilice una imagen de contenedor existente. Para obtener más información sobre cómo crear la imagen manualmente, consulte [Creación de una imagen de contenedor para Amazon GameLift Servers](#).

Para iniciar el flujo de trabajo de contenedores administrados por Amazon GameLift Servers:

- En la barra de herramientas principal del editor de Unreal, elija el menú Amazon GameLift Servers y seleccione Contenedores administrados. Esta acción abre la página del plugin Host with Managed Containers, que presenta un step-by-step proceso para crear una imagen de contenedor con la versión del servidor del juego, desplegarla en una flota de contenedores y lanzar el juego.

Paso 0: Configura tu perfil

En esta sección se muestra el perfil de usuario seleccionado. Compruebe que el perfil de usuario actual es el que desea utilizar para este flujo de trabajo. Todos los recursos que cree en este flujo de trabajo se asocian a la AWS cuenta del perfil y se colocan en la AWS región predeterminada del perfil. Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

Es posible que tenga que modificar el perfil de usuario seleccionado si:

- Actualmente no hay ningún perfil seleccionado.

- Desea seleccionar un perfil diferente o crear uno nuevo.
- Debe arrancar el perfil seleccionado (si el estado de arranque está inactivo).

Configuración o cambio del perfil de usuario seleccionado

- En el Amazon GameLift Servers menú, elija Credenciales de AWS acceso abierto.

Paso 1: Evaluar la preparación del contenedor

Antes de implementar el servidor de juegos en una flota de contenedores, debe empaquetarlo en una imagen de contenedor y almacenarlo en un repositorio de Amazon ECR. El complemento puede realizar estas tareas por usted, o puede hacerlas usted manualmente. En este paso se proporciona información sobre el estado de la imagen del contenedor y el repositorio de ECR.

Utilice las preguntas de evaluación para indicar al complemento qué pasos debe seguir:

- Creación de una nueva imagen de contenedor. Si elige esta opción, el siguiente paso le pedirá la ubicación del directorio de compilación del servidor de juegos y del ejecutable de la compilación. El complemento utiliza una plantilla de Dockerfile (proporcionada por Amazon GameLift Servers) y la configura automáticamente para el juego. Puede ver la plantilla en [Creación de una imagen de contenedor para Amazon GameLift Servers](#). Tras elegir esta opción, indique dónde desea que el complemento almacene la nueva imagen:
 - Cree un nuevo repositorio de Amazon ECR y aplíquelo la imagen de contenedor. El complemento crea un repositorio de ECR privado utilizando la AWS cuenta y está predeterminado Región de AWS en el perfil de usuario seleccionado.
 - Aplique la imagen de contenedor a un repositorio de Amazon ECR creado previamente. Si elige esta opción, el siguiente paso le pedirá que seleccione un repositorio de Amazon ECR existente de una lista. La lista incluye todos los repositorios de Amazon ECR de la AWS cuenta y se incluyen de forma predeterminada Región de AWS en el perfil de usuario seleccionado. Puede seleccionar un repositorio público o privado.
- Uso de una imagen de contenedor existente. Si ha creado una imagen manualmente, le recomendamos que utilice la plantilla de Dockerfile proporcionada por Amazon GameLift Servers, que está disponible en [Creación de una imagen de contenedor para Amazon GameLift Servers](#). Tras seleccionar esta opción, indique dónde se encuentra la imagen.
 - Imagen generada por Docker y almacenada localmente. Si elige esta opción, el complemento crea un nuevo repositorio privado de Amazon ECR y le aplica el archivo de imagen local. En

el paso siguiente se le pedirá un ID de imagen, que el complemento utilizará para localizar el archivo de imagen.

- Imagen de contenedor que ya está almacenada en un repositorio de Amazon ECR. Si elige esta opción, el siguiente paso le pedirá que seleccione un repositorio de Amazon ECR existente y una imagen de una lista. La lista incluye todos los repositorios de Amazon ECR de la AWS cuenta y se incluyen de forma predeterminada Región de AWS en el perfil de usuario seleccionado. Puede seleccionar un repositorio público o privado.

Paso 2: Configurar la implementación de imágenes

En este paso, proporcione la información que el complemento necesita para implementar la imagen de contenedor en una flota de contenedores. En este paso se le solicitará la siguiente información:

- La ubicación de la compilación del servidor de juegos, la imagen del contenedor o el repositorio de Amazon ECR, según lo que seleccione en el paso 1.
- El escenario que se utilizará para la implementación de sus contenedores administrados.
- La ruta de salida de la configuración del cliente. Seleccione la carpeta de la compilación de su cliente que contiene su AWS configuración. Búsquelo en la siguiente ubicación: `[client-build]/[project-name]/Content/CloudFormation`.
- Configuración de implementación opcional. Esta sección contiene los ajustes de configuración que utiliza el complemento de forma predeterminada. Puede modificar los valores predeterminados o conservarlos.
 - El nombre del juego se establece en el nombre del proyecto de juego de forma predeterminada. Todos los recursos de AWS que crea el complemento hacen referencia al valor del nombre del juego.
 - El rango de puertos, el límite de memoria y el límite de vCPU son ajustes de configuración de la flota de contenedores. Para obtener más información sobre la personalización de estos valores, consulte [Configuración de conexiones de red](#) para conocer el rango de puertos de conexión y los límites de recursos de [Establecimiento de los límites de recursos](#).
- La etiqueta de imagen de contenedor se usa para clasificar las imágenes del contenedor en Amazon ECR. El valor predeterminado es `unreal-gamelift-plugin`.
- Nombre del repositorio de Amazon ECR. Puede editar este campo para sugerir un nombre personalizado solo si el complemento va a crear un repositorio de ECR para usted. El valor predeterminado es `unreal-game lift-plugin-ecr-repository`.

Opciones de escenario de implementación

Flota de contenedores de una sola región

En este escenario, el servidor de juegos se implementa en una sola flota de contenedores. Es un buen punto de partida para probar la integración del servidor con AWS y la configuración del contenedor. Permite implementar los siguientes recursos:

- La definición del grupo de contenedores de Amazon GameLift Servers describe cómo implementar y ejecutar las imágenes de contenedor en una flota de contenedores.
- La flota de contenedores de Amazon GameLift Servers (bajo demanda) con el contenedor del servidor de juegos instalado y en ejecución, con un alias.
- Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
- Autorizador de API Gateway que vincula el grupo de usuarios con APIs.
- Lista de control de acceso (ACL) web para limitar las llamadas excesivas de los jugadores a la puerta de enlace de la API.
- Servicio de backend para realizar solicitudes al servicio Amazon GameLift Servers en nombre de los clientes de juego, por ejemplo, para solicitar sesiones de juego y unirse al juego:
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten un espacio de sesión de juego. Esta función llama a `CreateGameSession()` si no hay espacios abiertos disponibles.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.

Flota de contenedores de una sola región con FlexMatch

Este escenario implementa el servidor de juegos en una flota de contenedores, configura la ubicación de las sesiones de juego y configura el emparejamiento de FlexMatch. Este escenario resultará útil cuando esté listo para empezar a diseñar un emparejador personalizado para su solución de alojamiento. Utilice este escenario para crear los recursos básicos para la solución, y que podrá personalizar más tarde según sea necesario. Permite implementar los siguientes recursos:

- La definición del grupo de contenedores de Amazon GameLift Servers que describe cómo implementar y ejecutar las imágenes de contenedor en una flota de contenedores.
- La flota de contenedores de Amazon GameLift Servers (bajo demanda) con el contenedor del servidor de juegos instalado y en ejecución, con un alias.

- Configuración de emparejamiento de FlexMatch y reglas de emparejamiento establecidas para aceptar las solicitudes de los jugadores y formar emparejamientos.
- Cola de sesiones de juego de Amazon GameLift Servers que responde a las solicitudes de emparejamiento propuesto mediante la búsqueda del mejor recurso de alojamiento posible (en función de la viabilidad, el costo, la latencia de los jugadores, etc.) y el inicio de una sesión de juego.
- Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
- Autorizador de API Gateway que vincula el grupo de usuarios con APIs.
- Lista de control de acceso (ACL) web para limitar las llamadas excesivas de los jugadores a la puerta de enlace de la API.
- Servicio de backend para realizar solicitudes al servicio Amazon GameLift Servers en nombre de los clientes de juego, por ejemplo, para solicitar sesiones de juego y unirse al juego:
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten un espacio de sesión de juego. Esta función llama a `StartMatchmaking()` si no hay espacios abiertos disponibles.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- Tablas de DynamoDB para almacenar tickets de emparejamiento para jugadores e información sobre las sesiones de juego.
- Tema de Amazon SNS más función Lambda para gestionar eventos. `GameSessionQueue`

Implementación de una flota de contenedores

Cuando la configuración de la flota esté completa, seleccione el botón Implementar flota de contenedores para iniciar la implementación. Este proceso puede tardar varios minutos mientras el complemento crea una imagen de contenedor y la envía a ECR, aprovisiona los recursos de alojamiento para la flota de contenedores e implementa la flota y otros recursos de AWS para el escenario de solución de alojamiento seleccionado.

Al iniciar una implementación, puede seguir el progreso de cada paso. Dependiendo de la configuración, los pasos pueden incluir lo siguiente:

- Configurar la imagen de contenedor
- Creación de un nuevo repositorio de Amazon ECR

- Crear una imagen e insertarla en Amazon ECR
- Crear la definición del grupo de contenedores
- Crear la flota de contenedores

Para obtener información más detallada sobre la implementación, seleccione Ver en la consola de administración de AWS . Una vez que la flota de contenedores alcance el estado activo, la flota estará ejecutando activamente los contenedores con los procesos del servidor que estén listos para alojar sesiones de juego.

Una vez que se complete la implementación, dispondrá de una flota de contenedores activa que estará lista para alojar sesiones de juego y aceptar conexiones de jugadores.

No es posible detener una implementación en curso. Si la implementación pasa a un estado defectuoso o falla, puede volver a empezar con la opción Reiniciar implementación.

Lanzamiento del cliente

En este punto, ya ha completado todas las tareas necesarias para lanzar e iniciar el juego multijugador alojado con Amazon GameLift Servers. Para jugar, elija Iniciar cliente para lanzar una instancia local del cliente de juego.

- Si ha implementado el escenario de flota única, puede abrir una instancia de cliente única con un jugador y especificar el mapa del servidor para desplazarse. Puede abrir una segunda instancia del cliente de juego para añadir un segundo jugador al mismo mapa de juego del servidor.
- Si ha implementado el escenario de FlexMatch, la solución de alojamiento espera a que al menos dos clientes del juego envíen solicitudes de emparejamiento. Abra al menos dos instancias del cliente de juego con un jugador. Se emparejará a los dos jugadores y se les pedirá que se unan a una sesión de juego para la partida.

Actualización de una flota de contenedores

Si ha implementado correctamente una solución de alojamiento de contenedores administrados, puede utilizar la característica Actualizar implementación. Esta opción le permite actualizar los ajustes de configuración de una flota de contenedores implementada sin necesidad de crear una nueva flota.

Al actualizar una implementación, puede implementar una imagen de contenedor con una compilación de servidor de juegos diferente, cambiar el repositorio de Amazon ECR, elegir un escenario de implementación diferente y personalizar los ajustes de configuración opcionales.

Cuando lo tenga todo listo para implementar los cambios, elija Actualizar. El tiempo necesario para una actualización de la implementación es similar al de una implementación completa. Para obtener información detallada sobre la implementación, seleccione Ver en la consola AWS de administración.

Limpie los recursos desplegados.

Como práctica recomendada, limpie los AWS recursos de su solución de contenedores gestionados tan pronto como ya no los necesite. Podría seguir incurriendo en costos por estos recursos si no los elimina.

Elimine los siguientes recursos:

- Pila de recursos de contenedores administrados. Los recursos de esta pila dependen del escenario de implementación que haya seleccionado. Para eliminar toda la pila, usa la CloudFormation consola. Las pilas que se generan a partir del complemento Amazon GameLift Servers utilizan la siguiente convención de nomenclatura: `GameLiftPluginForUnreal-{GameName}-Containers`. Espere a que se complete el proceso de eliminación de la pila antes de iniciar una nueva implementación de contenedores administrados en el complemento. Para obtener más información, consulte [Eliminar una pila de la CloudFormation consola](#).
- Repositorio de Amazon ECR. Si utilizó el complemento para crear un repositorio para la imagen de contenedor, quizás le interese eliminar los repositorios que ya no son necesarios. No es necesario eliminar un repositorio antes de restablecer una implementación de contenedores administrados. Si actualiza o restablece una implementación, el complemento usará automáticamente el mismo repositorio, a menos que se le indique usar otro. Para obtener más información, consulte [Eliminación de un repositorio privado en Amazon ECR](#).

Complemento de Amazon GameLift Servers para Unity (SDK 5.x del servidor)

Este complemento agrega las herramientas y el SDK del servidor de C# para Amazon GameLift Servers al editor de Unity. Utilice los flujos de trabajo guiados de la interfaz de usuario para integrar la funcionalidad del SDK del servidor en su proyecto de juego e implementar una solución de alojamiento de Amazon GameLift Servers para su servidor de juegos.

Con el complemento, puede crear rápidamente una solución de alojamiento básica y, a continuación, optimizarla y personalizarla según sea necesario. Configure una flota de Amazon GameLift

Servers Anywhere con su estación de trabajo local como host. Para el alojamiento en la nube con flotas de EC2 administradas o flotas de contenedores administradas, implemente el servidor de juegos con una solución completa para administrar las solicitudes de sesiones de juego y las conexiones de los clientes.

Temas

- [Instalación del complemento para su proyecto de juego de Unity](#)
- [Plugin para Unity: configura un perfil AWS de usuario](#)
- [Complemento para Unity: configuración de pruebas locales con Amazon GameLift Servers Anywhere](#)
- [Complemento para Unity: implementación del juego en una flota de EC2 administrada](#)
- [Complemento para Unity: implementación del juego en una flota de contenedores administrada](#)

Instalación del complemento para su proyecto de juego de Unity

[Obtén el Amazon GameLift Servers complemento para Unity en GitHub](#)

Consulta el GitHub repositorio readme para obtener información sobre cómo instalar el complemento para un proyecto de juego.

El complemento incluye los siguientes componentes:

- Módulos de complementos para el editor de Unity. Cuando instale el complemento, una nueva opción del menú principal le proporcionará acceso a la funcionalidad de Amazon GameLift Servers.
- Bibliotecas de C# para la API del servicio de Amazon GameLift Servers con funcionalidad en el cliente.
- Bibliotecas de C# para el SDK del servidor de Amazon GameLift Servers (versión 5.x).
- Pruebe el contenido del juego, como los activos y las escenas; así podrá probar Amazon GameLift Servers aunque no tenga un juego multijugador listo para compilar.
- Configuraciones de solución, proporcionadas como CloudFormation plantillas, que el complemento utiliza al implementar el servidor de juegos en la nube como alojamiento.

Este complemento usa plantillas de AWS CloudFormation para implementar soluciones de alojamiento en escenarios de juego comunes. Use estas soluciones tal y como se proporcionan o personalícelas según sea necesario para los juegos.

Plugin para Unity: configura un perfil AWS de usuario

Tras instalar el complemento, configure un perfil de usuario con una AWS cuenta válida. Puede mantener varios perfiles en el complemento, pero solo puede tener un perfil seleccionado a la vez. Siempre que trabaje con el complemento, seleccione el perfil que quiera utilizar. Cada página del flujo de trabajo muestra el perfil seleccionado.

El mantenimiento de varios perfiles le ofrece la posibilidad de cambiar entre diferentes implementaciones de alojamiento. Por ejemplo, puedes configurar perfiles que usen la misma AWS cuenta pero que se desplieguen en distintas AWS regiones. O bien, puede configurar perfiles con AWS cuentas o usuarios y conjuntos de permisos diferentes.

Note

Si ha instalado la AWS CLI en su estación de trabajo y ya tiene un perfil configurado, el Amazon GameLift Servers complemento lo detectará y lo incluirá como un perfil existente. El complemento selecciona automáticamente cualquier perfil nombrado como `[default]`. Puede utilizar un perfil existente o crear uno nuevo.

Todos los perfiles deben estar inicializados para configurar algunos de los AWS recursos necesarios para el usuario de su cuenta.

Para administrar sus perfiles AWS

1. En la barra de herramientas principal de Unity, elija el Amazon GameLift Servers menú y seleccione Credenciales de acceso a AWS . Esta acción abre el complemento de Amazon GameLift Servers a la página Configuración del perfil de usuario.
2. Utilice los botones para crear una AWS cuenta nueva o configurar un perfil de usuario para una AWS cuenta que ya tenga.
3. Si aún no tiene un perfil de usuario, se le solicitará que introduzca los detalles del perfil y cree uno nuevo. Proporcione la información siguiente:
 - Una AWS cuenta. Si crea una AWS cuenta nueva, utilice el enlace a la Consola de administración de AWS y siga las instrucciones. Consulte [Creación de una cuenta de AWS](#) para obtener más información.
 - Un AWS usuario con permisos de uso Amazon GameLift Servers y otros AWS servicios necesarios. Consulte [Configurar una cuenta AWS de usuario](#) para obtener instrucciones

sobre cómo configurar un usuario AWS Identity and Access Management (IAM) con Amazon GameLift Servers permisos y acceso programático con credenciales de larga duración.

- Credenciales de su usuario. AWS Estas credenciales constan de un identificador de clave de AWS acceso y una clave AWS secreta. Consulte [Obtención de claves de acceso](#) para obtener más información.
 - AWS región. Se trata de una ubicación geográfica en la que desea crear tus AWS recursos de alojamiento. Durante el desarrollo, le recomendamos que utilice una región cercana a su ubicación física. Elige una región de la lista de [AWS regiones compatibles](#).
4. Si el complemento detecta perfiles existentes, muestra una lista de perfiles disponibles. Seleccione un perfil existente de la lista o elija Añadir otro perfil para crear uno nuevo.

Arranque de un perfil de usuario

Todos los perfiles deben estar arrancados para poder utilizarlos con el complemento de Amazon GameLift Servers. Con el arranque, se crea un bucket de Amazon S3 específico para el perfil. Se usa para almacenar configuraciones de proyectos, artefactos de compilación y otras dependencias. Los buckets no se comparten entre otros perfiles.

La puesta en marcha implica la creación de nuevos AWS recursos y puede implicar costes.

Para arrancar el perfil, tenga en cuenta la siguiente información:

1. En la página Credenciales de acceso a AWS , compruebe el estado de arranque del perfil de usuario que desea utilizar. Si el estado de arranque del perfil es “Inactivo” y no aparece ningún bucket de S3 en la lista, debe arrancar el perfil.
2. Seleccione el perfil de seguridad que desea utilizar y elija Arrancar perfil.
3. Espere a que el estado del arranque cambie a “Activo”. Este proceso puede tardar unos minutos.

Complemento para Unity: configuración de pruebas locales con Amazon GameLift Servers Anywhere

En este flujo de trabajo, se añadirá el código de juego del cliente y del servidor para la funcionalidad de Amazon GameLift Servers y se utilizará el complemento para designar la estación de trabajo local como host del servidor de juegos de prueba. Cuando haya completado las tareas de integración, utilice el complemento para crear los componentes de cliente y servidor de juegos.

Para iniciar el flujo de trabajo de Amazon GameLift Servers Anywhere:

- En el menú principal del editor de Unity, elija Amazon GameLift Servers y seleccione Alojar con Anywhere. Esta acción abre la página del complemento para configurar el juego con una flota de @Anywhere. La página muestra un proceso de cinco pasos para integrar, compilar y lanzar los componentes del juego.

Configuración del perfil

Elija el perfil que desee utilizar al seguir este flujo de trabajo. El perfil que seleccione afectará a todos los pasos del flujo de trabajo. Todos los recursos que cree están asociados a la AWS cuenta del perfil y se colocan en la región predeterminada del perfil. AWS Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

1. Seleccione un perfil de la lista desplegable de perfiles disponibles. Si aún no tienes un perfil o quieres crear uno nuevo, ve al Amazon GameLift Servers menú y selecciona Establecer perfiles de AWS cuenta.
2. Si el estado del arranque no es “Activo”, seleccione Perfil de arranque y espere a que el estado cambie a “Activo”.

Integre el código del juego con el SDK del servidor de C#

Note

Si ha importado el juego de ejemplo, puede omitir este paso. Los activos del juego de ejemplo ya tienen el código de servidor y cliente necesario.

Para este paso del flujo de trabajo, debe actualizar el código del cliente y del servidor en el proyecto del juego.

- Los servidores de juegos deben poder comunicarse con el servicio de Amazon GameLift Servers a fin de recibir peticiones de inicio de sesión de juego, proporcionar datos de conexión de la sesión de juego e informar sobre el estado.
- Los clientes del juego deben poder obtener información sobre las sesiones de juego, iniciar una sesión o unirse a ella y obtener datos de conexión para entrar en una partida.

Integración del código del servidor

Si emplea su propio proyecto de juego con escenas personalizadas, utilice el código de ejemplo proporcionado para añadir el código de servidor necesario al proyecto del juego:

1. En los archivos del proyecto del juego, abra la carpeta `Assets/Scripts/Server`. Si el archivo no existe, créelo.
2. Ve al GitHub repositorio [aws/ amazon-gamelift-plugin-unity](#) y abre la ruta. `Samples~/SampleGame/Assets/Scripts/Server`
3. Busque el archivo `GameLiftServer.cs` y cópielo en la carpeta `Server` del proyecto del juego. Cuando compile un ejecutable de servidor, use este archivo como destino de la compilación.

El código de ejemplo incluye, como mínimo, estos elementos obligatorios, que utilizan el SDK del servidor de C# de Amazon GameLift Servers (versión 5):

- Inicializa un cliente de API de Amazon GameLift Servers. La llamada `InitSDK()` con los parámetros del servidor es obligatoria en el caso de las flotas de Amazon GameLift Servers Anywhere. Estos ajustes se configuran automáticamente para el uso en el complemento.
- Implementa las funciones de devolución de llamada necesarias para responder a las solicitudes del servicio Amazon GameLift Servers, incluidas `OnStartGameSession`, `OnProcessTerminate` y `onHealthCheck`.
- Llama a `ProcessReady()` con un puerto designado para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté listo para alojar sesiones de juego.

Si desea personalizar el código de servidor de ejemplo, consulte estos recursos:

- [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)

Integración del código del cliente

Si utiliza su propio proyecto de juego con escenas personalizadas, necesita integrar las funciones básicas en el cliente del juego. También debe añadir elementos de interfaz de usuario para que los jugadores puedan iniciar sesión y unirse a una sesión de juego. Usa la API de servicio Amazon GameLift Servers (en el AWS SDK) para obtener información sobre las sesiones de juego, crear nuevas sesiones de juego o unirse a las sesiones de juego existentes.

Al crear un cliente para pruebas locales con una flota de Anywhere, puede añadir llamadas directas al servicio Amazon GameLift Servers. Cuando desarrolle el juego para el alojamiento en la nube, o si planea usar flotas de Anywhere para el alojamiento de producción, necesitará crear un servicio de backend en el cliente a fin de gestionar toda la comunicación entre los clientes del juego y el servicio Amazon GameLift Servers.

Para integrar Amazon GameLift Servers en el código del cliente, use los siguientes recursos como guía.

- Integre el cliente con la `GameLiftCoreApi` clase en el GitHub repositorio `amazon-gamelift-plugin-unity aws/`. Esta clase proporciona controles para la autenticación de los jugadores y para recuperar información de la sesión de juego.
- Vea ejemplos de integraciones de juegos, disponibles en el repositorio `aws/`, GitHub `amazon-gamelift-plugin-unity Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- Siga las instrucciones para añadir Amazon GameLift Servers al cliente de juegos de Unity.

Los clientes de juego que se conecten a una flota de Anywhere necesitan la siguiente información. El complemento actualiza automáticamente el proyecto de juego para que use los recursos que haya creado en el complemento.

- `FleetId` - El identificador único de tu flota de Anywhere.
- `FleetLocation` - La ubicación personalizada de su flota de Anywhere.
- `AwsRegion` - La AWS región en la que está alojada tu flota de Anywhere. esta es la región que ha configurado en el perfil de usuario.
- `ProfileName` - Un perfil de AWS credenciales en su máquina local que le permita acceder al AWS SDK de Amazon GameLift Servers. El cliente del juego utiliza estas credenciales para autenticar las solicitudes al servicio Amazon GameLift Servers.

Note

El perfil con credenciales está generado por el complemento y se guarda en el equipo local. Por lo tanto, debe ejecutar el cliente en el equipo local (o en un equipo con el mismo perfil).

Conexión a una flota de Anywhere

En este paso, tendrá que designar la flota de Anywhere que desee utilizar. Una flota de Anywhere define un conjunto de recursos informáticos, que se pueden ubicar en cualquier lugar para el alojamiento de servidores de juegos.

- Si la AWS cuenta que utilizas actualmente tiene flotas de Anywhere existentes, abre el campo desplegable del nombre de la flota y elige una flota. Este menú desplegable solo muestra las flotas de Anywhere de la AWS región para el perfil de usuario activo actualmente.
- Si no hay ninguna flota existente, o si desea crear una nueva, seleccione Create new Anywhere fleet e introduzca un nombre para la flota.

Una vez que haya elegido una flota de Anywhere para su proyecto, Amazon GameLift Servers verifica que el estado de la flota esté activo y muestra el ID de la flota. Puede hacer un seguimiento del progreso de esta solicitud en el registro de salida del editor de Unity.

Registro de un recurso de computación

En este paso, se registrará su estación de trabajo local como recurso informático en la nueva flota de Anywhere.

1. Especifique un nombre de equipo para la máquina local. Si añade más de un recurso informático a la flota, los nombres deben ser únicos.
2. Seleccione Register compute. Puede hacer un seguimiento del progreso de esta solicitud en el registro de salida del editor de Unity.

El complemento registra la estación de trabajo local con la dirección IP establecida en localhost (127.0.0.1). Esta configuración da por sentado que ejecutará el cliente y el servidor del juego en el mismo equipo.

Como respuesta a esta acción, Amazon GameLift Servers comprueba que se puede conectar al recurso informático y devuelve información sobre el recurso de computación recién registrado.

Lanzamiento del juego

En este paso, compilará los componentes del juego y los lanzará para poder jugar. Realice los siguientes pasos:

1. Configure el cliente del juego. En este paso, le pide al complemento que actualice un activo `GameLiftClientSettings` para el proyecto del juego. El complemento utiliza este activo para almacenar cierta información que el cliente del juego necesita a fin de conectarse al servicio Amazon GameLift Servers.
 - a. Si no ha importado ni inicializado el juego de ejemplo, cree un nuevo activo `GameLiftClientSettings`. En el menú principal del editor de Unity, selecciona **Assets, Create, Amazon GameLift, Client Settings**. Si crea varias copias de `GameLiftClientSettings` en el proyecto, el complemento lo detectará automáticamente y le dirá qué recurso actualizará el complemento.
 - b. En **Launch Game**, seleccione **Configure Client: Apply Anywhere Settings**. Esta acción actualiza la configuración del cliente del juego para que use la flota de Anywhere que acaba de configurar.
2. Compile y ejecute el cliente del juego.
 - a. Compile un ejecutable del cliente mediante el proceso de compilación estándar de Unity. En **File, Build Settings**, cambie la plataforma a **Windows, Mac, Linux**. Si ha importado el juego de ejemplo y ha inicializado la configuración, la lista y el destino de la compilación se actualizarán automáticamente.
 - b. Lance una o más instancias del ejecutable del cliente de juego recién compilado.
3. Lance un servidor de juegos en la flota de Anywhere. Seleccione **Server: Launch Server** en el editor. Esta tarea inicia un servidor activo al que el cliente puede conectarse mientras el editor de Unity permanezca abierto.
4. Inicie una sesión de juego o únase a una sesión. En las instancias del cliente de juego, use la interfaz de usuario para unir a cada cliente a una sesión de juego. La forma de hacerlo dependerá de cómo haya añadido la funcionalidad en el cliente.

Si utiliza el cliente de juegos de ejemplo, tendrá las siguientes características:

- Un componente de inicio de sesión para jugadores. Al conectarse a un servidor de juegos en una flota de Anywhere, no hay validación de jugadores. Puede introducir cualquier valor para unirse a la sesión de juego.
- Una interfaz de usuario sencilla para unirse al juego. Cuando un cliente intenta unirse a una partida, busca automáticamente una sesión de juego activa con espacio disponible para un jugador. Si no hay ninguna sesión de juego disponible, el cliente solicita una nueva sesión de juego. Si hay alguna sesión de juego disponible, el cliente solicita unirse a esta. Al probar el juego

con varios clientes simultáneos, el primer cliente inicia la sesión de juego y los demás se unen automáticamente a la sesión de juego existente.

- Sesiones de juego con espacio para cuatro jugadores. Puede lanzar hasta cuatro instancias de clientes de juego simultáneas: se unirán todas a la misma sesión de juego.

Lanzamiento desde un ejecutable de servidor (opcional)

Puede compilar y lanzar el ejecutable del servidor de juegos para probarlo en una flota de Anywhere.

1. Compile un ejecutable del servidor mediante el proceso de compilación estándar de Unity. En File, Build Settings, cambie la plataforma a Dedicated Server y haga la compilación.
2. Obtenga un token de autenticación a corto plazo llamando al comando AWS CLI [get-compute-auth-token](#) con su ID de flota y AWS región de Anywhere. El ID de flota aparece en Connect to an Anywhere Fleet al crear la flota. La AWS región se muestra en Defina su perfil cuando selecciona su perfil activo.

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

3. Lance el ejecutable del servidor de juegos recién compilado desde una línea de comandos y pase un token de autenticación válido.

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

Complemento para Unity: implementación del juego en una flota de EC2 administrada

En este flujo de trabajo, utilizará el complemento a fin de preparar el juego para el alojamiento en recursos de computación basados en la nube que están administrados por Amazon GameLift Servers. Usted añade el código de juego de cliente y servidor para la funcionalidad de Amazon GameLift Servers y, a continuación, carga la compilación del servidor en el servicio Amazon GameLift Servers para el alojamiento. Cuando complete este flujo de trabajo, tendrá servidores de juego ejecutándose en la nube y un cliente de juegos funcional que se pueda conectar a ellos.

Para iniciar el flujo de trabajo de Amazon EC2 administrado por Amazon GameLift Servers:

- En el menú principal del editor de Unity, elija Amazon GameLift Servers y seleccione Alojar con EC2 administrado. Este flujo de trabajo presenta un proceso de seis pasos para integrar, compilar, implementar y lanzar los componentes del juego.

Configuración del perfil

Elija el perfil que desee utilizar al seguir este flujo de trabajo. El perfil que seleccione afectará a todos los pasos del flujo de trabajo. Todos los recursos que cree están asociados a la AWS cuenta del perfil y se colocan en la AWS región predeterminada del perfil. Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

1. Seleccione un perfil de la lista desplegable de perfiles disponibles. Si aún no tienes un perfil o quieres crear uno nuevo, ve al Amazon GameLift Servers menú y selecciona Establecer perfiles de AWS cuenta.
2. Si el estado del arranque no es “Activo”, seleccione Perfil de arranque y espere a que el estado cambie a “Activo”.

Integración del juego con Amazon GameLift Servers

Para esta tarea, debe actualizar el código del cliente y del servidor en el proyecto del juego.

- Los servidores de juegos deben poder comunicarse con el servicio de Amazon GameLift Servers a fin de recibir peticiones de inicio de sesión de juego, proporcionar datos de conexión de la sesión de juego e informar sobre el estado.
- Los clientes del juego deben poder obtener información sobre las sesiones de juego, iniciar una sesión o unirse a ella y obtener datos de conexión para entrar en una partida.

Note

Si ha importado el juego de ejemplo, puede omitir este paso. Los activos del juego de ejemplo ya tienen el código de servidor y cliente necesario.

Integración del código del servidor

Si emplea su propio proyecto de juego con escenas personalizadas, utilice el código de ejemplo proporcionado para añadir el código de servidor necesario al proyecto del juego. Si integró el proyecto del juego para probarlo con una flota de Anywhere, ya ha completado las instrucciones de este paso.

1. En los archivos del proyecto del juego, abra la carpeta Assets/Scripts/Server. Si el archivo no existe, créelo.

2. Ve al GitHub repositorio [aws/ amazon-gamelift-plugin-unity](https://github.com/aws/amazon-gamelift-plugin-unity) y abre la ruta. `Samples~/SampleGame/Assets/Scripts/Server`
3. Busque el archivo `GameLiftServer.cs` y cópielo en la carpeta `Server` del proyecto del juego. Cuando compile un ejecutable de servidor, use este archivo como destino de la compilación.

El código de ejemplo incluye, como mínimo, estos elementos obligatorios, que utilizan el SDK del servidor de C# de Amazon GameLift Servers (versión 5):

- Inicializa un cliente de API de Amazon GameLift Servers. La llamada `InitSDK()` con los parámetros del servidor es obligatoria en el caso de las flotas de Amazon GameLift Servers Anywhere. Estos ajustes se configuran automáticamente para el uso en el complemento.
- Implementa las funciones de devolución de llamada necesarias para responder a las solicitudes del servicio Amazon GameLift Servers, incluidas `OnStartGameSession`, `OnProcessTerminate` y `onHealthCheck`.
- Llama a `ProcessReady()` con un puerto designado para informar al servicio Amazon GameLift Servers cuando el proceso del servidor esté listo para alojar sesiones de juego.

Si desea personalizar el código de servidor de ejemplo, consulte estos recursos:

- [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)

Integración del código del cliente

Para los clientes de juego que se conectan a servidores de juego basados en la nube, es una práctica recomendada utilizar un servicio de backend en el cliente para hacer llamadas al servicio Amazon GameLift Servers, en lugar de hacer las llamadas directamente desde el cliente del juego.

En el flujo de trabajo del complemento para el alojamiento en una flota de EC2 administrada, cada escenario de implementación incluye un servicio de backend preconfigurado con los siguientes componentes:

- Un conjunto de funciones de Lambda y tablas de DynamoDB que se utilizan para solicitar sesiones de juego y recuperar información sobre las sesiones de juego. Estos componentes utilizan una puerta de enlace de la API como proxy.

- Un grupo de usuarios de Amazon Cognito que genera un reproductor único IDs y autentica las conexiones de los jugadores.

Para usar estos componentes, el cliente del juego necesita una funcionalidad que le permita enviar solicitudes al servicio de backend para hacer lo siguiente:

- Crea un usuario jugador en el grupo de usuarios de AWS Cognito y auténticate.
- Unirse a una sesión de juego y recibir información de conexión.
- Unirse a un juego mediante el emparejamiento.

Utilice los siguientes recursos como guía.

- [Integre el cliente con la `GameLiftCoreApiclas` en el GitHub repositorio `aws/.amazon-gamelift-plugin-unity`](#) Esta clase proporciona controles para la autenticación de los jugadores y para recuperar información de la sesión de juego.
- [Para ver los ejemplos de integraciones de juegos, ve al repositorio `aws/.GitHub amazon-gamelift-plugin-unity`](#) `Samples~/SampleGame/Assets/Scripts/Client/GameLiftClient.cs`
- [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers.](#)

(Solo Unity 6.3 y versiones posteriores) Integre el script `install.sh`

Unity 6.3 en Linux requiere glibc 2.35. Amazon Linux 2023 incluye glibc 2.34. Se proporciona un script `install.sh` que se encarga de compilar glibc 2.35 desde el código fuente y de configurar los binarios de Unity para usar glibc 2.35.

1. [Ve al repositorio `aws/` y abre la ruta `GitHub .amazon-gamelift-plugin-unity`](#)
`ExtraServerResources`
2. Localice el archivo `install.sh` y cópielo en la carpeta que contiene la compilación de su servidor.
3. Si ya tiene `uninstall.sh`, combine el contenido de ambos scripts en uno solo `install.sh`.

Selección del escenario de implementación

En este paso, tendrá que elegir la solución de alojamiento de juegos que desee implementar en ese momento. Puede disponer de varias implementaciones del juego mediante cualquiera de los escenarios.

- Flota de una sola región: despliega tu servidor de juegos en una sola flota de recursos de alojamiento en la región predeterminada AWS del perfil activo. Este escenario es un buen punto de partida para probar la integración del servidor con AWS la configuración de compilación del servidor. Permite implementar los siguientes recursos:
 - La flota de AWS (bajo demanda) con la compilación del servidor de juegos instalada y en ejecución.
 - Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
 - Autorizador de puerta de enlace de API que vincula el grupo de usuarios con APIs.
 - Web ACI para limitar las llamadas excesivas de los jugadores a la puerta de enlace API.
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten una ranura de juego. Esta función llama a `CreateGameSession()` si no hay ninguna disponible.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- FlexMatch flota: despliega tu servidor de juego en un conjunto de flotas y configura un FlexMatch emparejador con reglas para crear partidas de jugadores. En este escenario, se utiliza un alojamiento de spot de bajo costo con una estructura con múltiples flotas y con múltiples ubicaciones para garantizar la disponibilidad. Este método resultará útil cuando esté listo para empezar a diseñar un componente del emparejador para la solución de alojamiento. En este escenario, se crearán los recursos básicos para esta solución, que podrá personalizar después según sea necesario. Permite implementar los siguientes recursos:
 - FlexMatch Configuración y reglas de emparejamiento establecidas para aceptar las solicitudes de los jugadores y formar partidas.
 - Tres flotas de AWS con la compilación del servidor de juegos instalada y en ejecución en varios lugares. Incluye dos flotas de spot y una flota bajo demanda como respaldo.
 - Cola de ubicación de sesión de juego de AWS que responde a las solicitudes de emparejamientos propuestos mediante la búsqueda del mejor recurso de alojamiento posible (en función de la viabilidad, el costo, la latencia de los jugadores, etc.) y el inicio de una sesión de juego.
 - Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
 - Autorizador de pasarela API que vincula el grupo de usuarios con APIs
 - Web ACI para limitar las llamadas excesivas de los jugadores a la puerta de enlace API.

- Puerta de enlace de la API + función de Lambda para que los jugadores soliciten una ranura de juego. Esta función llama a `StartMatchmaking()`.
- Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- Tablas de Amazon DynamoDB para almacenar tickets de emparejamiento para jugadores e información sobre las sesiones de juego.
- Tema de SNS más función Lambda para `GameSessionQueue` gestionar eventos.

Parámetros del juego

En este paso, debes describir el juego en el que quieres subirlo. AWS

- Nombre del juego: proporcione un nombre descriptivo para el proyecto del juego; este nombre se usará en el complemento.
- Nombre de la flota: asigne un nombre significativo a la flota de EC2 administrada. Amazon GameLift Servers usa este nombre (junto con el ID de flota) al enumerar los recursos en la consola de AWS .
- Nombre de la compilación: proporciona un nombre significativo para la compilación de tu servidor. AWS usa este nombre para hacer referencia a la copia de la compilación del servidor que se carga Amazon GameLift Servers y se usa para las implementaciones.
- Parámetros de lanzamiento: introduzca las instrucciones opcionales que se ejecutarán al lanzar el ejecutable del servidor en una instancia de flota de EC2 administrada; la longitud máxima es de 1024 caracteres.
- Carpeta del servidor de juegos: proporcione la ruta a una carpeta local que contenga la compilación del servidor.
- Archivo del servidor de juegos: especifique el nombre del archivo ejecutable del servidor.

Escenario de implementación

En este paso deberá implementar el juego en una solución de alojamiento en la nube en función del escenario de implementación que elija. Este proceso puede tardar varios minutos, mientras AWS valida la compilación del servidor, realiza un aprovisionamiento de los recursos de alojamiento, instala el servidor de juegos y lanza los procesos del servidor y los prepara para alojar sesiones de juego.

Para iniciar la implementación, elija Implementar CloudFormation. Puede realizar el seguimiento del estado del alojamiento de su juego aquí. Para obtener información más detallada, puede iniciar sesión en la consola AWS de administración AWS y ver las notificaciones de eventos. Asegúrate de iniciar sesión con la misma cuenta, usuario y AWS región que el perfil de usuario activo del complemento.

Cuando se complete la implementación, tendrá el servidor de juegos instalado en una instancia de EC2 de AWS . Hay al menos un proceso del servidor en ejecución y listo para iniciar una sesión de juego.

Lanzamiento del cliente de juegos

Cuando la flota se haya implementado correctamente, tendrá servidores de juegos en funcionamiento y disponibles para alojar sesiones de juego. Ahora puede crear el cliente, lanzarlo y conectarse para unirse a la sesión de juego.

1. Configure el cliente del juego. En este paso, le pide al complemento que actualice un activo `GameLiftClientSettings` para el proyecto del juego. El complemento utiliza este activo para almacenar cierta información que el cliente del juego necesita a fin de conectarse al servicio Amazon GameLift Servers.
 - a. Si no ha importado ni inicializado el juego de ejemplo, cree un nuevo activo `GameLiftClientSettings`. En el menú principal del editor de Unity, selecciona Assets, Create, Amazon GameLift, Client Settings. Si crea varias copias de `GameLiftClientSettings` en el proyecto, el complemento lo detectará automáticamente y le dirá qué recurso actualizará el complemento.
 - b. En Launch Game, seleccione Configure Client: Apply Managed EC2 Settings. Esta acción actualiza la configuración del cliente del juego para que use la flota de EC2 que acaba de implementar.
2. Compile el cliente del juego. Compile un ejecutable del cliente mediante el proceso de compilación estándar de Unity. En File, Build Settings, cambie la plataforma a Windows, Mac, Linux. Si ha importado el juego de ejemplo y ha inicializado la configuración, la lista y el destino de la compilación se actualizarán automáticamente.
3. Lance el ejecutable del cliente del juego recién compilado. Para empezar a jugar, inicie entre dos y cuatro instancias de cliente y use la interfaz de usuario de cada una de ellas para unirse a una sesión de juego.

Si utiliza el cliente de juegos de ejemplo, tendrá las siguientes características:

- Un componente de inicio de sesión para jugadores. Al conectarse a un servidor de juegos en una flota de Anywhere, no hay validación de jugadores. Puede introducir cualquier valor para unirse a la sesión de juego.
- Una interfaz de usuario sencilla para unirse al juego. Cuando un cliente intenta unirse a una partida, busca automáticamente una sesión de juego activa con espacio disponible para un jugador. Si no hay ninguna sesión de juego disponible, el cliente solicita una nueva sesión de juego. Si hay alguna sesión de juego disponible, el cliente solicita unirse a esta. Al probar el juego con varios clientes simultáneos, el primer cliente inicia la sesión de juego y los demás se unen automáticamente a la sesión de juego existente.
- Sesiones de juego con espacio para cuatro jugadores. Puede lanzar hasta cuatro instancias de clientes de juego simultáneas: se unirán todas a la misma sesión de juego.

Complemento para Unity: implementación del juego en una flota de contenedores administrada

Utilice este flujo de trabajo guiado del complemento para crear una imagen de contenedor para el servidor de juegos e implementarla en una solución de alojamiento basada en contenedores. Cuando haya completado correctamente este flujo de trabajo, el servidor de juegos en contenedores se ejecutará en la nube, y podrá usar el complemento para iniciar un cliente de juego, conectarse a una sesión de juego y jugar.

Antes de comenzar

Este flujo de trabajo presupone que ha completado las siguientes tareas:

- Integración del código del servidor de juegos con el SDK del servidor de Amazon GameLift Servers. Nuestro servidor de juegos alojado debe poder comunicarse con el servicio Amazon GameLift Servers para poder responder a las solicitudes de inicio de nuevas sesiones de juego e informar sobre el estado de las sesiones de juego. Si no ha completado esta tarea, le recomendamos que siga primero el flujo de trabajo del complemento Alojamiento con Anywhere. Para obtener instrucciones sobre cómo preparar el código del servidor de juegos, consulte [Integración del código del servidor](#). Para las flotas de contenedores administradas, debe integrar el juego con la versión 5.2 o superior del SDK del servidor.

Note

Si has importado el juego de muestra, esta tarea ya está hecha.

- Empaquetado del archivo ejecutable del servidor de juegos para su ejecución en Linux.

- Recopilación de los archivos para implementarlos con la compilación del servidor de juegos. En la máquina local, cree un directorio de trabajo para organizar los archivos que se incluirán en la imagen de contenedor del servidor de juegos. Estos pueden incluir las dependencias del juego, un script para lanzar los servidores de juegos y otros procesos al iniciar un contenedor, etc.
- Integración del código de cliente de juego con Amazon GameLift Servers. Una forma de completar esta tarea consiste en añadir un recurso de muestra (incluido con el complemento) que ya esté integrado. Para obtener instrucciones sobre cómo preparar el código del cliente de juego, consulte [Integración del código del cliente](#).
- Instalación de Docker en la máquina local. Necesita tener esta herramienta instalada si desea que el complemento cree imágenes de contenedor por usted y las envíe a un repositorio de ECR. Como alternativa, puede realizar estas tareas manualmente e indicar al complemento que utilice una imagen de contenedor existente. Para obtener más información sobre cómo crear la imagen manualmente, consulte [Creación de una imagen de contenedor para Amazon GameLift Servers](#).

Para iniciar el flujo de trabajo de contenedores administrados por Amazon GameLift Servers:

- En la barra de herramientas principal del editor de Unity, elija el menú Amazon GameLift Servers y seleccione Contenedores administrados. Esta acción abre la página del plugin Host with Managed Containers, que presenta un step-by-step proceso para crear una imagen de contenedor con la versión del servidor del juego, desplegarla en una flota de contenedores y lanzar el juego.

Paso 0: Configura tu perfil

En esta sección se muestra el perfil de usuario seleccionado. Compruebe que el perfil de usuario actual es el que desea utilizar para este flujo de trabajo. Todos los recursos que cree en este flujo de trabajo se asocian a la AWS cuenta del perfil y se colocan en la AWS región predeterminada del perfil. Los permisos del usuario del perfil determinan su acceso a AWS los recursos y las acciones.

Es posible que tenga que modificar el perfil de usuario seleccionado si:

- Actualmente no hay ningún perfil seleccionado.
- Desea seleccionar un perfil diferente o crear uno nuevo.
- Debe arrancar el perfil seleccionado (si el estado de arranque está inactivo).

Configuración o cambio del perfil de usuario seleccionado

- En el Amazon GameLift Servers menú, elija Credenciales de AWS acceso abierto.

Paso 1: Evaluar la preparación del contenedor

Antes de implementar el servidor de juegos en una flota de contenedores, debe empaquetarlo en una imagen de contenedor y almacenarlo en un repositorio de Amazon ECR. El complemento puede realizar estas tareas por usted, o puede hacerlas usted manualmente. En este paso se proporciona información sobre el estado de la imagen del contenedor y el repositorio de ECR.

Utilice las preguntas de evaluación para indicar al complemento qué pasos debe seguir:

- Creación de una nueva imagen de contenedor. Si elige esta opción, el siguiente paso le pedirá la ubicación del directorio de compilación del servidor de juegos y del ejecutable de la compilación. El complemento utiliza una plantilla de Dockerfile (proporcionada por Amazon GameLift Servers) y la configura automáticamente para el juego. Puede ver la plantilla en [Creación de una imagen de contenedor para Amazon GameLift Servers](#). Tras elegir esta opción, indique dónde desea que el complemento almacene la nueva imagen:
 - Cree un nuevo repositorio de Amazon ECR y aplíquelo la imagen de contenedor. El complemento crea un repositorio de ECR privado utilizando la AWS cuenta y está predeterminado Región de AWS en el perfil de usuario seleccionado.
 - Aplique la imagen de contenedor a un repositorio de Amazon ECR creado previamente. Si elige esta opción, el siguiente paso le pedirá que seleccione un repositorio de Amazon ECR existente de una lista. La lista incluye todos los repositorios de Amazon ECR de la AWS cuenta y están predeterminados Región de AWS en el perfil de usuario seleccionado. Puede seleccionar un repositorio público o privado.
- Uso de una imagen de contenedor existente. Si ha creado una imagen manualmente, le recomendamos que utilice la plantilla de Dockerfile proporcionada por Amazon GameLift Servers, que está disponible en [Creación de una imagen de contenedor para Amazon GameLift Servers](#). Tras seleccionar esta opción, indique dónde se encuentra la imagen.
 - Imagen generada por Docker y almacenada localmente. Si elige esta opción, el complemento crea un nuevo repositorio privado de Amazon ECR y le aplica el archivo de imagen local. En el paso siguiente se le pedirá un ID de imagen, que el complemento utilizará para localizar el archivo de imagen.
 - Imagen de contenedor que ya está almacenada en un repositorio de Amazon ECR. Si elige esta opción, el siguiente paso le pedirá que seleccione un repositorio de Amazon ECR existente

y una imagen de una lista. La lista incluye todos los repositorios de Amazon ECR de la AWS cuenta y están predeterminados Región de AWS en el perfil de usuario seleccionado. Puede seleccionar un repositorio público o privado.

- Unity 6.3 para Linux requiere glibc 2.35. Amazon Linux 2023 incluye glibc 2.34. A continuación se proporciona una plantilla de Dockerfile que permite compilar glibc 2.35 desde el código fuente y configurar los binarios de Unity para usar glibc 2.35.

Plantilla de Dockerfile para una imagen contenedora de un servidor de juegos Unity 6.3

Esta plantilla contiene las instrucciones mínimas que necesita un contenedor de servidor de juegos para poder utilizarse en una flota de Amazon GameLift Servers. Modifique el contenido como sea necesario para el servidor de juegos.

```
# Base image
# -----
# Add the base image that you want to use over here,
# 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 game build directory in the 'GAME_BUILD_DIRECTORY' env variable below.
#
# Game executable
# -----
# 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 gamelift server sdk.
# This template assumes that the executable path is relative to the game build
directory.
#
# Launch parameters
# -----
# Add any launch parameters to pass into your executable in the 'LAUNCH_PARAMS' env
variable below.
#
# Default directory
# -----
# The value provided in 'HOME_DIR' below will be where the game executable and logs
exist.
```

```

#
ARG GAME_BUILD_DIRECTORY
ARG GAME_EXECUTABLE
ARG LAUNCH_PARAMS

ENV GAME_BUILD_DIRECTORY=$GAME_BUILD_DIRECTORY \
    GAME_EXECUTABLE=$GAME_EXECUTABLE \
    LAUNCH_PARAMS=$LAUNCH_PARAMS \
    HOME_DIR="/local/game"

# install dependencies as necessary
RUN yum install -y shadow-utils bison wget gcc make patchelf tar gzip && \
    yum clean all && rm -fr /var/cache

RUN mkdir -p $HOME_DIR
COPY .$GAME_BUILD_DIRECTORY/ $HOME_DIR

# Change directory to home
WORKDIR $HOME_DIR

# Build glibc 2.35 and patch Unity binaries
RUN GLIBC_VERSION="2.35" && \
    INSTALL_PREFIX="/opt/glibc-${GLIBC_VERSION}" && \
    ARCH=$(uname -m) && \
    if [ "$ARCH" = "x86_64" ]; then \
        INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-x86-64.so.2"; \
        MONO_ARCH="x86_64"; \
    elif [ "$ARCH" = "aarch64" ]; then \
        INTERPRETER="${INSTALL_PREFIX}/lib/ld-linux-aarch64.so.1"; \
        MONO_ARCH="aarch64"; \
    else \
        echo "ERROR: Unsupported architecture: $ARCH"; exit 1; \
    fi && \
    cd /tmp && \
    wget -q "https://ftp.gnu.org/gnu/glibc/glibc-${GLIBC_VERSION}.tar.gz" && \
    tar -xzf "glibc-${GLIBC_VERSION}.tar.gz" && \
    cd "glibc-${GLIBC_VERSION}" && mkdir glibc-build && cd glibc-build && \
    touch /etc/ld.so.conf && \
    ../configure --prefix="${INSTALL_PREFIX}" && \
    make -s all && make -s install && \
    cd / && rm -rf /tmp/glibc-* && \
    GLIBC_LIB="${INSTALL_PREFIX}/lib" && \
    EXECUTABLE_NAME=$(echo "$GAME_EXECUTABLE" | sed 's/\.[^\.]*$//') && \
    DATA_DIR="$HOME_DIR/${EXECUTABLE_NAME}_Data" && \

```

```

    patchelf --set-interpreter "$INTERPRETER" "$HOME_DIR/$GAME_EXECUTABLE" && \
    patchelf --set-rpath "$GLIBC_LIB:$HOME_DIR:/lib64:$DATA_DIR/MonoBleedingEdge/
$MONO_ARCH" "$HOME_DIR/$GAME_EXECUTABLE" && \
    patchelf --set-rpath "$GLIBC_LIB:/lib64" "$HOME_DIR/UnityPlayer.so" && \
    MONO_LIB="$DATA_DIR/MonoBleedingEdge/$MONO_ARCH/libmonobdwgc-2.0.so" && \
    if [ -f "$MONO_LIB" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64" "$MONO_LIB";
fi && \
    GAME_ASSEMBLY="$HOME_DIR/GameAssembly.so" && \
    if [ -f "$GAME_ASSEMBLY" ]; then patchelf --set-rpath "$GLIBC_LIB:/lib64"
"$GAME_ASSEMBLY"; fi

RUN useradd -m gamescale && \
    echo "gamescale ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers && \
    chown -R gamescale:gamescale $HOME_DIR

# Add permissions to game build
RUN chmod +x ./ $GAME_EXECUTABLE

USER gamescale

# 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"]

```

Paso 2: Configurar la implementación de imágenes

En este paso, proporcione la información que el complemento necesita para implementar la imagen de contenedor en una flota de contenedores. En este paso se le solicitará la siguiente información:

- La ubicación de la compilación del servidor de juegos, la imagen del contenedor o el repositorio de Amazon ECR, según lo que seleccione en el paso 1.
- El escenario que se utilizará para la implementación de sus contenedores administrados.
- Configuración de implementación opcional. Esta sección contiene los ajustes de configuración que utiliza el complemento de forma predeterminada. Puede modificar los valores predeterminados o conservarlos.

- El nombre del juego se establece en el nombre del proyecto de juego de forma predeterminada. Todos los recursos de AWS que crea el complemento hacen referencia al valor del nombre del juego.
- El rango de puertos, el límite de memoria y el límite de vCPU son ajustes de configuración de la flota de contenedores. Para obtener más información sobre la personalización de estos valores, consulte [Configuración de conexiones de red](#) para conocer el rango de puertos de conexión y los límites de recursos de [Establecimiento de los límites de recursos](#).
- La etiqueta de imagen de contenedor se usa para clasificar las imágenes del contenedor en Amazon ECR. El valor predeterminado es `unity-gamelift-plugin`.

Opciones de escenario de implementación

Flota de contenedores de una sola región

En este escenario, el servidor de juegos se implementa en una sola flota de contenedores. Es un buen punto de partida para probar la integración del servidor con AWS y la configuración del contenedor. Permite implementar los siguientes recursos:

- La definición del grupo de contenedores de Amazon GameLift Servers describe cómo implementar y ejecutar las imágenes de contenedor en una flota de contenedores.
- La flota de contenedores de Amazon GameLift Servers (bajo demanda) con el contenedor del servidor de juegos instalado y en ejecución, con un alias.
- Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
- Autorizador de API Gateway que vincula el grupo de usuarios con APIs.
- Lista de control de acceso (ACL) web para limitar las llamadas excesivas de los jugadores a la puerta de enlace de la API.
- Servicio de backend para realizar solicitudes al servicio Amazon GameLift Servers en nombre de los clientes de juego, por ejemplo, para solicitar sesiones de juego y unirse al juego:
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten un espacio de sesión de juego. Esta función llama a `CreateGameSession()` si no hay espacios abiertos disponibles.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.

Flota de contenedores de una sola región con FlexMatch

Este escenario implementa el servidor de juegos en una flota de contenedores, configura la ubicación de las sesiones de juego y configura el emparejamiento de FlexMatch. Este escenario resultará útil cuando esté listo para empezar a diseñar un emparejador personalizado para su solución de alojamiento. Utilice este escenario para crear los recursos básicos para la solución, y que podrá personalizar más tarde según sea necesario. Permite implementar los siguientes recursos:

- La definición del grupo de contenedores de Amazon GameLift Servers que describe cómo implementar y ejecutar las imágenes de contenedor en una flota de contenedores.
- La flota de contenedores de Amazon GameLift Servers (bajo demanda) con el contenedor del servidor de juegos instalado y en ejecución, con un alias.
- Configuración de emparejamiento de FlexMatch y reglas de emparejamiento establecidas para aceptar las solicitudes de los jugadores y formar emparejamientos.
- Cola de sesiones de juego de Amazon GameLift Servers que responde a las solicitudes de emparejamiento propuesto mediante la búsqueda del mejor recurso de alojamiento posible (en función de la viabilidad, el costo, la latencia de los jugadores, etc.) y el inicio de una sesión de juego.
- Grupo de usuarios y cliente de Amazon Cognito para permitir a los jugadores autenticarse e iniciar un juego.
- Autorizador de API Gateway que vincula el grupo de usuarios con APIs.
- Lista de control de acceso (ACL) web para limitar las llamadas excesivas de los jugadores a la puerta de enlace de la API.
- Servicio de backend para realizar solicitudes al servicio Amazon GameLift Servers en nombre de los clientes de juego, por ejemplo, para solicitar sesiones de juego y unirse al juego:
 - Puerta de enlace de la API + función de Lambda para que los jugadores soliciten un espacio de sesión de juego. Esta función llama a `StartMatchmaking()` si no hay espacios abiertos disponibles.
 - Puerta de enlace de la API + función de Lambda para que los jugadores obtengan información de la conexión para su solicitud de juego.
- Tablas de DynamoDB para almacenar tickets de emparejamiento para jugadores e información sobre las sesiones de juego.
- Tema de Amazon SNS y función Lambda para gestionar eventos. `GameSessionQueue`

Implementación de una flota de contenedores

Cuando la configuración de la flota esté completa, seleccione el botón Implementar flota de contenedores para iniciar la implementación. Este proceso puede tardar varios minutos mientras el complemento crea una imagen de contenedor y la envía a ECR, aprovisiona los recursos de alojamiento para la flota de contenedores e implementa la flota y otros recursos de AWS para el escenario de solución de alojamiento seleccionado.

Al iniciar una implementación, puede seguir el progreso de cada paso. Dependiendo de la configuración, los pasos pueden incluir lo siguiente:

- Configurar la imagen de contenedor
- Creación de un nuevo repositorio de Amazon ECR
- Crear una imagen e insertarla en Amazon ECR
- Crear la definición del grupo de contenedores
- Crear la flota de contenedores

Para obtener información más detallada sobre la implementación, seleccione Ver en la consola de administración de AWS . Una vez que la flota de contenedores alcance el estado activo, la flota estará ejecutando activamente los contenedores con los procesos del servidor que estén listos para alojar sesiones de juego.

Una vez que se complete la implementación, dispondrá de una flota de contenedores activa que estará lista para alojar sesiones de juego y aceptar conexiones de jugadores.

No es posible detener una implementación en curso. Si la implementación pasa a un estado defectuoso o falla, puede volver a empezar con la opción Reiniciar implementación.

Lanzamiento del cliente

En este punto, ya ha completado todas las tareas necesarias para lanzar e iniciar el juego multijugador alojado con Amazon GameLift Servers. Para jugar, elija Iniciar cliente para lanzar una instancia local del cliente de juego.

- Si ha implementado el escenario de flota única, puede abrir una instancia de cliente única con un jugador y especificar el mapa del servidor para desplazarse. Puede abrir una segunda instancia del cliente de juego para añadir un segundo jugador al mismo mapa de juego del servidor.
- Si ha implementado el escenario de FlexMatch, la solución de alojamiento espera a que al menos dos clientes del juego envíen solicitudes de emparejamiento. Abra al menos dos instancias del

cliente de juego con un jugador. Se emparejará a los dos jugadores y se les pedirá que se unan a una sesión de juego para la partida.

Actualización de una flota de contenedores

Si ha implementado correctamente una solución de alojamiento de contenedores administrados, puede utilizar la característica Actualizar implementación. Esta opción le permite actualizar los ajustes de configuración de una flota de contenedores implementada sin necesidad de crear una nueva flota.

Al actualizar una implementación, puede implementar una imagen de contenedor con una compilación de servidor de juegos diferente, cambiar el repositorio de Amazon ECR, elegir un escenario de implementación diferente y personalizar los ajustes de configuración opcionales.

Cuando lo tenga todo listo para implementar los cambios, elija Actualizar. El tiempo necesario para una actualización de la implementación es similar al de una implementación completa. Para obtener información detallada sobre la implementación, seleccione Ver en la consola AWS de administración.

Limpie los recursos desplegados.

Como práctica recomendada, limpie los AWS recursos de su solución de contenedores gestionados tan pronto como ya no los necesite. Podría seguir incurriendo en costos por estos recursos si no los elimina.

Elimine los siguientes recursos:

- Pila de recursos de contenedores administrados. Los recursos de esta pila dependen del escenario de implementación que haya seleccionado. Para eliminar toda la pila, usa la CloudFormation consola. Las pilas que se generan a partir del complemento Amazon GameLift Servers utilizan la siguiente convención de nomenclatura: `GameLiftPluginForUnity-{GameName}-Containers`. Espere a que se complete el proceso de eliminación de la pila antes de iniciar una nueva implementación de contenedores administrados en el complemento. Para obtener más información, consulte [Eliminar una pila de la CloudFormation consola](#).
- Repositorio de Amazon ECR. Si utilizó el complemento para crear un repositorio para la imagen de contenedor, quizás le interese eliminar los repositorios que ya no son necesarios. No es necesario eliminar un repositorio antes de restablecer una implementación de contenedores administrados. Si actualiza o restablece una implementación, el complemento usará automáticamente el mismo repositorio, a menos que se le indique usar otro. Para obtener más información, consulte [Eliminación de un repositorio privado en Amazon ECR](#).

Complemento de Amazon GameLift Servers para Unity (SDK 4 del servidor)

Note

En este tema se proporciona información sobre una versión anterior del complemento de Amazon GameLift Servers para Unity. La versión 1.x usa el SDK del servidor para la versión 4.x o versiones anteriores de Amazon GameLift Servers. Para obtener documentación sobre la versión más reciente del complemento, que utiliza el SDK del servidor 5.x y es compatible con características más recientes, como el alojamiento de Amazon GameLift Servers Anywhere y de contenedores administrados, consulte [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

Amazon GameLift Servers proporciona herramientas a fin de preparar los servidores de juegos multijugador para que se ejecuten en Amazon GameLift Servers. El complemento de Amazon GameLift Servers para Unity facilita la integración de Amazon GameLift Servers en los proyectos de juegos de Unity, así como la implementación de los recursos de Amazon GameLift Servers para el alojamiento en la nube. Usa el complemento de Unity para acceder Amazon GameLift Servers APIs e implementar AWS CloudFormation plantillas para escenarios de juego comunes.

Una vez que hayas configurado el complemento, puedes probar el [ejemplo de Amazon GameLift Servers Unity](#) GitHub.

Temas

- [Integración de Amazon GameLift Servers en un proyecto de servidor de juegos de Unity](#)
- [Integración de Amazon GameLift Servers en un proyecto de cliente de juegos de Unity](#)
- [Instalación y configuración del complemento](#)
- [Pruebas de juegos en el ámbito local](#)
- [Implementación de un escenario](#)
- [Integración de juegos con Amazon GameLift Servers en Unity](#)
- [Importación y ejecución de un juego de ejemplo](#)

Integración de Amazon GameLift Servers en un proyecto de servidor de juegos de Unity

Note

En este tema se proporciona información sobre una versión anterior del complemento de Amazon GameLift Servers para Unity. La versión 1.x usa el SDK del servidor para la versión 4.x o versiones anteriores de Amazon GameLift Servers. Para obtener documentación sobre la versión más reciente del complemento, que utiliza el SDK del servidor 5.x y es compatible con características más recientes, como el alojamiento de Amazon GameLift Servers Anywhere y de contenedores administrados, consulte [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

En este tema le ayudaremos a preparar su servidor de juegos personalizado para el alojamiento en Amazon GameLift Servers. El servidor de juegos debe poder informar a Amazon GameLift Servers de su estado, iniciar y detener sesiones de juego cuando se le solicite, y realizar otras tareas. Para obtener más información, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Requisitos previos

Antes de integrar el servidor de juegos, realice las siguientes tareas:

- [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#)
- [Instalación y configuración del complemento](#)

Configuración de un proceso del servidor nuevo

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Establezca la comunicación con Amazon GameLift Servers e informe de que el proceso del servidor está listo para alojar una sesión de juego.

1. Inicie el SDK del servidor mediante una llamada a `InitSDK()`

2. Para preparar el servidor para que acepte una sesión de juego, llame a `ProcessReady()` con el puerto de conexión y los detalles de la ubicación de la sesión de juego. Incluya los nombres de las funciones de devolución de llamada que invoca el servicio de Amazon GameLift Servers, como `OnGameSession()`, `OnGameSessionUpdate()`, `OnProcessTerminate()` o `OnHealthCheck()`. Amazon GameLift Servers podría tardar unos minutos en proporcionar una devolución de llamada.
3. Amazon GameLift Servers actualiza el estado del proceso del servidor a `ACTIVE`.
4. Amazon GameLift Servers llama a `onHealthCheck` de forma periódica.

El siguiente ejemplo de código muestra cómo configurar un proceso de servidor sencillo con 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;
}
```

Inicio de una sesión de juego

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Una vez completada la inicialización del juego, podrá iniciar una sesión de juego.

1. Implemente la función de devolución de llamada `onStartGameSession`. Amazon GameLift Servers invoca este método para iniciar una nueva sesión de juego en el proceso del servidor y recibir las conexiones de los jugadores.
2. Para activar una sesión de juego, llame a `ActivateGameSession()`. Para obtener más información sobre los SDK, consulte [SDK del servidor de C# para Amazon GameLift Servers 4.x: acciones](#).

El siguiente código de ejemplo muestra cómo iniciar una sesión de juego con 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();
}
```

```
}
```

Finalización de una sesión de juego

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Informe a Amazon GameLift Servers cuando finalice una sesión de juego. Como práctica recomendada, cierre los procesos del servidor una vez finalizada la sesión de juego para reciclar y actualizar los recursos de alojamiento.

1. Configure una función denominada `onProcessTerminate` para recibir solicitudes de Amazon GameLift Servers y llamar a `ProcessEnding()`.
2. El estado del proceso cambia a `TERMINATED`.

El siguiente ejemplo describe cómo finalizar un proceso de una sesión de juego.

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

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

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

Creación de una compilación del servidor y carga en Amazon GameLift Servers

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Una vez que haya integrado el servidor de juegos con Amazon GameLift Servers, cargue los archivos de compilación en una flota para que Amazon GameLift Servers pueda implementarlos para

el alojamiento de juegos. Para obtener más información sobre cómo cargar el servidor en Amazon GameLift Servers, consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#).

Integración de Amazon GameLift Servers en un proyecto de cliente de juegos de Unity

Note

En este tema se proporciona información sobre una versión anterior del complemento de Amazon GameLift Servers para Unity. La versión 1.x usa el SDK del servidor para la versión 4.x o versiones anteriores de Amazon GameLift Servers. Para obtener documentación sobre la versión más reciente del complemento, que utiliza el SDK del servidor 5.x y es compatible con características más recientes, como el alojamiento de Amazon GameLift Servers Anywhere y de contenedores administrados, consulte [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

Este tema le ayudará a configurar un cliente de juegos para conectarse a las sesiones de juego alojadas en Amazon GameLift Servers a través de un servicio de backend. Úsalo Amazon GameLift Servers APIs para iniciar el matchmaking, solicitar la ubicación de una sesión de juego y mucho más.

Añada código al proyecto del servicio de backend para permitir la comunicación con el servicio de Amazon GameLift Servers. Un servicio de back-end gestiona todas las comunicaciones del cliente del juego con el GameLift servicio. Para obtener más información sobre los servicios de backend, consulte .

El servicio de backend se encarga de las siguientes tareas de cliente de juegos:

- Personalización de la autenticación de sus jugadores..
- Solicitar información acerca de las sesiones de juego activas del servicio Amazon GameLift Servers.
- Creación de una sesión de juego nueva.
- Adición de un jugador a una sesión de juego existente.
- Eliminación de un jugador de una sesión de juego existente.

Temas

- [Requisitos previos](#)

- [Inicialización de un cliente de juegos](#)
- [Creación de una sesión de juego en una flota específica](#)
- [Adición de jugadores a las sesiones de juego](#)
- [Eliminación de un jugador de una sesión de juego](#)

Requisitos previos

Antes de configurar la comunicación del servidor de juegos con el cliente de Amazon GameLift Servers, realice las siguientes tareas:

- [Configurar una cuenta AWS de usuario](#)
- [Instalación y configuración del complemento](#)
- [Integración de Amazon GameLift Servers en un proyecto de servidor de juegos de Unity](#)
- [Implementación de flotas de alojamiento para Amazon GameLift Servers](#)

Inicialización de un cliente de juegos

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Añada código para inicializar un cliente de juegos. Ejecute este código en el momento del lanzamiento, ya que es necesario para otras funciones de Amazon GameLift Servers.

1. Inicialice `AmazonGameLiftClient`. Llame a `AmazonGameLiftClient` con una configuración de cliente predeterminada o con una configuración personalizada. Para obtener más información sobre cómo configurar un cliente, consulte [Configuración de la API de Amazon GameLift Servers](#).
2. Genere un ID de jugador único para que cada jugador se conecte a una sesión de juego. Para obtener más información, consulte [Generación de ID de jugador](#).

Los siguientes ejemplos muestran cómo configurar un Amazon GameLift Servers cliente.

```
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);
    }
}
```

Creación de una sesión de juego en una flota específica

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Añada código para iniciar sesiones de juego nuevas en las flotas implementadas y ponerlas a disposición de los jugadores. Una vez que Amazon GameLift Servers haya creado la nueva sesión de juego y devuelto un `GameSession`, podrá añadirle jugadores.

- Realice una solicitud para una nueva sesión de juego.
 - Si el juego utiliza flotas, llame a `CreateGameSession()` con un ID de la flota o alias, un nombre de sesión y el número máximo de jugadores simultáneos para el juego.
 - Si el juego utiliza colas, llame a `StartGameSessionPlacement()`

En el siguiente ejemplo se muestra cómo crear una sesión de juego.

```
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;
}
```

Adición de jugadores a las sesiones de juego

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Una vez que Amazon GameLift Servers haya creado la nueva sesión de juego y devuelto un objeto `GameSession`, podrá añadirle jugadores.

1. Reserve una ranura de jugador en una sesión de juego mediante la creación de una sesión de jugador nueva. Utilice `CreatePlayerSession` o `CreatePlayerSessions` con el ID de sesión de juego y un ID único para cada jugador.
2. Conéctese a la sesión de juego. Recupere el objeto `PlayerSession` para obtener la información de conexión de la sesión de juego. Puede utilizar esta información para establecer una conexión directa con el proceso del servidor:
 - a. Utilice el puerto especificado y el nombre de DNS o la dirección IP del proceso del servidor.
 - b. Utilice el nombre y el puerto DNS de sus flotas. El nombre y el puerto DNS son obligatorios si las flotas tienen la generación del certificado TLS habilitada.
 - c. Haga referencia al ID de sesión del jugador. Si el servidor de juegos valida las conexiones de los jugadores entrantes, el ID de sesión del jugador será obligatorio.

Los siguientes ejemplos muestran cómo reservar un spot de jugador en una sesión de juego.

```
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;
}
```

El siguiente código muestra cómo conectar a un jugador con la sesión de juego.

```
public bool ConnectPlayer(int playerId, string playerSessionId)
{
    //Call ConnectPlayer with player ID and player session ID.
    return server.ConnectPlayer(playerId, playerSessionId);
}
```

Eliminación de un jugador de una sesión de juego

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Puede eliminar a los jugadores de la sesión de juego cuando abandonen el juego.

1. Informe al servicio de Amazon GameLift Servers de que un jugador se ha desconectado del proceso del servidor. Llame a `RemovePlayerSession` con el ID de sesión del jugador.
2. Verifique que `RemovePlayerSession` devuelva `Success`. A continuación, Amazon GameLift Servers cambia el estado de la ranura de jugador a disponible, por lo que Amazon GameLift Servers puede asignarla a un jugador nuevo.

En el ejemplo siguiente se muestra cómo eliminar una sesión de jugador.

```
public void DisconnectPlayer(int playerId)
{
    //Receive the player session ID.
    string playerIdSessionId = 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());
    }
}
```

Instalación y configuración del complemento

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

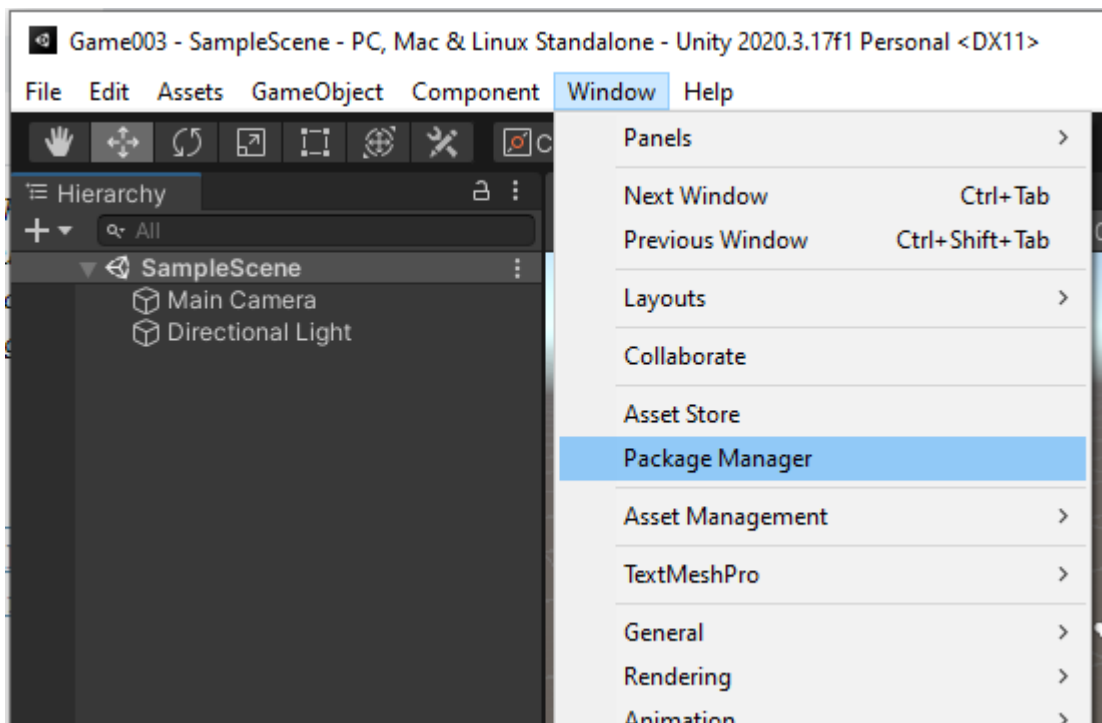
En esta sección, se describe cómo descargar, instalar y configurar el complemento de Amazon GameLift Servers para Unity (versión 1.0.0).

Requisitos previos

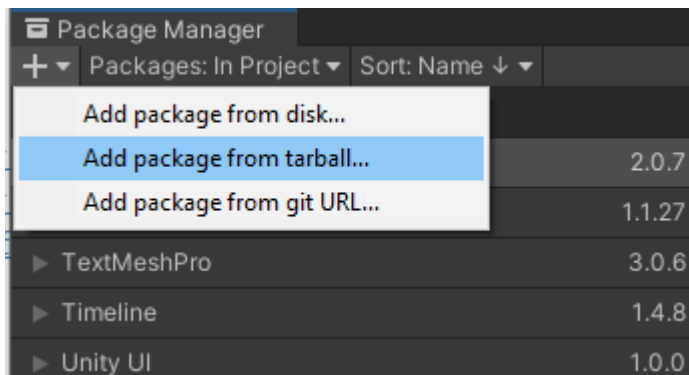
- Unity para Windows 2019.4 LTS, Windows 2020.3 LTS o Unity para MacOS
- Versión actual de Java
- Versión actual de .NET 4.x

Descarga e instalación del complemento para Unity

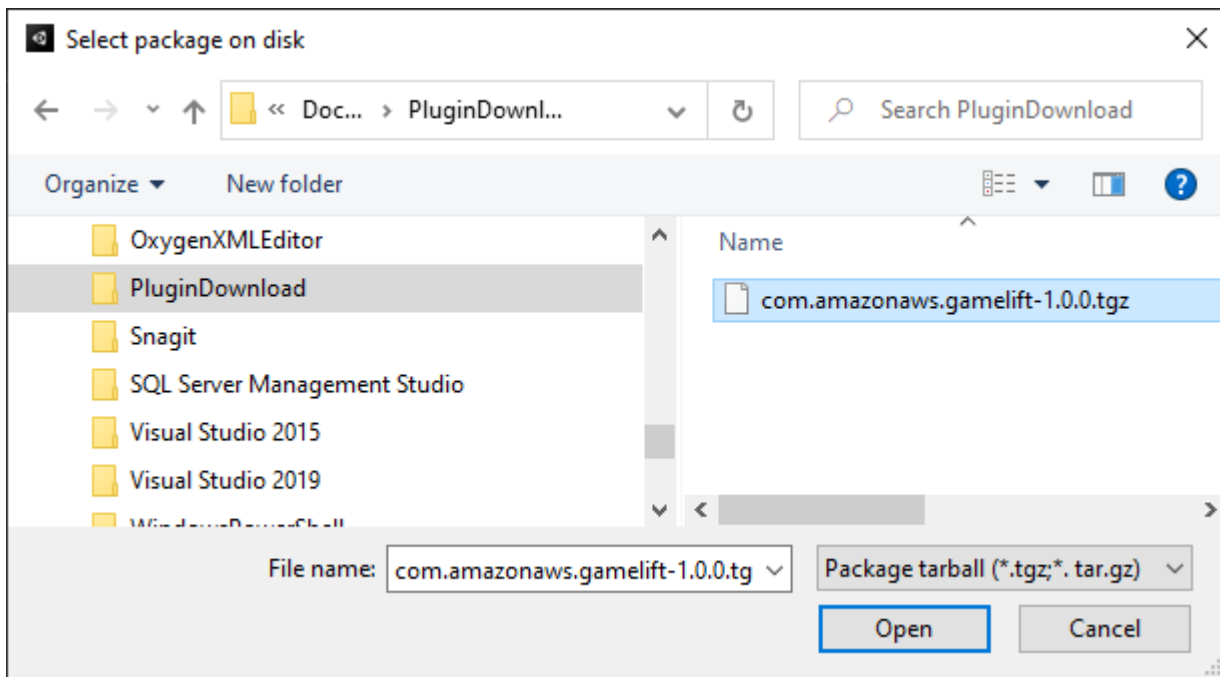
1. Descargue el complemento de Amazon GameLift Servers para Unity. Puede encontrar la versión más reciente en la página del [repositorio del complemento de Amazon GameLift Servers para Unity](#). En la [versión más reciente](#), elija Activos y, a continuación, descargue el archivo `com.amazonaws.gamelift-version.tgz`.
2. Lance Unity y elija un proyecto.
3. En la barra de navegación superior, en Ventana, elija Administrador de paquetes:



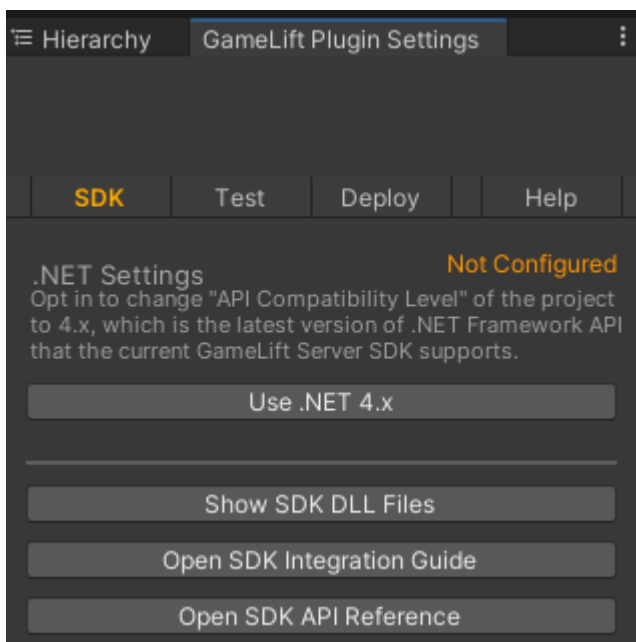
4. En la pestaña Administrador de paquetes, seleccione + y, a continuación, elija Añadir paquete desde archivo tarball... :



5. En la ventana Seleccionar paquetes en disco, navegue hasta la `com.amazonaws.gamelift` carpeta, elija el archivo `ycom.amazonaws.gamelift-version.tgz` , a continuación, elija Abrir:



- Una vez que Unity haya cargado el complemento, Amazon GameLift Servers aparecerá como un elemento nuevo en el menú de Unity. La instalación y recopilación de los scripts puede llevar unos minutos. La pestaña Configuración del complemento de Amazon GameLift Servers se abrirá automáticamente.



- En el panel del SDK, elija Usar .NET 4.x.

Cuando esté configurado, el estado cambiará de No configurado a Configurado.

Pruebas de juegos en el ámbito local

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Utilice Amazon GameLift Servers local para ejecutar Amazon GameLift Servers en su dispositivo local. Puede utilizar Amazon GameLift Servers local para verificar los cambios de código en cuestión de segundos y sin necesidad de una conexión de red.

Configuración de pruebas locales

1. En la ventana del complemento para Unity, elija la pestaña Prueba.
2. En el panel Prueba, elija Descargar Amazon GameLift Servers local. El complemento para Unity abrirá una ventana del navegador y descargará el archivo `GameLift_06_03_2021.zip` en la carpeta de descargas.

La descarga incluye el SDK del servidor de C#, los archivos de origen de .NET y un componente de .NET compatible con Unity.

3. Descomprima el archivo `GameLift_06_03_2021.zip` descargado.
4. En la ventana Configuración del complemento de Amazon GameLift Servers, elija Ruta de Amazon GameLift Servers local, diríjase a la carpeta sin comprimir, elija el archivo `GameLiftLocal.jar` y, a continuación, seleccione Abrir.

Cuando esté configurado, el estado de la prueba local cambiará de No configurado a Configurado.

5. Verifique el estado de JRE. Si el estado es No configurado, seleccione Descargar JRE e instale la versión de Java recomendada.

Después de instalar y configurar el entorno de Java, el estado cambiará a Configurado.

Ejecución del juego local

1. En la pestaña del complemento para Unity, elija la pestaña Prueba.
2. En el panel Prueba, elija Abrir interfaz de usuario de prueba local.

3. En la ventana Prueba local, especifique una ruta de un archivo ejecutable del servidor. Elija ... para seleccionar la ruta y el nombre del archivo ejecutable de la aplicación del servidor.
4. En la ventana Prueba local, especifique un puerto local de GL.
5. Elija Implementar y ejecutar para implementar y ejecutar el servidor.
6. Para detener el servidor de juegos, elija Detener o cierre las ventanas del servidor de juegos.

Implementación de un escenario

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Un escenario usa una CloudFormation plantilla para crear los recursos que necesitas para implementar una solución de alojamiento en la nube para tu juego. En esta sección se describen los escenarios que ofrece Amazon GameLift Servers y cómo utilizarlos.

Requisitos previos

Para implementar el escenario, necesita un rol de IAM para el servicio Amazon GameLift Servers. Para obtener información sobre cómo crear un rol para Amazon GameLift Servers, consulte [Configurar una cuenta AWS de usuario](#).

Cada situación requiere permisos para los siguientes recursos:

- Amazon GameLift Servers
- Amazon S3
- CloudFormation
- API Gateway
- AWS Lambda
- AWS WAFV2
- Amazon Cognito

Escenarios

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

El complemento de Amazon GameLift Servers para Unity incluye los siguientes escenarios:

Solo autenticación

En este escenario se crea un servicio de backend del juego que realiza la autenticación del jugador sin la capacidad del servidor de juegos. La plantilla crea los siguientes recursos en su cuenta:

- Un grupo de usuarios de Amazon Cognito para almacenar la información de autenticación de los jugadores.
- Un AWS Lambda controlador respaldado por un punto final REST de Amazon API Gateway que inicia los juegos y ve la información de conexión del juego.

Flota de una sola región

En este escenario se crea un servicio de backend de juegos con una sola flota de Amazon GameLift Servers. Crea los siguientes recursos:

- Grupo de usuarios de Amazon Cognito para un jugador para permitir a los jugadores autenticarse e iniciar un juego.
- Un AWS Lambda controlador para buscar una sesión de juego existente con un espacio libre para jugadores en la flota. Si no encuentra una ranura abierta, cree una nueva sesión de juego.

Flota de varias regiones con una cola y un emparejador personalizado

En este escenario se forman emparejamientos mediante el uso de colas de Amazon GameLift Servers y un emparejador personalizado para agrupar a los jugadores más antiguos en el grupo de espera. Crea los siguientes recursos:

- Un tema de Amazon Simple Notification Service en el que Amazon GameLift Servers publica mensajes. Para obtener más información sobre los temas y notificaciones de SNS, consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego..](#)

- Una función de Lambda que se invoca mediante el mensaje que comunica los detalles de ubicación y conexión al juego.
- Una tabla de Amazon DynamoDB para almacenar los detalles de ubicación y conexión al juego. Las llamadas a `GetGameConnection` se leen de esta tabla y se devuelve la información de conexión al cliente de juego.

Flotas de spot con una cola y un emparejador personalizado

En este escenario se forman emparejamientos mediante colas de Amazon GameLift Servers y un emparejador personalizado, y se configuran tres flotas. Crea los siguientes recursos:

- Dos flotas de spot que contienen diferentes tipos de instancias para garantizar la durabilidad en caso de que spot no esté disponible.
- Una flota bajo demanda que actúa como respaldo para las demás flotas de spot. Para obtener más información sobre el diseño de las flotas, consulte [Personalizaciones de recursos de alojamiento](#).
- Una Amazon GameLift Servers cola para mantener una alta disponibilidad de servidores y un coste bajo. Para obtener más información sobre las prácticas recomendadas sobre colas, consulte [Personalización de colas de sesiones de juego](#).

FlexMatch

Este escenario utiliza FlexMatch, un servicio de emparejamiento administrado, para unir a los jugadores del juego. Para obtener más información acerca de FlexMatch, consulte [Qué es Amazon GameLift Servers FlexMatch](#). En este escenario se crean los siguientes recursos:

- Una función de Lambda para crear un ticket de emparejamiento después de recibir las solicitudes de `StartGame`.
- Una función de Lambda independiente para escuchar los eventos de emparejamiento de FlexMatch.

Para evitar cargos innecesarios Cuenta de AWS, elimina los recursos creados por cada escenario una vez que hayas terminado de usarlos. Elimina la CloudFormation pila correspondiente.

Actualice AWS las credenciales

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

El complemento de Amazon GameLift Servers para Unity requiere credenciales de seguridad para implementar un escenario. Puede crear credenciales nuevas o utilizar las existentes.

Para obtener más información sobre la configuración de las credenciales, consulte [Descripción y obtención de AWS las credenciales](#).

Para actualizar AWS las credenciales

1. En Unity, en la pestaña del complemento para Unity, elija la pestaña Implementar.
2. En el panel Implementar, elija Credenciales de AWS .
3. Puede crear AWS credenciales nuevas o elegir las existentes.
 - Para crear credenciales, elija Crear nuevo perfil de credenciales y, a continuación, especifique el nombre del nuevo perfil, el ID de la clave de acceso de AWS , la clave secreta de AWS y la Región de AWS.
 - Para elegir las credenciales existentes, seleccione Elegir perfil de credenciales existente y, a continuación, seleccione un nombre de perfil y la Región de AWS.
4. En la ventana Actualizar AWS credenciales, seleccione Actualizar perfil de credenciales.

Actualización del arranque de la cuenta

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

La ubicación de arranque es un bucket de Amazon S3 que se utiliza durante la implementación. Se utiliza para almacenar los recursos del servidor de juegos y otras dependencias. La Región de

AWS que elija para el depósito debe ser la misma región que utilizará para la implementación del escenario.

Para obtener más información sobre los buckets de Amazon S3, consulte [Creación, configuración y trabajo con buckets de Amazon Simple Storage Service](#).

Para actualizar la ubicación de arranque de la cuenta, realice el siguiente procedimiento:

1. En Unity, en la pestaña del complemento para Unity, elija la pestaña Implementar.
2. En el panel Implementar, seleccione Actualizar arranque de cuenta.
3. En la ventana de Arranque de cuenta, elija un bucket de Amazon S3 existente o cree un bucket de Amazon S3 nuevo:
 - Para elegir un bucket existente, seleccione Elegir un bucket de Amazon S3 existente y Actualizar para guardar la selección.
 - Seleccione Crear un bucket de Amazon S3 nuevo para crear un nuevo bucket de Amazon Simple Storage Service y, a continuación, elija una política. La política especifica cuándo caducará el bucket de Amazon S3. Elija Crear para crear el bucket.

Implementación de un escenario de juegos

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

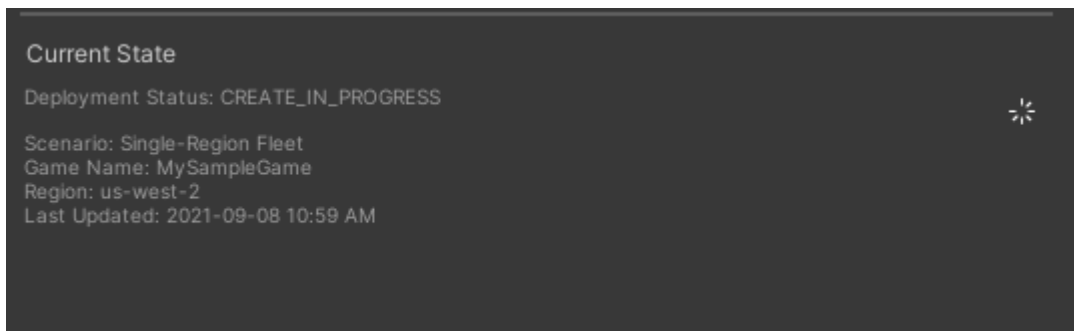
Puede utilizar un escenario para probar el juego con Amazon GameLift Servers. Cada escenario utiliza una plantilla de CloudFormation para crear una pila con los recursos necesarios. La mayoría de los escenarios requieren un servidor de juegos ejecutable y una ruta de compilación. Al implementar el escenario, Amazon GameLift Servers copia los recursos del juego en la ubicación de arranque como parte de la implementación.

Debe configurar AWS las credenciales y el bootstrap de una AWS cuenta para implementar un escenario.

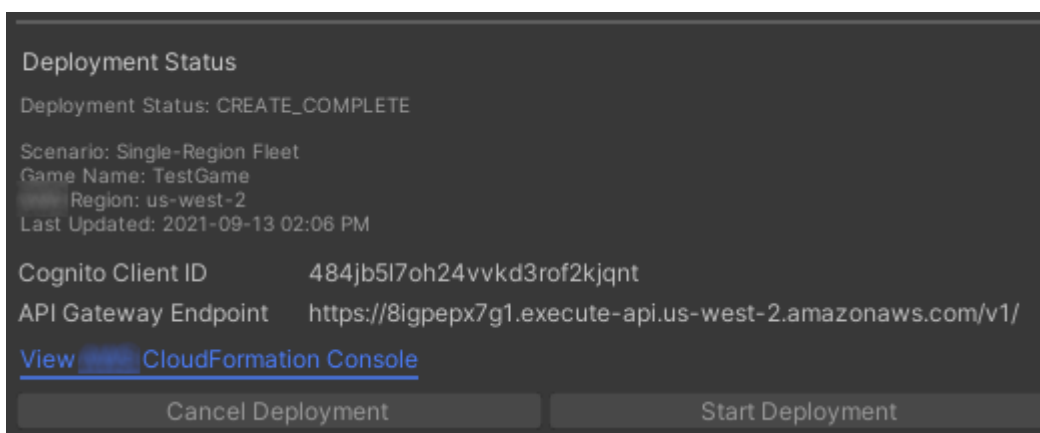
Implementación de un escenario

1. En Unity, en la pestaña del complemento para Unity, elija la pestaña Implementar.

2. En el panel Implementar, seleccione Abrir interfaz de usuario de implementación.
3. En la ventana Implementación, elija un escenario.
4. Especifique un nombre de juego. Deben ser únicos. El nombre del juego forma parte del nombre de la CloudFormation pila cuando despliegas el escenario.
5. Elija la ruta de la carpeta de compilación del servidor de juegos. La ruta de la carpeta de compilación apunta a la carpeta que contiene el archivo ejecutable del servidor y las dependencias.
6. Elija la ruta del archivo .exe de compilación del servidor de juegos. La ruta del archivo ejecutable de compilación apunta al archivo ejecutable del servidor de juegos.
7. Elija Iniciar implementación para comenzar a implementar un escenario. Puede seguir el estado de la actualización en la ventana Implementación, en Estado actual. Los escenarios pueden tardar varios minutos en implementarse.



8. Cuando el escenario finalice la implementación, el estado actual se actualizará para incluir el ID de cliente de Cognito y el punto de conexión de API Gateway, que podrá copiar y pegar en el juego.



9. Para actualizar la configuración del juego, en el menú de Unity, elija Ir a la configuración de conexión del cliente. Aparecerá la pestaña Inspector en el lado derecho de la pantalla de Unity.
10. Anule la selección de la opción Modo de prueba local.

11. Especifique el punto de conexión de API Gateway y el ID de cliente de Cognito. Elige el mismo Región de AWS que usaste para la implementación del escenario. A continuación, podrá volver a compilar y ejecutar el cliente de juego con los recursos del escenario implementados.

Eliminación de los recursos creados para el escenario

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Para eliminar los recursos creados para el escenario, elimine la pila de CloudFormation correspondiente.

Para eliminar recursos creados por el escenario, realice el siguiente procedimiento:

1. En la ventana de implementación del Amazon GameLift Servers complemento para Unity, selecciona View AWS CloudFormation Console para abrir la CloudFormation consola.
2. En la CloudFormation consola, selecciona Pilas y, a continuación, elige la pila que incluya el nombre del juego especificado durante el despliegue.
3. Elija Eliminar para borrar la pila. La pila tarda unos minutos en eliminarse. Tras CloudFormation eliminar la pila utilizada en el escenario, su estado cambia a. ROLLBACK_COMPLETE

Integración de juegos con Amazon GameLift Servers en Unity

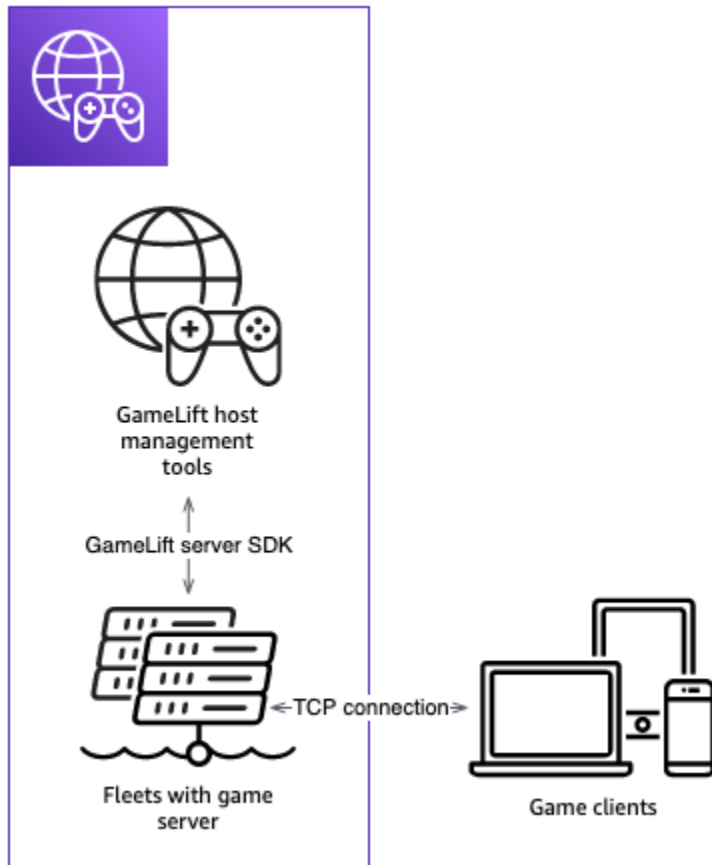
Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Integre su juego de Unity con Amazon GameLift Servers realizando las siguientes tareas:

- [Integración de Amazon GameLift Servers en un proyecto de servidor de juegos de Unity](#)
- [Integración de Amazon GameLift Servers en un proyecto de cliente de juegos de Unity](#)

En el siguiente diagrama se muestra un ejemplo de flujo de integración de un juego. En el diagrama, se implementa una flota con el servidor de juegos en Amazon GameLift Servers. El cliente de juego se comunica con el servidor de juegos, que se comunica con Amazon GameLift Servers.



Importación y ejecución de un juego de ejemplo

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

El complemento de Amazon GameLift Servers para Unity incluye un juego de ejemplo que puede utilizar para explorar los aspectos básicos de la integración del juego con Amazon GameLift Servers. En esta sección, compile el cliente y el servidor de juegos y, a continuación, realice las pruebas de forma local con Amazon GameLift Servers local.

Requisitos previos

- [Configurar una cuenta AWS de usuario](#)
- [Instalación y configuración del complemento](#)

Compilación y ejecución del servidor de juegos de ejemplo

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Configure los archivos del servidor de juegos del juego de ejemplo.

1. En Unity, en el menú, elija Amazon GameLift Servers y, a continuación, seleccione Importar un juego de muestra.
2. En la ventana Importar un juego de muestra, elija Importar para importar el juego, sus recursos y dependencias.
3. Compile el servidor de juegos. En Unity, en el menú, elija Amazon GameLift Servers y, a continuación, seleccione Aplicar configuración de compilación del servidor de ejemplo de Windows o Aplicar configuración de compilación del servidor de ejemplo de MacOS. Después de configurar los ajustes del servidor de juegos, Unity vuelve a compilar los recursos.
4. En Unity, en el menú, elija Archivo y, a continuación, elija Compilación. Elija Compilación del servidor, Compilación y, a continuación, seleccione una carpeta de compilación específica para los archivos del servidor.

Unity compila el servidor de juegos de ejemplo y coloca el archivo ejecutable y los recursos necesarios en la carpeta de compilación especificada.

Compilación y ejecución del cliente de juegos de ejemplo

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Configure los archivos del cliente de juegos del juego de ejemplo.

1. En Unity, en el menú, elija Amazon GameLift Servers y, a continuación, seleccione Aplicar configuración de compilación del cliente de ejemplo de Windows o Aplicar configuración de compilación del cliente de ejemplo de MacOS. Una vez configurados los ajustes del cliente de juegos, Unity volverá a compilar los recursos.
2. En Unity, en el menú, seleccione Ir a la configuración del cliente. Aparecerá la pestaña Inspector en el lado derecho de la pantalla de Unity. En la pestaña Configuración del cliente de Amazon GameLift Servers, seleccione Modo de prueba local.
3. Compile el cliente de juegos. En Unity, en el menú, elija Archivo. Confirme que la opción Compilación del servidor no esté activa, elija Compilación y, a continuación, seleccione una carpeta de compilación específica para los archivos del cliente.

Unity compila el cliente de juegos de ejemplo y coloca el archivo ejecutable y los recursos necesarios en la carpeta de compilación del cliente especificada.

4. No ha compilado el servidor y el cliente de juegos. En los siguientes pasos podrá ejecutar el juego y observar cómo interactúa con Amazon GameLift Servers.

Realización de la prueba del juego de ejemplo de forma local

 Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Ejecute el juego de ejemplo que ha importado con Amazon GameLift Servers local.

1. Lance el servidor de juegos. En Unity, en la pestaña del complemento para Unity, elija la pestaña Implementar.
2. En el panel Prueba, elija Abrir interfaz de usuario de prueba local.
3. En la ventana Prueba local, especifique una ruta de archivo .exe del servidor de juegos. La ruta debe incluir el nombre del archivo ejecutable. Por ejemplo, `C:/MyGame/GameServer/MyGameServer.exe`.
4. Elija Implementar y ejecutar. El complemento de Unity inicia el servidor del juego y abre una ventana de registro Amazon GameLift Servers local. La ventana contiene mensajes de registro, incluidos los mensajes enviados entre el servidor de juegos y Amazon GameLift Servers local.

5. Lance el cliente de juegos. Busque la ubicación de la compilación con el cliente de juegos de ejemplo y elija el archivo ejecutable.
6. En el juego de ejemplo de Amazon GameLift Servers, especifique un correo electrónico y una contraseña y, a continuación, elija Iniciar sesión. El correo electrónico y la contraseña no están validados o no se utilizan.
7. En el juego de ejemplo de Amazon GameLift Servers, elija Iniciar. El cliente de juegos busca una sesión de juego. Si no encuentra una sesión, crea una. A continuación, el cliente de juegos inicia la sesión de juego. Puede ver la actividad del juego en los registros.

Ejemplos de registros del servidor de juegos

```
...
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
```

Ejemplos de registros de Amazon GameLift Servers local

```
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"

```

Detención de un proceso del servidor

Note

Este tema hace referencia a la versión 1.0.0 del complemento de Amazon GameLift Servers para Unity, que usa el SDK del servidor 4.x o una versión anterior.

Cuando termine con el juego de ejemplo, apague el servidor en Unity.

1. En el cliente de juegos, elija Salir o cierre la ventana para detener el cliente de juegos.
2. En Unity, en la ventana Pruebas locales, seleccione Detener o cierre las ventanas del servidor de juegos para detener el servidor.

Integración de un servidor de juegos con Amazon GameLift Servers

Su servidor de juegos personalizado debe poder interactuar con Amazon GameLift Servers cuando esté implementado y ejecutándose en recursos de alojamiento. En los temas de esta sección se explica cómo integrar su proyecto de servidor de juegos con el SDK del servidor de Amazon GameLift Servers para añadir funciones necesarias y opcionales. Esta sección incluye instrucciones generales que puede utilizar con cualquiera de los servidores de idiomas específicos. SDKs También incluye directrices detalladas sobre la integración de proyectos de juego de Unreal Engine y de Unity.

Temas

- [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
- [Integración de SDK del servidor para motores de juegos](#)

Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor

En este tema se proporcionan instrucciones generales sobre la funcionalidad del SDK del servidor que debe añadir al código del servidor de juegos. Esta funcionalidad permite que los procesos del servidor de juegos se comuniquen con el servicio Amazon GameLift Servers. Siga estas instrucciones para los servidores de juegos que vaya a implementar en flotas de EC2 administradas por Amazon GameLift Servers, flotas de contenedores administrados o flotas de Anywhere.

Los procesos del servidor de juegos se comunican con el servicio Amazon GameLift Servers para recibir instrucciones del servicio e informar del estado de los procesos del servidor y del estado de las sesiones de juego. Para obtener información detallada sobre las interacciones entre los componentes de la solución de alojamiento de juegos (servidor del juego, servicio de backend, cliente del juego y Amazon GameLift Servers), consulte [client/server Interacciones del juego con Amazon GameLift Servers](#).

Obtención del SDK del servidor

Para preparar el juego para el alojamiento, añada el SDK del servidor de Amazon GameLift Servers al proyecto de servidor de juegos. El SDK del servidor está disponible en varios idiomas. Para obtener más información sobre el soporte de herramientas para servidores de juegos, incluido el SDK del servidor, consulte [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#).

Descargue el SDK del servidor correspondiente a su lenguaje de desarrollo en github.com/amazon-gamelift. Puede encontrar repositorios para las distintas versiones del SDK del servidor, los complementos y otras herramientas de desarrollo.

Note

Si está desarrollando con Unreal Engine o Unity, descargue el complemento de Amazon GameLift Servers para estos motores de juegos. El SDK del servidor está integrado y listo para usarse. Consulte estos enlaces para obtener información adicional:

- Unreal Engine ([descargar el complemento](#)) ([guía de integración](#))
- Unity ([descargar el complemento](#)) ([guía de integración](#))

Referencias de la API del SDK del servidor:

- [SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#)
- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)
- [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
- [SDK del servidor de Go para Amazon GameLift Servers: acciones](#)

Inicialización del proceso del servidor

Añada código para establecer la comunicación con el servicio Amazon GameLift Servers e informar cuando el proceso del servidor esté listo para alojar una sesión de juego. Este código debe ejecutarse antes que cualquier código de Amazon GameLift Servers.

1. Inicialice un cliente de API de Amazon GameLift Servers mediante una llamada a `InitSdk()`. Si está preparando el servidor de juegos para que se ejecute en flotas de EC2 administradas por Amazon GameLift Servers, utilice el valor `InitSDK()` predeterminado ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) sin parámetros. El cliente de API se encarga de la conexión al servicio Amazon GameLift Servers en su nombre.

Note Si está preparando el servidor de juegos para usarlo en una flota de Amazon GameLift Servers Anywhere:

Inicialice el cliente de API de Amazon GameLift Servers llamando a `InitSdk()` con los siguientes valores `ServerParameters`:

- La URL del WebSocket que se utiliza para conectarse al servidor de juegos.
- El ID del proceso utilizado para alojar su servidor de juegos.

- El ID del proceso utilizado para alojar los procesos del servidor de juegos.
- El ID de la flota que contiene su recurso de computación de Amazon GameLift Servers Anywhere.
- El token de autorización generado por la operación [GetComputeAuthToken](#) de Amazon GameLift Servers.

2. Notifique al servicio que el proceso del servidor está listo para alojar una sesión de juego. Llame a `ProcessReady()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) con los siguientes `ProcessParameters`. Cada proceso del servidor de juegos debe llamar a `ProcessReady()` una sola vez.

- Número de puerto del proceso del servidor. Cuando el proceso del servidor inicia una sesión de juego, proporciona el puerto al servicio Amazon GameLift Servers, que actualiza la información de la sesión de juego. El juego puede recuperar esta información y proporcionarla a los clientes del juego, que la utilizan para conectarse al proceso del servidor y unirse a la sesión de juego.
- Las ubicaciones de los archivos que desea que Amazon GameLift Servers almacene para usted. Pueden incluir registros de las sesiones de juego y otros archivos que el proceso del servidor genera durante una sesión de juego. Aunque Amazon GameLift Servers guarda estos archivos temporalmente en el equipo en el que se ejecuta el proceso del servidor, solo están disponibles hasta que la instancia se cierra. Puede acceder a los archivos almacenados a través de la [Amazon GameLift Servers consola](#) o llamando a la operación Amazon GameLift Servers API [GetGameSessionLogUrl\(\)](#).

 Si está preparando el servidor de juegos para usarlo en una flota de contenedores administrada por Amazon GameLift Servers:

No es necesario que especifique parámetros de registro para las flotas de contenedores. En su lugar, envíe los datos de las sesiones de juego y otros datos de registro a la salida estándar. Las flotas de contenedores capturan automáticamente toda la salida estándar de los contenedores como un flujo de registro.

- Las siguientes funciones de devolución de llamada permiten a Amazon GameLift Servers enviar mensajes o peticiones a un proceso del servidor de juegos. Usted debe implementar cada una de estas funciones en el código del servidor de juegos. Para obtener más información, consulte `ProcessParameters`([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) .

- (Opcional) `onHealthCheck`: Amazon GameLift Servers llama a esta función con frecuencia para solicitar al servidor un informe del estado.
- `onStartGameSession`— Amazon GameLift Servers llama a esta función en respuesta a la solicitud del cliente [CreateGameSession\(\)](#).
- `onProcessTerminate`: Amazon GameLift Servers fuerza la detención del proceso del servidor, lo que permite que se cierre sin problemas.
- (Opcional) `onUpdateGameSession`: Amazon GameLift Servers ofrece un objeto de sesión de juego actualizado al servidor de juegos o proporciona una actualización de estado de una solicitud de reposición de emparejamiento. La característica de [reposición de FlexMatch](#) requiere esta devolución de llamada.

También puede configurar un servidor de juegos para que pueda acceder de forma segura a otros recursos de AWS que le pertenezcan o que estén bajo su control. Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Notificación del estado del proceso del servidor (opcional)

Añada código a su servidor de juegos para implementar la función de devolución de llamada `onHealthCheck()`. Amazon GameLift Servers invoca este método de devolución de llamadas periódicamente para recopilar métricas de estado. Al implementar esta función de devolución de llamada, realice el siguiente procedimiento:

- Evalúe el estado del proceso del servidor. Por ejemplo, puede informar de que el proceso del servidor no está en buen estado si alguna dependencia externa ha fallado.
- Complete la evaluación del estado y responda a la devolución de llamada en un plazo de 60 segundos. Si Amazon GameLift Servers no recibe una respuesta durante ese periodo, considerará automáticamente que el proceso del servidor no está funcionando correctamente.
- Devuelva un valor booleano: `true` para buen estado y `false` para mal estado.

Si no implementa una devolución de llamada de comprobación de estado, Amazon GameLift Servers considerará que el proceso del servidor está en buen estado a menos que el servidor no responda.

El servicio Amazon GameLift Servers utiliza el estado de los procesos del servidor para finalizar los procesos que no funcionan correctamente y liberar recursos. Si se sigue informando de que un

proceso del servidor no funciona correctamente o no responde a tres comprobaciones de estado consecutivas, el servicio puede cerrar el proceso e iniciar uno nuevo. El servicio recopila métricas sobre el estado de los procesos de servidor de una flota.

(Opcional) Obtención de un certificado TLS

Si el proceso del servidor se ejecuta en una flota que tiene activada la generación de certificados TLS, puede recuperar el certificado TLS para establecer una conexión segura con un cliente de juegos y cifrar la comunicación cliente/servidor. En la instancia se almacena una copia del certificado. Para obtener la ubicación del archivo, llame a `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) .

Inicio de una sesión de juego

Añada código para implementar la función de devolución de llamada `onStartGameSession`. Amazon GameLift Servers invoca esta devolución de llamada para iniciar una sesión de juego en el proceso del servidor.

La `onStartGameSession` función toma un [GameSession](#) objeto como parámetro de entrada. Este objeto incluye información clave de la sesión de juego, como el número máximo de jugadores. También puede incluir datos del juego y de los jugadores. Durante el proceso de implementación de la función se deben realizar las siguientes tareas:

- Inicie acciones para crear una nueva sesión de juego en función de las propiedades de `GameSession`. Como mínimo, el servidor de juegos debe asociar el ID de sesión del juego, al que los clientes del juego hacen referencia cuando se conectan al proceso del servidor.
- Procese los datos del juego y de los jugadores según sea necesario. Esos datos se encuentran en el objeto `GameSession`.
- Informe al servicio Amazon GameLift Servers cuando haya una nueva sesión de juego lista para aceptar jugadores. Llame a la operación `ActivateGameSession()` de la API del servidor ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) . En respuesta a una llamada realizada correctamente, el servicio cambia el estado de la sesión de juego a `ACTIVE`.

Validación de un jugador nuevo (opcional)

Si está monitorizando el estado de las sesiones de jugador, añada código para validar a los nuevos jugadores cuando se conecten a un servidor de juegos. Amazon GameLift Servers hace un seguimiento de los jugadores actuales y de los espacios disponibles en las sesiones de juego.

Para la validación, un cliente de juego que intente unirse a una sesión de juego debe incluir un identificador de sesión de jugador. Amazon GameLift Servers genera este ID cuando el juego inicia nuevas sesiones de juego llamando a [StartGameSessionPlacement\(\)](#) o [StartMatchmaking\(\)](#). En el caso de estas solicitudes, se reserva un espacio libre en una sesión de juego para la sesión del jugador.

Cuando el proceso del servidor de juegos recibe una solicitud de conexión del cliente de juego, llama a `AcceptPlayerSession()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) con el ID de sesión de jugador. Como respuesta, Amazon GameLift Servers comprueba que el ID de sesión del jugador se corresponde con una ranura abierta reservada en la sesión de juego. Una vez que Amazon GameLift Servers valida el ID de sesión del jugador, el proceso del servidor acepta la conexión. A continuación, el jugador puede unirse a la sesión de juego. Si Amazon GameLift Servers no valida el ID de sesión del jugador, el proceso del servidor rechaza la conexión.

Notificación de la finalización de una sesión de jugador (opcional)

Si está realizando un seguimiento del estado de las sesiones de los jugadores, añada un código para informar a Amazon GameLift Servers cuando un jugador abandone la sesión de juego. Este código debería ejecutarse siempre que el proceso del servidor detecte una conexión interrumpida. Amazon GameLift Servers utiliza esta notificación para realizar un seguimiento de los jugadores actuales y de los espacios disponibles en la sesión de juego.

Para gestionar las conexiones interrumpidas, añada una llamada a la operación `RemovePlayerSession()` de la API del servidor ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) con el ID de sesión de jugador correspondiente.

Finalización de una sesión de juego

Añada código a la secuencia de cierre del proceso del servidor para informar a Amazon GameLift Servers cuando finalice una sesión de juego. Para reciclar y actualizar los recursos de alojamiento, cierre los procesos del servidor una vez finalizada la sesión de juego.

Al inicio del código de cierre del proceso del servidor, llame a la operación `ProcessEnding()` de la API del servidor ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) g . Esta llamada informa a Amazon GameLift Servers de que el proceso del servidor se está cerrando. Amazon GameLift Servers cambia el estado de la sesión de juego y del proceso del servidor a `TERMINATED`. Después de llamar a `ProcessEnding()`, es seguro que el proceso se cierre.

Respuesta a una notificación de cierre del proceso del servidor

Añada código para cerrar el proceso del servidor como respuesta a una notificación de Amazon GameLift Servers. El servicio envía esta notificación cuando el proceso del servidor informa constantemente de que no funciona correctamente o si la instancia en la que se está ejecutando el proceso del servidor está finalizando. Amazon GameLift Servers puede detener una instancia como parte de un evento de reducción vertical de la capacidad o como respuesta a una interrupción de una instancia de spot. La interrupción de una instancia de spot proporciona un aviso con dos minutos de antelación, lo que proporciona al proceso del servidor tiempo para desconectar a los jugadores de forma correcta, mantener los datos de estado del juego y realizar otras tareas de limpieza.

Para gestionar una notificación de cierre, realice los siguientes cambios en el código del servidor de juegos:

- Implemente la función de devolución de llamada `onProcessTerminate()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) . Esta función debe llamar al código que cierra el proceso del servidor.
- Llame a la operación de la API del servidor `GetTerminationTime()` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) desde el código de cierre del servidor de juegos. Si Amazon GameLift Servers ha emitido una llamada para detener el proceso del servidor, `GetTerminationTime()` devuelve el tiempo de finalización estimado.
- Al inicio del código de cierre del servidor de juegos, llame a la operación `ProcessEnding()` de la API del servidor ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) . Esta llamada notifica al servicio Amazon GameLift Servers que el proceso del servidor se va a apagar. A continuación, el servicio cambia el estado del proceso del servidor a `TERMINATED`. Después de llamar a `ProcessEnding()`, es seguro que el proceso se cierre.

Integración de SDK del servidor para motores de juegos

Puede utilizar el servicio de Amazon GameLift Servers administrado con la mayoría de motores de juegos compatibles con bibliotecas de C++ o C#, incluidos O3DE, Unreal Engine y Unity. Compile la versión que necesite para el juego; consulte los archivos README de cada versión para ver las instrucciones de compilación y conocer los requisitos mínimos. Para obtener más información sobre las plataformas de desarrollo y los sistemas operativos disponibles Amazon GameLift Servers SDKs y compatibles, consulta [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#) para servidores de juegos.

Además de la información específica del motor proporcionada en este tema, encontrará más ayuda para la integración de Amazon GameLift Servers en los servidores, clientes y servicios de juegos en los siguientes temas:

- [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#): instrucciones detalladas sobre la integración de Amazon GameLift Servers en un servidor de juegos.
- [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#): instrucciones detalladas sobre la integración en un cliente o servicio de juego, incluida la creación de sesiones de juego y la conexión de jugadores a juegos.

Unreal Engine

Servidores de juegos

Prepare los servidores de juegos para el alojamiento en Amazon GameLift Servers añadiendo el [SDK del servidor de Amazon GameLift Servers para Unreal Engine](#) al proyecto e implementando la funcionalidad de servidor necesaria. Para obtener ayuda para configurar el complemento Unreal Engine y añadir el código de Amazon GameLift Servers, consulte [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#).

Clientes y servicios de juego

Permita que los servicios de and/or juego de sus clientes de juegos interactúen con el Amazon GameLift Servers servicio, por ejemplo, para buscar sesiones de juego disponibles o crear nuevas, y añadir jugadores a los juegos. Consulte el [SDK de AWS para C++](#) para obtener información sobre las funcionalidades principales del cliente. Para integrar Amazon GameLift Servers en el proyecto de juego de Unreal Engine, consulte [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#).

Unity

Servidores de juegos

Prepare los servidores de juegos para el alojamiento en Amazon GameLift Servers agregando el [SDK del servidor de Amazon GameLift Servers para C# al proyecto](#) e implementando la funcionalidad de servidor necesaria. Para obtener ayuda sobre la configuración con Unity y la adición de código de Amazon GameLift Servers, consulte [Integración de Amazon GameLift Servers en un proyecto de Unity](#).

Clientes y servicios de juego

Permita que los servicios de and/or juego de sus clientes de juego interactúen con el Amazon GameLift Servers servicio, por ejemplo, para buscar sesiones de juego disponibles o crear nuevas, y añadir jugadores a los juegos. Consulte el [AWS SDK para .NET](#) para obtener información sobre las funcionalidades principales del cliente. Para integrar Amazon GameLift Servers en el proyecto de juego de Unity, consulte [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#).

Otros motores

Para ver una lista completa de los servidores y clientes de juegos Amazon GameLift Servers SDKs disponibles, consulte [the section called “Obtención de herramientas de desarrollo de ”](#).

Integración de Amazon GameLift Servers en un proyecto de Unreal Engine

Aprenda a integrar el SDK de Amazon GameLift Servers para Unreal Engine en sus proyectos de juegos para acceder al conjunto completo de características del SDK del servidor.

Sugerencia

Para agilizar la implementación, pruebe el complemento independiente de Amazon GameLift Servers para Unreal Engine. Proporciona flujos de trabajo de la IU guiados que le permitirán implementar rápidamente el servidor de juegos con una configuración mínima, para así poder probar los componentes del juego en acción. Consulte [Complemento de Amazon GameLift Servers para Unreal Engine](#).

Recursos adicionales:

- [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
- [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#)

Instalación del SDK del servidor para Unreal

Obtenga el Amazon GameLift Servers SDK de código abierto para Unreal Engine en. [GitHub](#) Los archivos readme del repositorio contienen los requisitos previos y las instrucciones de instalación.

Configuración de los destinos de compilación y las reglas de los módulos

Modifique los archivos del proyecto del juego para generar correctamente los componentes de compilación para usarlos con Amazon GameLift Servers.

Para añadir destinos de compilación del cliente y el servidor, siga estos pasos:

1. Abra los archivos de código del proyecto de juego y localice el archivo `.../Games/[your application name]Source/[your application name]Target.cs`. Ejemplo: `.../Source/GameLiftUnrealAppTarget.cs`. Si utiliza Visual Studio, abra el archivo `.sln` del proyecto.
2. Copie este archivo para crear dos nuevos archivos de destino en el directorio `Source/`.
 - Destino del cliente: cambie el nombre del nuevo archivo a `[your application name]Client.Target.cs`. Edite el contenido para actualizar el nombre de clase y los valores de tipo de destino, como se muestra en el siguiente ejemplo de código:

```
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" );
    }
}
```

- Destino del servidor: cambie el nombre del nuevo archivo a `[your application name]Server.Target.cs`. Edite el contenido para actualizar el nombre de clase y los valores de tipo de destino, como se muestra en el siguiente ejemplo de código:

```
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. Regenera los archivos del proyecto. Si utiliza Visual Studio, puede hacer clic con el botón derecho en el archivo `.uproject` del proyecto de juego y seleccionar Generar archivos de proyecto de Visual Studio.

Para actualizar las reglas del módulo del proyecto de juego:

Actualice las reglas del módulo del proyecto de juego para que dependan del complemento.

1. Abra los archivos de código del proyecto de juego y localice el archivo `.../Games/[your application name]Source/[your application name].Build.cs`. Ejemplo: `.../Source/GameLiftUnrealApp.Build.cs`. Si utiliza Visual Studio, abra el archivo `.sln` del proyecto.
2. Localice la clase `ModuleRules` y actualícela como se ilustra en el siguiente código de ejemplo:

```
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. Tras crear los nuevos archivos de destino y modificar las reglas del módulo, recompile su proyecto de juego.

Añada la funcionalidad de alojamiento de juegos al código de su servidor

Tras instalar y configurar el SDK del servidor, el siguiente paso es integrar la funcionalidad de alojamiento de juegos en el código del servidor. El SDK del servidor permite al servidor de juegos comunicarse con el servicio Amazon GameLift Servers, recibir instrucciones sobre las sesiones de juego, e informar sobre el estado general y de progreso, y realizar otras acciones.

En este tema se proporciona un ejemplo de código que añade la funcionalidad mínima necesaria para alojar el juego en Amazon GameLift Servers.

Paso 1: Actualizar el archivo de encabezado **GameMode**

1. Abra los archivos de código del proyecto de juego y localice el archivo *Your-application-name*GameMode.h. Ejemplo: GameLiftUnrealAppGameMode.h. Si utiliza Visual Studio, abra el archivo .sln del proyecto de juego.
2. Cambie el archivo de encabezado para incluir el siguiente código de ejemplo. Asegúrese de reemplazar "GameLiftUnrealApp" por el nombre de su propia aplicación.

Código de ejemplo para h GameMode.

```
// 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;
};
```

Paso 2: Añadir las llamadas del SDK del servidor necesarias al código del servidor de juegos

Utilice el código de ejemplo de esta sección para integrar el código del servidor de juegos y usarlo con Amazon GameLift Servers. Para obtener más información sobre los efectos de este código, consulte [Inicialización del proceso del servidor](#) y [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#).

Note

El indicador de preproceso `WITH_GAMELIFT` sirve para dos fines:

- Restringe las llamadas a la API de backend de Amazon GameLift Servers a solo las compilaciones de servidor de Unreal
- Garantiza la compatibilidad entre los diferentes destinos de compilación de Unreal

1. Abra el archivo ***Your-application-name***GameMode.cpp del archivo de origen relacionado. En nuestro ejemplo: `GameLiftUnrealAppGameMode.cpp`.
2. Cambie el código para alinearlo con el siguiente código de ejemplo. Asegúrese de reemplazar cualquier instancia de «GameLiftUnrealApp» por el nombre de su propia aplicación.

El ejemplo de código proporcionado muestra cómo añadir los elementos necesarios para la integración con Amazon GameLift Servers. Entre ellos se incluyen:

- Inicializar un cliente de API de Amazon GameLift Servers.

- Implemente funciones de devolución de llamada para responder a las solicitudes del servicio de Amazon GameLift Servers, incluidas `OnStartGameSession`, `OnProcessTerminate` y `onHealthCheck`.
- Llame a `ProcessReady()` para informar al servicio Amazon GameLift Servers cuando el proceso esté listo para alojar sesiones de juego.

Código de ejemplo para `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"

#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();

#if WITH_GAMELIFT
    InitGameLift();
#endif
}

void AGameLiftUnrealAppGameMode::InitGameLift()
{
#if 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 = FCString::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
}
```

Paso 3: Recompilar el proyecto del juego

- Compile un proyecto de juego para los dos tipos de destino siguientes: Editor de desarrollo y Servidor de desarrollo.

Note

No es necesario volver a compilar la solución. En su lugar, compile el proyecto en la carpeta /Games/ de su aplicación. De lo contrario, Visual Studio reconstruirá todo el UE5 proyecto, lo que puede tardar hasta una hora.

Empaquetado del servidor de juegos para su alojamiento

Ahora que el código del servidor de juegos está integrado con la funcionalidad mínima requerida del SDK del servidor, ya está listo para empaquetar la compilación del servidor de juegos con el editor de Unreal.

Empaquetado de la compilación del servidor de juegos

1. Abra el proyecto del juego en Unreal Editor.
2. Siga los pasos del editor de Unreal para empaquetar el servidor de juegos:
 - Elija la plataforma de destino (Windows o Linux).
 - Seleccione el destino de la compilación del servidor (*[your application name]*Server).

El proceso de empaquetado genera el ejecutable del servidor de juegos: *[your application name]*Server.exe.

3. Prepare la compilación del servidor de juegos para su implementación en los recursos de alojamiento. La compilación debe incluir los siguientes archivos.
 - El archivo ejecutable del servidor de juegos
 - Si está utilizando la versión 5.5 o anterior de Unreal Engine, incluya los siguientes archivos para las compilaciones de Windows. Los encontrará en su versión integrada de 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\)
 - Todas las demás dependencias necesarias para el servidor de juegos.
 - Bibliotecas de OpenSSL, si es necesario. Puede saltarse este paso si el servidor de juegos está integrado con la versión 5.3 o posterior del SDK del servidor de Amazon GameLift Servers. [La última versión del SDK del servidor está disponible aquí.](#)

Se requieren bibliotecas OpenSSL para los servidores de juegos creados con el SDK del servidor para Unreal (versión 5.2 y anteriores)

Debe incluir la misma versión de las bibliotecas OpenSSL que se usaron al empaquetar el servidor de juegos en Unreal. Estas bibliotecas se encuentran en el origen del motor de juegos. La ubicación varía en función del entorno de desarrollo:

En 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

En 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`

Copie las bibliotecas OpenSSL en el paquete de compilación del juego, en el mismo directorio que el archivo ejecutable del servidor de juegos.

Siguientes pasos

Ahora que ha preparado una compilación del servidor de juegos con la funcionalidad mínima necesaria para el alojamiento con Amazon GameLift Servers, considere la posibilidad de seguir estos pasos:

- Implemente el servidor de juegos integrado con fines de prueba y desarrollo. Con una flota de Anywhere, puede configurar su máquina local como recurso de alojamiento y utilizarla para probar las conexiones del servidor de juegos y los clientes de juegos. Para el alojamiento basado en la nube, implemente el servidor de juegos en una flota de EC2 administrada o en una flota de contenedores administrada. Consulte estos temas para obtener instrucciones:
 - [Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere](#)
 - [Flotas de Amazon GameLift Servers Anywhere](#)
 - [Amazon GameLift Servers EC2 flotas gestionadas](#)
 - [Flotas de contenedores administradas de Amazon GameLift Servers](#)
- Para personalizar la integración del servidor de juegos, añada características opcionales. Por ejemplo, puede que quieras añadir sesiones de jugadores con un jugador único IDs, configurar el sistema de reabastecimiento de partidas o gestionar el acceso del servidor del juego a tus otros AWS recursos (como una base de datos o un servicio de almacenamiento de contenido). Consulte estos temas para obtener instrucciones:
 - [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
 - [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
- Personalice el componente del cliente de juegos para solicitar sesiones de juego, recibir información de conexión y conectarse directamente a un servidor de juegos para jugar a un juego. Consulte estos temas para obtener instrucciones:
 - [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#)

Integración de Amazon GameLift Servers en un proyecto de Unity

Aprenda a integrar el SDK de Amazon GameLift Servers para Unity en sus proyectos de juegos para acceder al conjunto completo de características del SDK del servidor.

 Sugerencia

Para agilizar la implementación, pruebe el complemento independiente de Amazon GameLift Servers para Unity. Proporciona flujos de trabajo de la IU guiados que le permitirán implementar rápidamente el servidor de juegos con una configuración mínima, para así poder probar los componentes del juego en acción. Consulte [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

Recursos adicionales:

- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)
- [the section called “Obtención de herramientas de desarrollo de ”](#)

Instalación del SDK del servidor para Unity

Obtén el código abierto Amazon GameLift Servers de Unity en. [GitHub](#) Los archivos readme del repositorio contienen los requisitos previos y las instrucciones de instalación.

Configuración de una flota de Amazon GameLift Servers Anywhere con fines de prueba

Puede configurar su estación de trabajo de desarrollo como una flota de alojamiento de Amazon GameLift Servers Anywhere para probar de forma iterativa la integración con Amazon GameLift Servers. Con esta configuración, puede iniciar los procesos del servidor de juegos en su estación de trabajo, enviar solicitudes de unión o emparejamiento de jugadores a Amazon GameLift Servers para iniciar sesiones de juego y conectar clientes a las nuevas sesiones de juego. Con su propia estación de trabajo configurada como servidor de alojamiento, podrá supervisar todos los aspectos de la integración de los juegos con Amazon GameLift Servers.

Para obtener instrucciones sobre cómo configurar su estación de trabajo, consulte [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#) para completar los siguientes pasos:

1. Cree una ubicación personalizada para su estación de trabajo.
2. Cree una flota de Amazon GameLift Servers Anywhere con su nueva ubicación personalizada. Si se realiza correctamente, esta solicitud devuelve un ID de la flota. Tome nota de ese valor, ya que lo necesitará más tarde.
3. Registre su estación de trabajo como recurso de computación en la nueva flota de Anywhere. Proporcione un nombre de procesamiento único y especifique la dirección IP de su estación de

trabajo. Si se realiza correctamente, esta solicitud devuelve un punto final del SDK del servicio, en forma de WebSocket URL. Tome nota de ese valor, ya que lo necesitará más tarde.

4. Genere un token de autenticación para el procesamiento de su estación de trabajo. Esta autenticación de corta duración incluye el token y una fecha de caducidad. El servidor de juegos lo usa para autenticar la comunicación con el servicio de Amazon GameLift Servers. Guarde la autenticación en el recurso informático de su estación de trabajo para que los procesos del servidor de juegos en ejecución puedan acceder a él.

Adición del código de servidor de Amazon GameLift Servers al proyecto

El servidor de juegos se comunica con el servicio de Amazon GameLift Servers para recibir instrucciones e informar sobre su estado actual. Para ello, debe añadir un código de servidor de juegos que utilice el SDK del servidor de Amazon GameLift Servers.

El ejemplo de código proporcionado muestra los elementos básicos de integración necesarios. Utiliza un `MonoBehavior` para ilustrar una inicialización sencilla de un servidor de juegos con Amazon GameLift Servers. En el ejemplo, se presupone que el servidor de juegos se ejecuta en una flota de Amazon GameLift Servers con fines de prueba. Incluye código para lo siguiente:

- Inicializar un cliente de API de Amazon GameLift Servers. En el ejemplo, se utiliza la versión de `InitSDK()` con parámetros de servidor los recursos de computación y la flota de Anywhere. Usa la WebSocket URL, el ID de flota, el nombre de proceso (ID de host) y el token de autenticación, tal y como se definió en el tema anterior [Configuración de una flota de Amazon GameLift Servers Anywhere con fines de prueba](#).
- Implemente funciones de devolución de llamada para responder a las solicitudes del servicio de Amazon GameLift Servers, incluidas `OnStartGameSession`, `OnProcessTerminate` y `onHealthCheck`.
- Llama a `ProcessReady()` con un puerto designado para notificar al Amazon GameLift Servers servicio cuando el proceso esté listo para albergar sesiones de juego.

El código de muestra que se proporciona establece la comunicación con el servicio de Amazon GameLift Servers. También implementa un conjunto de funciones de devolución de llamada que responden a las solicitudes procedentes del servicio de Amazon GameLift Servers. Para obtener más información sobre cada función y lo que hace el código, consulte [Inicialización del proceso del servidor](#). Para obtener más información sobre las acciones del SDK y los tipos de datos que

se utilizan en este código, consulte [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#).

El código de muestra indica cómo añadir la funcionalidad necesaria, tal y como se describe en [Adición de Amazon GameLift Servers al servidor de juegos](#). Para obtener más información sobre las acciones del SDK del servidor, consulte [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#).

Código de muestra para la integración

```
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);  
    }  
}  
}
```

Siguientes pasos

Ahora que ha preparado una compilación del servidor de juegos con la funcionalidad mínima necesaria para el alojamiento con Amazon GameLift Servers, considere la posibilidad de seguir estos pasos:

- Implemente el servidor de juegos integrado con fines de prueba y desarrollo. Con una flota de Anywhere, puede configurar su máquina local como recurso de alojamiento y utilizarla para probar las conexiones del servidor de juegos y los clientes de juegos. Para el alojamiento basado en la nube, implemente el servidor de juegos en una flota de EC2 administrada o en una flota de contenedores administrada. Consulte estos temas para obtener instrucciones:
 - [Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere](#)
 - [Flotas de Amazon GameLift Servers Anywhere](#)
 - [Amazon GameLift Servers EC2 flotas gestionadas](#)
 - [Flotas de contenedores administradas de Amazon GameLift Servers](#)

- Para personalizar la integración del servidor de juegos, añada características opcionales. Por ejemplo, es posible que desees añadir sesiones de jugadores con un jugador único IDs, configurar el sistema de reabastecimiento de partidas o gestionar el acceso del servidor del juego a tus otros AWS recursos (como una base de datos o un servicio de almacenamiento de contenido). Consulte estos temas para obtener instrucciones:
 - [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
 - [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
- Personalice el componente del cliente de juegos para solicitar sesiones de juego, recibir información de conexión y conectarse directamente a un servidor de juegos para jugar a un juego. Consulte estos temas para obtener instrucciones:
 - [Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers](#)

Empaquetado de una compilación de servidor de juegos para su implementación

Prepare el software de servidor de juegos para la implementación en recursos de alojamiento. Una vez implementado, el software se instala en cada recurso de alojamiento y, a continuación, se inician uno o más procesos del servidor de juegos y se preparan para alojar las sesiones de juego de los jugadores.

La preparación de una compilación de servidor de juegos varía en función del tipo de opciones de alojamiento de Amazon GameLift Servers que se esté utilizando. Todas las compilaciones de servidor de juegos deben estar integradas con el SDK del servidor de Amazon GameLift Servers, tal como se describe en [Integración de un servidor de juegos con Amazon GameLift Servers](#).

En los temas de esta sección, se proporcionan instrucciones sobre cómo preparar el software para la implementación en los siguientes escenarios.

- Para el EC2 alojamiento gestionado, empaquete el software de su servidor y cárguelo Amazon GameLift Servers para su implementación.
- Para el alojamiento de contenedores administrados, cree una imagen de contenedor con el software de servidor y guárdela en Amazon Elastic Container Registry para su implementación.
- Para el alojamiento con Amazon GameLift Servers Anywhere, empaquete el software de servidor según corresponda para instalarlo en sus propios recursos de alojamiento.

 Note

Si está implementando un script configurado de Amazon GameLift Servers Realtime, consulte [Carga de un script para Amazon GameLift Servers Realtime](#).

Temas

- [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#)
- [Creación de una imagen de contenedor para Amazon GameLift Servers](#)

Creación de una compilación de servidor de juegos para Amazon GameLift Servers

Después de integrar el servidor de juegos con Amazon GameLift Servers (consulte [Preparación de un juego para su alojamiento con Amazon GameLift Servers](#)), instale el software del servidor de juegos en los recursos de computación para el alojamiento. Este proceso varía en función del tipo de alojamiento de Amazon GameLift Servers que esté usando.

Implementación para el alojamiento administrado

Si utiliza el alojamiento de EC2 administrado por Amazon GameLift Servers, debe empaquetar el software del servidor de juegos y cargarlo en Amazon GameLift Servers. Al crear una flota administrada, Amazon GameLift Servers la implementa automáticamente en cada instancia de la flota.

En los temas de esta sección se describe cómo empaquetar los archivos de compilación para cargarlos, cómo crear un script de instalación de compilación opcional y, a continuación, cómo cargar los archivos mediante [AWS Command Line Interface \(AWS CLI\)](#) o el AWS SDK.

Implementación para el alojamiento en Anywhere

Si utiliza flotas de Amazon GameLift Servers Anywhere para el alojamiento autoadministrado, deberá instalar el software del servidor de juegos en cada uno de los recursos de computación de la flota y mantenerlo actualizado.

Cuando se empieza a ejecutar un proceso del servidor de juegos integrado, se inicializa automáticamente y establece la comunicación con el servicio Amazon GameLift Servers. El proceso del servidor inicia las sesiones de juego a partir de las peticiones de Amazon GameLift Servers y le devuelve al servicio información sobre la actividad.

Temas

- [Empaquetado de los archivos de compilación del juego](#)
- [Agregar un script de instalación de la compilación](#)
- [Cree un recurso de Amazon GameLift Servers compilación para el alojamiento gestionado](#)

Empaquetado de los archivos de compilación del juego

Antes de cargar el servidor de juegos configurado en Amazon GameLift Servers, empaquete los archivos de compilación del juego en un directorio de compilación. Este proceso es obligatorio cuando el alojamiento se lleva a cabo con una flota de EC2 administrada, y es la práctica recomendada al hacer el alojamiento con una flota de Anywhere. El directorio de compilación debe incluir todos los componentes necesarios para ejecutar los servidores de juegos y alojar las sesiones de juego. Esto podría incluir lo siguiente:

- Archivos binarios del servidor de juegos: son los archivos binarios necesarios para ejecutar el servidor de juegos. Una compilación puede incluir archivos binarios para varios servidores de juegos diseñados para ejecutarlos en la misma plataforma. Para obtener una lista de plataformas admitidas, consulte [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#).
- Dependencias: cualquier archivo dependiente necesario para que los archivos ejecutables del servidor de juegos funcione. Por ejemplo, activos, archivos de configuración y bibliotecas dependientes.

Note

En el caso de compilaciones de juegos creadas con el SDK del servidor Amazon GameLift Servers para C++ (incluidas las creadas con el complemento de Unreal), incluya el DLL de OpenSSL para la misma versión de OpenSSL con la que creó el SDK del servidor. Consulte el archivo README del SDK del servidor para obtener más información.

- Script de instalación (opcional): archivo de script para administrar las tareas de instalación de la compilación del juego en los servidores de alojamiento de Amazon GameLift Servers. Coloque ese archivo en la raíz del directorio de compilación. Amazon GameLift Servers ejecuta el script de instalación como parte de la creación de la flota.

Puedes configurar cualquier aplicación de tu compilación, incluido el script de instalación, para acceder a tus recursos de forma segura en otros AWS servicios. Para obtener información sobre

cómo hacerlo, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Después de empaquetar los archivos de compilación, asegúrese de que el servidor de juegos pueda ejecutarse en una instalación limpia de su sistema operativo de destino, a fin de verificar que se hayan incluido todas las dependencias obligatorias y que el script de la instalación sea preciso.

Agregar un script de instalación de la compilación

Cree un script de instalación para el sistema operativo (SO) de la compilación del juego:

- Windows: cree un archivo de procesamiento por lotes denominado `install.bat`.
- Linux: cree un archivo de script de intérprete de comandos denominado `install.sh`.

Al crear un script de instalación, tenga en cuenta lo siguiente:

- El script no puede aceptar ninguna entrada por parte del usuario.
- Amazon GameLift Servers instala la compilación y vuelve a crear los directorios de archivos del paquete de compilación en un servidor de alojamiento en las siguientes ubicaciones:
 - Flotas de Windows: `C:\game`
 - Flotas de Linux: `/local/game`
- Durante el proceso de instalación para flotas de Linux, el usuario que realiza la ejecución tiene acceso limitado a la estructura de archivos de la instancia. Dicho usuario dispone de acceso completo al directorio en el que se instalan los archivos de compilación. Si el script de instalación realiza acciones que requieren permisos de administrador, especifique el acceso de administrador mediante `sudo`. El usuario en ejecución para las flotas de Windows dispone de permisos de administrador de forma predeterminada. Los errores de permisos relacionados con el script de instalación generan un mensaje de evento que indica un problema con el script.
- En Linux, Amazon GameLift Servers admite lenguajes comunes de intérprete de comandos, como `bash`. Añada un shebang (como `#!/bin/bash`) al principio del script de instalación. Para verificar la compatibilidad con los comandos de intérprete de comandos de su preferencia, inicie una sesión remota en una instancia de Linux activa y abra una petición del intérprete de comandos. Para obtener más información, consulte [Conexión remota a instancias de flota de Amazon GameLift Servers](#).
- El script de instalación no puede confiar en una conexión de emparejamiento de VPC. La conexión de emparejamiento de VPC no estará disponible hasta que Amazon GameLift Servers instale la compilación en las instancias de la flota.

Example Archivo bash de instalación de Windows

En este ejemplo, el archivo `install.bat` instala los componentes del tiempo de ejecución de Visual C++ necesarios para el servidor de juegos y escribe los resultados en un archivo de registro. El script incluye el archivo de componente en el paquete de compilación en la raíz.

```
vcredist_x64.exe /install /quiet /norestart /log c:\game\vc_redist_2013_x64.log
```

Example Script de intérprete de comandos de instalación de Linux

Este archivo `install.sh` de ejemplo utiliza bash en el script de instalación y escribe los resultados en un archivo de registro.

```
#!/bin/bash
echo 'Hello World' > install.log
```

Este `install.sh` archivo de ejemplo muestra cómo puedes usar el CloudWatch agente de Amazon para recopilar métricas personalizadas y a nivel del sistema y gestionar la rotación de registros. Puesto que Amazon GameLift Servers se ejecuta en una VPC de servicio, debe conceder permisos a Amazon GameLift Servers para que asuma un rol de AWS Identity and Access Management (IAM) en su nombre. Amazon GameLift ServersPara poder asumir un rol, crea un rol que incluya la política AWS CloudWatchAgentAdminPolicy gestionada y utilízalo cuando crees una flota.

```
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

```

Cree un recurso de Amazon GameLift Servers compilación para el alojamiento gestionado

Al crear una compilación y cargar sus archivos, tiene dos opciones:

- [Creación de una compilación a partir de un directorio de archivos](#). Esta es la opción más sencilla y la que se utiliza habitualmente.
- [Cree una compilación con archivos en Amazon Simple Storage Service \(Amazon S3\)](#). Con esta opción, puede administrar sus versiones de compilación en Amazon S3.

Con ambos métodos, Amazon GameLift Servers crea un nuevo recurso de compilación con un ID de compilación único y otros metadatos. La compilación comienza con el estado Inicializado. Cuando Amazon GameLift Servers adquiere los archivos del servidor de juegos, la compilación pasa al estado Listo.

Cuando la compilación esté lista, podrá implementarla en una nueva flota de Amazon GameLift Servers. Para obtener más información, consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#). Cuando Amazon GameLift Servers configure la nueva flota, descargará los archivos de compilación en cada instancia de la flota e instalará los archivos de compilación.

Creación de una compilación a partir de un directorio de archivos

Para crear una compilación de juego almacenada en cualquier ubicación, incluido un directorio local, usa el [upload-build](#) AWS CLI comando. Este comando crea un registro de compilación nuevo en Amazon GameLift Servers y carga archivos desde la ubicación que especifique.

Envíe una solicitud de carga. En una ventana de línea de comandos, especifique el comando upload-build y los parámetros siguientes.

```
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:** el entorno de tiempo de ejecución de la compilación del servidor de juegos. Debe especificar un valor para el sistema operativo. El valor no podrá actualizarse más tarde.
- **server-sdk-version:** la versión del SDK del servidor de Amazon GameLift Servers con la que se ha integrado el servidor de juegos. Si no proporciona un valor, Amazon GameLift Servers utiliza el valor por defecto 4.0.2. Si especifica una versión incorrecta del SDK del servidor, es posible que la compilación del servidor de juegos falle al llamar a `InitSdk` para establecer una conexión con el servicio Amazon GameLift Servers.
- **build-root:** es la ruta del directorio de los archivos de compilación.
- **name:** es un nombre descriptivo para la nueva compilación.
- **build-version:** los detalles de la versión de los archivos de compilación.
- **region—** La AWS región en la que quieres crear tu construcción. Cree la compilación en la región en la que tiene previsto implementar las flotas. Si va a implementar el juego en varias regiones, cree una compilación en cada región.

 Note

Vea su región predeterminada actual mediante el comando [aws configure get region](#). Utilice el comando [aws configure set region *region name*](#) para cambiar la región predeterminada.

Ejemplos

```
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
```

```
aws gamelift upload-build \  
  --operating-system WINDOWS_2022 \  
  --server-sdk-version "5.0.0" \  
  --build-root "C:\mygame" \  
  --name "My Game Nightly Build" \  
  --build-version "build 255" \  
  --region us-west-2
```

Como respuesta a su solicitud de carga, Amazon GameLift Servers proporciona el progreso de la carga. Si la carga se realiza correctamente, Amazon GameLift Servers devuelve el nuevo ID de registro de compilación. El tiempo de carga depende del tamaño de los archivos del juego y de la velocidad de conexión.

Creación de una compilación con archivos en Amazon S3

Puede almacenar sus archivos de compilación en Amazon S3 y cargarlos en Amazon GameLift Servers desde allí. Al crear la compilación, debe especificar la ubicación del bucket de S3 y Amazon GameLift Servers recuperará los archivos de compilación directamente desde Amazon S3.

Para crear un recurso de compilación, realice el siguiente procedimiento:

1. Almacene los archivos de compilación en Amazon S3. Cree un archivo.zip que contenga los archivos de compilación empaquetados y cárguelo en un bucket de S3 que tenga Cuenta de AWS. Toma nota de la etiqueta del depósito y del nombre del archivo, ya que los necesitarás para crear una Amazon GameLift Servers compilación.
2. Otorgue acceso a Amazon GameLift Servers a los archivos de compilación. Para crear un rol de IAM, siga las instrucciones en [Acceso a un archivo de compilación de un juego en Amazon S3](#). En cuanto haya creado el rol, tome nota del nombre de recurso de Amazon (ARN) del nuevo rol, ya que lo necesitará para crear una compilación.
3. Cree una compilación. Usa la Amazon GameLift Servers consola o el AWS CLI para crear un nuevo registro de compilación. Debe tener el permiso `PassRole`, tal y como se describe en [Ejemplo de permisos de IAM para Amazon GameLift Servers](#).

Console

1. En la [consola de Amazon GameLift Servers](#), en el panel de navegación, elija Alojamiento, Compilaciones.
2. En la página Compilaciones, seleccione Crear compilación.
3. En la página Crear compilación, en Configuración de compilación, realice el siguiente procedimiento:
 - a. En Nombre, especifique un nombre de script.
 - b. En Versión, escriba una versión. Como puede actualizar el contenido de una compilación, los datos de la versión pueden ayudarle a realizar un seguimiento de las actualizaciones.
 - c. Para Sistema operativo (SO), elija el sistema operativo de la compilación del servidor de juegos. El valor no podrá actualizarse más tarde.
 - d. En Compilación del servidor de juegos, especifique el URI de S3 del objeto de compilación que cargó en Amazon S3 y elija la versión del objeto. Si no recuerda el URI y la versión del objeto de Amazon S3, elija Explorar S3 y busque el objeto de compilación.
 - e. Para el rol de IAM, elija el rol creado que permite a Amazon GameLift Servers acceder a su bucket de S3 y al objeto de compilación.

4. En Etiquetas, añada etiquetas a la compilación introduciendo los pares Clave y Valor (opcional).
5. Seleccione Crear.

Amazon GameLift Servers asigna un ID a la nueva compilación y carga el archivo .zip designado. Puede ver la nueva compilación, incluido su estado, en la página Compilaciones

AWS CLI

Utilice el comando [create-build](#) para definir la nueva compilación y cargar los archivos de compilación del servidor.

1. Abra una ventana de línea de comandos y cambie a un directorio en el que pueda usar la AWS CLI.
2. Escriba el siguiente comando create-build:

```
aws gamelift create-build \  
  --name user-defined name of build \  
  --server-sdk-version server SDK for Amazon GameLift Servers version \  
  --operating-system supported OS \  
  --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: es un nombre descriptivo para la nueva compilación.
- server-sdk-version: la versión del SDK del servidor de Amazon GameLift Servers utilizado para integrar el servidor de juegos con Amazon GameLift Servers. Si no proporciona un valor, Amazon GameLift Servers utiliza el valor por defecto 4.0.2.
- operating-system: el entorno de tiempo de ejecución de la compilación del servidor de juegos. Debe especificar un valor para el sistema operativo. El valor no podrá actualizarse más tarde.
- build-version: los detalles de la versión de los archivos de compilación. Esa información puede resultar útil porque cada nueva versión del servidor de juegos requiere un nuevo recurso de compilación.
- storage-location

- **Bucket:** nombre del bucket de S3 que contiene la compilación. Por ejemplo, "my_build_files".
- **Key:** nombre del archivo .zip que contiene los archivos de compilación. Por ejemplo, "my_game_build_7.0.1, 7.0.2".
- **RoleARN:** el ARN asignado al rol de IAM creado. Por ejemplo, «arn:aws:iam: :111122223333:role/». GameLiftAccess Para ver una política de ejemplo, consulte [Acceso a un archivo de compilación de un juego en Amazon S3](#).
- **region—** Crea la construcción en la región en la que planeas desplegar las flotas. AWS Si va a implementar el juego en varias regiones, cree una compilación en cada región.

 Note

Le recomendamos que compruebe la región predeterminada actual mediante el comando [configure get](#). Utilice el comando [configure set](#) para cambiar la región predeterminada.

Ejemplo

```
aws gamelift create-build \  
  --operating-system WINDOWS_2022 \  
  --storage-location  
  "Bucket"="my_game_build_files", "Key"="mygame_build_101.zip", "RoleArn"="arn:aws:iam: :111122223333:role/  
gamelift" \  
  --name "My Game Nightly Build" \  
  --build-version "build 101" \  
  --region us-west-2
```

3. Para ver la nueva compilación, utilice el comando [describe-build](#).

Creación de una imagen de contenedor para Amazon GameLift Servers

En este tema se describe cómo crear una imagen de contenedor con el software de servidor de juegos para su uso con Amazon GameLift Servers. Una imagen de contenedor de servidor de juegos incluye el ejecutable del servidor de juegos, así como cualquier dependencia que este necesite para ejecutarse. Las imágenes de contenedor de servidor de juegos se utilizan con una solución de alojamiento de contenedores administrados de Amazon GameLift Servers. Para obtener más información sobre cómo crear la solución completa, consulte:

- [Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers](#)
- [Cómo funcionan los contenedores en Amazon GameLift Servers](#)

Complete las siguientes tareas para preparar la imagen de contenedor del servidor de juegos para su implementación en una flota de contenedores de Amazon GameLift Servers. Antes de iniciar estas tareas, termine de integrar el código del servidor de juegos con el SDK del servidor de Amazon GameLift Servers.

Temas

- [Creación de una imagen de contenedor del servidor de juegos](#)
- [Inserción de una imagen de contenedor en Amazon ECR](#)

Creación de una imagen de contenedor del servidor de juegos

Siga estas instrucciones cuando trabaje en una plataforma basada en Linux o utilice Windows Subsystem para Linux (WSL) con Docker instalado.

Creación de una imagen de contenedor del servidor de juegos

1. Prepare un directorio de trabajo con el software de servidor de juegos. En una máquina local, cree un directorio de trabajo para organizar los archivos del contenedor del servidor de juegos. La imagen de contenedor utiliza esta estructura de archivos al implementar el contenedor en los recursos de Amazon GameLift Servers para el alojamiento. Por ejemplo:

```
[~/]$ mkdir -p work/glc/gamebuild && cd work && find .  
.  
./glc  
./glc/gamebuild
```

Note

Si estás probando esta función y aún no tienes una versión de servidor de juegos que funcione, prueba nuestro servidor de juegos de muestra [SimpleServer](#), que está disponible en GitHub.

2. Cree un nuevo archivo Dockerfile utilizando la plantilla proporcionada.

3. Siga las instrucciones de la plantilla de Dockerfile para actualizarla para su propio uso.
 - Actualice la imagen base como corresponda.
 - Establezca variables de entorno para la compilación del servidor de juegos.
4. Compile la imagen de contenedor. Ejecute la `docker build` especificando el nombre de su propio repositorio. Por ejemplo:

```
[~/work/glc]$ docker build -t <local repository name>:<optional tag> .
```

Puedes ver tus repositorios e imágenes IDs con el `docker images` comando, tal y como se muestra en este ejemplo:

Plantilla de Dockerfile para una imagen de contenedor de servidor de juegos

Esta plantilla contiene las instrucciones mínimas que necesita un contenedor de servidor de juegos para poder utilizarse en una flota de Amazon GameLift Servers. Modifique el contenido como sea necesario para el servidor de juegos.

```
# 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")]
```

Inserción de una imagen de contenedor en Amazon ECR

Una vez que haya creado una imagen de contenedor para su implementación en Amazon GameLift Servers, almacene la imagen en un repositorio público o privado de Amazon ECR. A este repositorio se le asigna un valor de URI, que Amazon GameLift Servers utiliza para tomar una instantánea de la imagen para la implementación en una flota de contenedores.

Note

Si aún no tiene un repositorio privado de Amazon ECR, debe [crear uno](#).

Inserción de la imagen de contenedor en Amazon ECR

1. Obtenga sus credenciales de Amazon ECR. Antes de insertar una imagen de contenedor en Amazon ECR, primero debe adquirir sus credenciales de AWS de forma temporal y proporcionárselas a Docker. Docker necesita estas credenciales para iniciar sesión.

```
[~/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. Copie el URI del [repositorio privado de Amazon ECR](#) que desea utilizar.
3. Aplique una etiqueta de Amazon ECR a la imagen de contenedor.

Example

```
[~/work/glc]$ docker tag <IMAGE ID from above> <Amazon ECR private repository  
URI>:<optional tag>
```

4. Inserción de la imagen de contenedor en Amazon ECR

Example

```
[~/work/glc]$ docker image push <Amazon ECR private repository URI>
```

client/server Interacciones del juego con Amazon GameLift Servers

Los componentes de la solución de alojamiento de Amazon GameLift Servers interactúan entre sí de formas específicas para ejecutar sesiones de juego en respuesta a la demanda de jugadores. En este tema, se describe cómo se comunican los componentes entre sí cuando el servidor de juegos

está alojado en flotas de EC2 administradas por Amazon GameLift Servers, en flotas de Amazon GameLift Servers Anywhere autoadministradas o en una solución híbrida.

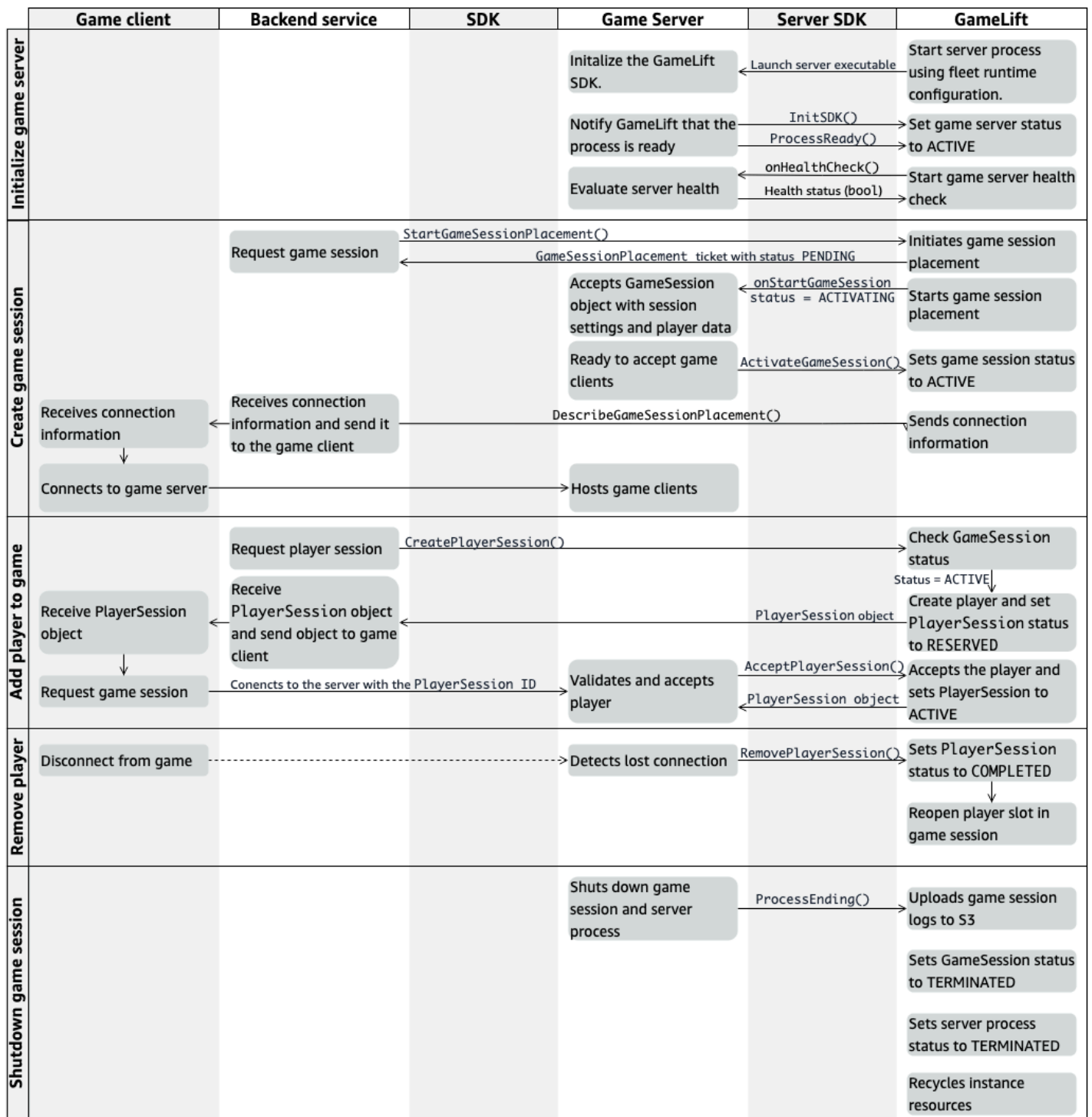
Algunos de los componentes de la solución de alojamiento son un servidor de juegos, el servicio de Amazon GameLift Servers, un servicio de backend en el cliente y un cliente de juegos. El servidor de juegos usa el [SDK del servidor de Amazon GameLift Servers](#) para interactuar con el servicio Amazon GameLift Servers. El servicio de backend utiliza la [API del Amazon GameLift Servers servicio](#) (parte del AWS SDK) para interactuar con el servicio en nombre del cliente del juego. Al unirse a una sesión de juego, el cliente de juego se conecta directamente a una sesión de juego mediante la dirección IP y el número de puerto exclusivos de la sesión de juego.

Temas

- [Diagrama de interacciones](#)
- [Comportamientos de interacción](#)

Diagrama de interacciones

El siguiente diagrama ilustra cómo interactúan los componentes de alojamiento del juego para que el servicio Amazon GameLift Servers pueda hacer un seguimiento del estado de la disponibilidad del servidor de juegos e iniciar sesiones de juego en respuesta a las demandas de jugadores.



Comportamientos de interacción

En las siguientes secciones, se describe la secuencia de eventos en cada una de las interacciones principales.

Inicialización de un proceso de servidor de juegos

En el arranque, un proceso de servidor de juegos establece la comunicación con el servicio Amazon GameLift Servers e indica que está preparado para alojar una sesión de juego.

1. Un nuevo proceso del ejecutable del servidor de juegos comienza a ejecutarse en un recurso de alojamiento.
2. El proceso del servidor de juego llama a las siguientes operaciones del SDK del servidor de forma secuencial:
 - a. `InitSDK()` para inicializar el SDK del servidor, autenticar el proceso del servidor y establecer la comunicación con el servicio Amazon GameLift Servers.
 - b. `ProcessReady()`, para comunicar que todo está listo para alojar una sesión de juego. Esta llamada también muestra los datos de conexión del proceso, cuál es el juego que los clientes utilizan para conectarse a la sesión de juego, etc.

A continuación, el proceso del servidor espera a que lleguen peticiones del servicio Amazon GameLift Servers.

3. Amazon GameLift Servers actualiza el estado del proceso del servidor a `ACTIVE` y disponible para alojar una nueva sesión de juego.
4. Amazon GameLift Servers comienza a hacer de forma periódica la devolución de llamada `onHealthCheck` para solicitar el estado de los procesos del servidor. Estas llamadas continúan mientras el proceso del servidor permanece activo. En menos de un minuto, el proceso del servidor debe indicar si se encuentra o no en buen estado. Si el proceso del servidor indica que no se encuentra en buen estado o no responde, el servicio Amazon GameLift Servers cambiará, en algún momento, el estado activo del proceso del servidor, y dejará de enviar solicitudes para iniciar la sesión de juego.

Creación de una sesión de juego

El servicio Amazon GameLift Servers inicia una nueva sesión de juego en respuesta a una solicitud de un jugador para empezar a jugar.

1. Un jugador que usa el cliente de juego solicita unirse a una sesión de juego. En función de cómo gestione el juego el proceso de incorporación de un jugador, el cliente de juego le envía una solicitud al servicio de backend.

2. Si el proceso de ingreso de un jugador requiere iniciar una nueva sesión de juego, el servicio de backend envía una solicitud de nueva sesión de juego al servicio Amazon GameLift Servers. Esta solicitud llama a la operación de API de servicio `StartGameSessionPlacement()` (otra opción es que el servicio de backend llame a `StartMatchmaking()` o `CreateGameSession()`).
3. El servicio Amazon GameLift Servers responde creando un nuevo ticket `GameSessionPlacement` con el estado `PENDING`. Devuelve la información de ticket al servicio de backend, para que este pueda hacer un seguimiento del estado del ticket de ubicación y determinar el momento en el que la sesión de juego esté lista para los jugadores. Para obtener más información, consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego.](#)
4. El servicio Amazon GameLift Servers inicia el proceso de ubicación de la sesión de juego. Identifica cuáles son las flotas que se deben examinar, y busca en esas flotas un proceso de servidor activo que no esté alojando una sesión de juego. Al encontrar un proceso de servidor disponible, el servicio Amazon GameLift Servers hace lo siguiente:
 - a. Crea un objeto `GameSession` con la configuración de la sesión de juego y los datos del jugador a partir de la solicitud de ubicación, y establece el estado en `ACTIVATING`.
 - b. Le pide al proceso del servidor que inicie una sesión de juego. El servicio invoca la devolución de llamada `onStartGameSession` del proceso del servidor y pasa el objeto `GameSession`.
 - c. Cambia el número de sesiones de juego del proceso del servidor a 1.
5. El proceso del servidor ejecuta la función de devolución de llamada `onStartGameSession`. Cuando el proceso del servidor está listo para aceptar las conexiones de jugadores, llama a la operación del SDK del servidor `ActivateGameSession()` y espera a que haya conexiones de jugadores.
6. El servicio Amazon GameLift Servers actualiza el objeto `GameSession` con la información de conexión para el proceso del servidor (tal y como se indica en la llamada a `ProcessReady()`) y establece el estado de la sesión de juego en `ACTIVE`. También actualiza el estado del ticket `GameSessionPlacement` en `FULFILLED`.
7. El servicio de backend llama a `DescribeGameSessionPlacement()` para comprobar el estado del ticket y obtener información sobre la sesión de juego. Cuando la sesión de juego está activa, el servicio de backend notifica al cliente de juego y transmite los datos de conexión de la sesión de juego.

8. El cliente de juego utiliza la información de conexión para conectarse directamente con el proceso del servidor de juegos y unirse a la sesión de juego.

Adición de un jugador a un juego

Como característica opcional, un juego puede usar las sesiones de los jugadores para hacer un seguimiento de las conexiones de los jugadores a sesiones de juego. Las sesiones de jugador se pueden crear de forma individual o como parte de una solicitud de ubicación de sesión de juego.

1. El servicio de backend llama a la operación de la API del servicio `CreatePlayerSession()` con un ID de sesión de juego.
2. El servicio Amazon GameLift Servers comprueba el estado de la sesión de juego (debe ser `ACTIVE`) y busca un espacio de jugador abierto en la sesión de juego. Si hay un espacio disponible, el servicio hace lo siguiente:
 - a. Crea un objeto `PlayerSession` nuevo y establece el estado en `RESERVED`.
 - b. Responde a la solicitud de servicio de backend con información de la sesión del jugador.
3. El servicio de backend pasa la información de la sesión del jugador al cliente de juego junto con los datos de conexión de la sesión de juego.
4. El cliente de juego utiliza los datos de conexión y el ID de sesión de jugador para conectarse directamente al proceso del servidor de juegos y unirse a la sesión de juego.
5. Cuando un cliente de juego intenta unirse, el proceso del servidor del juego llama a la operación `AcceptPlayerSession()` de la API del servicio para validar el ID de sesión del jugador. A continuación, el proceso del servidor acepta o rechaza la conexión.
6. El Amazon GameLift Servers servicio realiza una de las siguientes acciones:
 - a. Si se acepta la conexión, Amazon GameLift Servers establece el estado de `PlayerSession` en `ACTIVE` y transfiere la `PlayerSession` al proceso del servidor de juegos.
 - b. Si el proceso del servidor de juego no llama a `AcceptPlayerSession()` para el ID de sesión de jugador dentro de un período de tiempo determinado después de la solicitud `CreatePlayerSession()` original, el servicio Amazon GameLift Servers cambia el estado `PlayerSession` a `TIMEDOUT` y vuelve a abrir el espacio para jugadores en la sesión de juego.

Eliminación de un jugador

En el caso de los juegos que utilizan sesiones de jugadores, el proceso del servidor de juegos notifica al servicio Amazon GameLift Servers cuando un jugador se desconecta. El servicio utiliza esta información para hacer un seguimiento del estado de los espacios para jugadores en una sesión de juego, y puede permitir la entrada de nuevos jugadores en los espacios disponibles.

1. Un jugador se desconecta de la sesión de juego.
2. El proceso del servidor de juego detecta la conexión perdida y llama a la operación `RemovePlayerSession()` del SDK del servidor.
3. El servicio Amazon GameLift Servers cambia el estado de la sesión del jugador a `COMPLETED` y vuelve a abrir el espacio para jugadores en la sesión de juego.

Cierre de la sesión de juego

Al final de una sesión de juego, o al cerrar dicha sesión, el proceso del servidor notifica al servicio Amazon GameLift Servers el estado de la sesión de juego.

1. El proceso del servidor de juego finaliza la sesión de juego e inicia el cierre del proceso mediante una llamada a la operación `ProcessEnding()` del SDK del servidor.
2. El proceso del servidor del juego realiza llamadas `Destroy()` para liberar el SDK del servidor de la memoria. Este paso es necesario para que [las métricas de telemetría impidan que las](#) salidas normales del proceso se detecten como bloqueos.
3. El Amazon GameLift Servers servicio hace lo siguiente:
 - a. Carga registros de sesión de juego en Amazon Simple Storage Service (Amazon S3).
 - b. Cambia el estado de la sesión del juego a `TERMINATED`.
 - c. Cambia el estado del proceso del servidor a `TERMINATED`.
 - d. Según el diseño de la solución de alojamiento, los nuevos recursos de alojamiento disponibles se asignan para ejecutar un nuevo proceso de servidor de juegos.

Creación de un servicio de backend para Amazon GameLift Servers

Le recomendamos que implemente un servicio de cliente de juegos que autentique a sus jugadores y se comuniquen con la API de Amazon GameLift Servers. Al implementar un servicio de cliente de juegos personalizado, podrá realizar las siguientes acciones:

- Personalizar la autenticación de sus jugadores..
- Controle el modo en que Amazon GameLift Servers agrupa a los jugadores para las nuevas sesiones de juego o para añadirlos a sesiones de juego existentes.
- Recopile información de sus propios recursos para proporcionar al juego atributos de los jugadores tales como el nivel de habilidad para realizar emparejamientos en lugar de confiar en el cliente.

Si se usa un servicio de cliente de juegos, también se reducen los riesgos de seguridad que presentan los clientes de juegos que interactúan directamente con la API de Amazon GameLift Servers.

Autenticación de los jugadores

Puedes usar Amazon Cognito y la sesión del jugador para IDs autenticar tus clientes de juego. Para administrar el ciclo de vida y las propiedades de las identidades de sus jugadores, utilice los grupos de usuarios de Amazon Cognito.

Si lo prefiere, cree una solución de identidad personalizada y alójela en AWS. También puede utilizar autorizadores Lambda para una lógica de autorización personalizada con API Gateway.

Recursos adicionales:

- [Uso de grupos de identidades \(identidades federadas\)](#) (Guía para desarrolladores de Amazon Cognito)
- [Introducción a los grupos de usuarios](#) (Guía para desarrolladores de Amazon Cognito)
- [Cómo configurar la autenticación del jugador con Amazon Cognito \(AWS para el blog de juegos\)](#)

Integración de la funcionalidad de cliente de juego de Amazon GameLift Servers

Integre la funcionalidad de alojamiento de juegos de Amazon GameLift Servers en cualquier componente de la solución que necesite comunicarse con el servicio Amazon GameLift Servers, por ejemplo, para obtener información sobre las sesiones de juego o crear sesiones de juego nuevas. En la mayoría de los casos, esta funcionalidad está integrada en el componente de servicio de backend. Añada funciones para tareas básicas, incluidas las siguientes:

- Solicitar la información y el estado de sesiones de juego activas.
- Reservar un espacio para un jugador nuevo en una sesión de juego existente.
- Crear una sesión de juego nueva para un grupo de jugadores.
- Solicitar el emparejamiento de uno o varios jugadores.
- Proporcionar datos actualizados de las sesiones de juego existentes.

Consulte [client/server Interacciones del juego con Amazon GameLift Servers](#) para obtener detalles adicionales sobre cómo interactúan los componentes de alojamiento de juegos a través del Amazon GameLift Servers SDKs.

Configuración de la API de Amazon GameLift Servers

Agregue el AWS SDK para C++ with Amazon GameLift Servers a un proyecto.

Añada código para inicializar un cliente de Amazon GameLift Servers y almacenar la configuración clave. Este código debe ejecutarse antes que cualquier código que dependa de Amazon GameLift Servers.

1. Configure una configuración del cliente. Utilice la configuración de cliente predeterminada o cree un objeto de configuración de cliente personalizado. Para obtener más información, consulte [AWS::Client::ClientConfiguration](#)(C++) o [AmazonGameLiftConfig](#)(C#).

La configuración de un cliente especifica la región y el punto de conexión de destino que se deben usar al contactar con Amazon GameLift Servers. La región identifica el conjunto de recursos implementados (flotas, colas y emparejadores) que se van a utilizar. La configuración del cliente predeterminada establece la ubicación en la región Este de EE. UU. (Norte de Virginia). Para utilizar cualquier otra región, cree una configuración personalizada.

2. Inicialice un cliente Amazon GameLift Servers. Utilice [Aws:GameLift:: GameLiftClient \(\)](#) (C++) o [AmazonGameLiftClient\(\)](#) (C#) con una configuración de cliente predeterminada o personalizada.
3. Si utiliza sesiones de jugador, añada un mecanismo para generar un identificador único para cada jugador. Para obtener más información, consulte [Generación de ID de jugador](#).
4. Recopile y almacene la siguiente información:
 - Flota de destino: si está creando sesiones de juego de forma manual en una flota específica, proporcione un ID de flota o un ID de alias que apunte a la flota de destino. Si se trata de una flota con múltiples ubicaciones, especifique una ubicación de flota. Como práctica recomendada, utilice los alias de flota para poder cambiar los jugadores de una flota a otra sin tener que actualizar los servicios de backend.
 - Cola de destino: en el caso de los juegos que utilizan colas de varias flotas para ubicar las nuevas sesiones de juego, especifique el nombre de la cola a la que se deben enviar las solicitudes de ubicación.
 - AWS credenciales: todas las llamadas Amazon GameLift Servers deben proporcionar las credenciales de la entidad Cuenta de AWS que aloja el juego. Estas credenciales se pueden obtener mediante la creación de un usuario jugador, tal y como se describe en [Configuración de acceso mediante programación para su juego](#). En función de cómo administre el acceso del usuario jugador, realice los siguientes procedimientos:
 - Si utiliza un rol para administrar los permisos de usuario de los jugadores, añada código para asumir el rol antes de llamar a una API de Amazon GameLift Servers. La solicitud para asumir el rol devolverá un conjunto de credenciales de seguridad temporales. Para obtener más información, consulte [Cambiar a un rol de IAM \(AWS API\)](#) en la Guía del usuario de IAM.
 - Si dispone de credenciales de seguridad de larga duración, configure el código para localizar y utilizar las credenciales almacenadas. Consulte [Autenticarse con credenciales de larga duración](#) en la Guía de referencia de herramientas AWS SDKs y herramientas. Para obtener información sobre el almacenamiento de credenciales, consulte las referencias de las AWS API para [\(C++\)](#) y [\(.NET\)](#).
 - Si tiene credenciales de seguridad temporales, añada código para actualizarlas periódicamente mediante AWS Security Token Service (AWS STS), tal y como se describe en la sección [Uso de credenciales de seguridad temporales con la AWS SDKs](#) Guía del usuario de IAM. El código debe solicitar nuevas credenciales antes de que las antiguas caduquen.

Obtención de sesiones de juego activas

Añada código para descubrir las sesiones de juego disponibles y administrar la configuración de las sesiones de juego y los metadatos.

Búsqueda de sesiones de juego activas

Se utiliza [SearchGameSessions](#) para obtener información sobre una sesión de juego específica, todas las sesiones activas o las sesiones que cumplen un conjunto de criterios de búsqueda. Esta llamada devuelve un [GameSession](#) objeto por cada sesión de juego activa que coincide con tu solicitud de búsqueda. Este objeto contiene el nombre de DNS, la dirección IP y el puerto que necesita el cliente de juego para conectarse a la sesión de juego.

Utilice los criterios de búsqueda para obtener una lista filtrada de las sesiones de juego activas a las que puedan conectarse los jugadores. Por ejemplo, puede filtrar sesiones de la siguiente manera:

- Excluya las sesiones de juego llenas: `CurrentPlayerSessionCount = MaximumPlayerSessionCount`.
- Elija las sesiones de juego en función de la duración de la sesión: Evaluar `CreationTime`.
- Encuentre sesiones de juego en función de una propiedad de juego personalizada: `gameSessionProperties.gameMode = "brawl"`

Administración de los datos de las sesiones de juego

Utilice cualquiera de las operaciones siguientes para recuperar o actualizar la información de la sesión de juego.

- [DescribeGameSessionDetails\(\)](#) — Obtén el estado de protección de una sesión de juego además de la información sobre la sesión de juego.
- [UpdateGameSession\(\)](#) — Cambia los metadatos y la configuración de una sesión de juego según sea necesario.
- [GetGameSessionLogUrl](#) — Accede a los registros de sesiones de juego almacenados.

Creación de sesiones de juego

Añada código para iniciar sesiones de juego nuevas en las flotas implementadas y ponerlas a disposición de los jugadores. Existen dos opciones para crear sesiones de juego, dependiendo de la forma en que la solución de alojamiento de juegos administre la ubicación de las sesiones de juego.

Creación de una sesión de juego en una cola con múltiples ubicaciones

Se usa [StartGameSessionPlacement](#) para poner en cola una solicitud para una nueva sesión de juego. Para usar esta operación de, cree una cola. Esto determina dónde coloca Amazon GameLift Servers la nueva sesión de juego. Para obtener más información sobre las colas y cómo utilizarlas, consulte [Configuración de la ubicación de las sesiones de juego](#).

Al crear una ubicación de la sesión de juego, especifique el nombre de la cola que utilizar, un nombre de sesión de juego, la cantidad máxima de jugadores simultáneos y un conjunto de propiedades del juego opcionales. Si lo desea, también puede proporcionar una lista de los jugadores para que se unan automáticamente a la sesión de juego. Si incluye los datos de latencia de los jugadores de las regiones relevantes, Amazon GameLift Servers utiliza esa información para colocar la nueva sesión de juego en una flota que proporcione una experiencia de juego perfecta a los jugadores.

Para obtener mediciones de latencia precisas, utilice señalizadores de pings de UDP de Amazon GameLift Servers. Estos puntos de conexión le permiten medir la latencia real de la red UDP entre los dispositivos de los jugadores y las posibles ubicaciones de alojamiento, de forma que pueda tomar decisiones de ubicación más precisas que utilizando pings de ICMP. Para obtener más información sobre el uso de señalizadores de pings de UDP para medir la latencia, consulte [Señalizadores de pings de UDP](#).

La ubicación de sesiones de juego es un proceso asíncrono. Una vez realizada la solicitud, puede esperar a que funcione o a que caduque. También puedes cancelar la solicitud en cualquier momento utilizando [StopGameSessionPlacement](#). Para comprobar el estado de su solicitud de colocación, llame [DescribeGameSessionPlacement](#).

Creación de una sesión de juego en una flota específica

Se usa [CreateGameSession](#) para crear una nueva sesión en una flota específica. Esta operación síncrona funciona o no dependiendo de si la flota dispone de los recursos necesarios para alojar una sesión de juego nueva. Tras Amazon GameLift Servers crear la nueva sesión de juego y devolver un [GameSession](#) objeto, podréis unir a los jugadores a ella.

Si utiliza esta operación, proporcione un ID de la flota o de alias, un nombre de sesión y un número máximo de jugadores simultáneos para dicho juego. De forma opcional, puede incluir un conjunto de propiedades del juego. Las propiedades del juego se definen en una matriz de pares clave-valor.

Si utiliza la característica de protección de recursos de Amazon GameLift Servers para limitar el número de sesiones de juego que puede crear un jugador, proporcione el ID de jugador de la persona que va a crear la sesión de juego.

Conexión de un jugador a una sesión de juego

Añada código para reservar una ranura de jugador en una sesión de juego activa y conectar clientes de juego a sesiones de juego. Esta acción está disponible si el juego usa sesiones de jugadores con un jugador único IDs. Para obtener más información sobre las sesiones de jugador, consulte [Amazon GameLift Servers y la experiencia del jugador](#).

1. Reserva de una ranura de jugador en una sesión de juego

Para reservar un espacio de jugador, cree una sesión de jugador para la sesión de juego. Hay dos formas de hacer esto:

- Si utilizas [StartGameSessionPlacement](#) quieres [StartMatchmaking](#) crear una sesión de juego, puedes incluir a uno o más jugadores en la solicitud para crear sesiones de jugadores para ellos en la nueva sesión de juego.
- Para añadir jugadores a una sesión de juego existente, llama [CreatePlayerSession](#) o [CreatePlayerSessions](#) con un identificador de sesión de juego.

Una solicitud de sesión de jugador debe incluir un ID de jugador único. Para obtener más información, consulte [Generación de ID de jugador](#). Al recibir la solicitud, Amazon GameLift Servers verifica que la sesión de juego acepta jugadores nuevos y que tiene espacios de jugador disponibles. Si lo consigue, Amazon GameLift Servers reserva un espacio para el jugador, crea la nueva sesión de jugador y devuelve un [PlayerSession](#) objeto.

Una sesión de jugador puede incluir un conjunto de datos de jugador personalizados. Estos datos se almacenan en el objeto de sesión de jugador recién creado. Amazon GameLift Servers transfiere el objeto al servidor de juegos cuando el jugador se conecta directamente a la sesión de juego. A la hora de solicitar varias sesiones de jugador, asigne cada cadena de datos de jugador a un ID de jugador.

2. Conexión a una sesión de juego

Agrega código al servicio de backend para recuperar el `PlayerSession` objeto (por ejemplo, llamando a [DescribePlayerSessions\(\)](#)) y devuélvelo al cliente del juego. Este objeto contiene el nombre de DNS, la dirección IP y el puerto. El cliente de juego puede usar esta información para establecer una conexión directa con el servidor.

- Puede conectarse mediante el puerto especificado y el nombre de DNS o la dirección IP asignados al proceso del servidor.

- Si las flotas tienen habilitada la generación de certificados TLS, conéctese mediante el nombre de DNS y el puerto.
- Si el servidor de juegos valida las conexiones de los jugadores entrantes, consulte el ID de sesión del jugador.

Una vez establecida la conexión, el cliente de juegos y el proceso del servidor se comunican directamente sin involucrar a Amazon GameLift Servers. El servidor mantiene la comunicación con Amazon GameLift Servers para notificar el estado de conexión del jugador, el estado, etc.

Si el servidor de juegos valida a los jugadores entrantes, verifica que el ID de sesión del jugador coincide con un espacio reservado en la sesión de juego, y acepta o rechaza la conexión del jugador. Cuando el jugador se desconecte, el proceso del servidor notificará que se ha desconectado.

Transferencia de datos de juego personalizados a una sesión de juego

El cliente de juego puede transferir datos a una sesión de juego. Al crear una sesión de juego, puede incluir un conjunto de propiedades de juego (pares clave-valor) o datos de la sesión de juego (valores de cadena) en la solicitud. También puede actualizar una sesión de juego existente con datos de juego nuevos o actualizados. Estos datos se transfieren al proceso del servidor de juegos que aloja la sesión de juego para que los utilice el código del servidor de juegos. No puede eliminar las propiedades de juego.

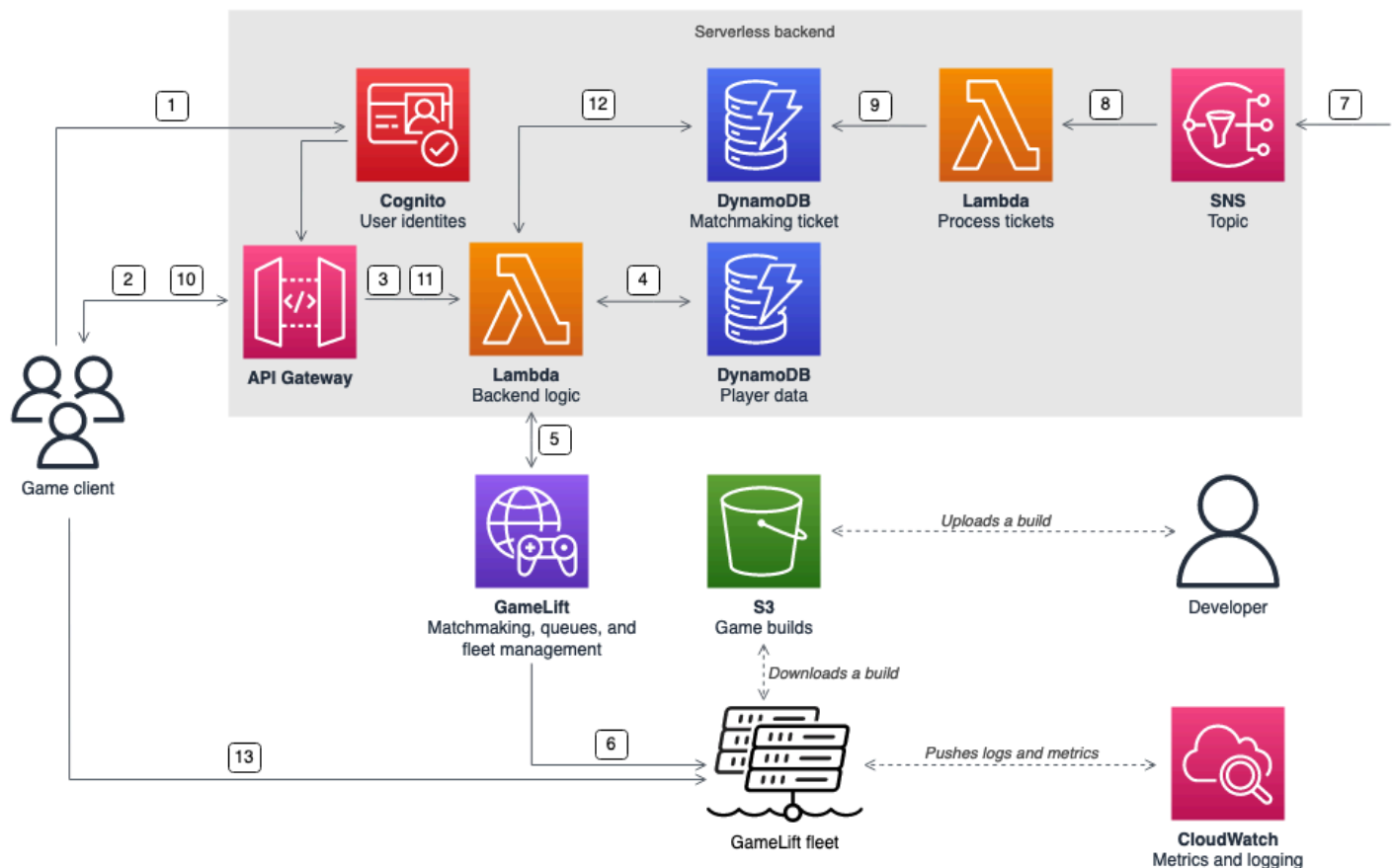
Como ejemplo, supongamos que el juego define niveles de dificultad: Novice, Easy, Intermediate y Expert. Un jugador elige unirse a un juego Easy. El cliente de juego, a través del servicio de backend, solicita una nueva sesión de juego con la siguiente propiedad de juego: {"Key": "Difficulty", "Value": "Easy"}. En respuesta, Amazon GameLift Servers solicita a un servidor de juegos disponible que inicie una sesión de juego nueva, y transfiere el objeto GameSession. El proceso del servidor de juegos utiliza la propiedad de juego proporcionada para establecer el nivel de dificultad de la sesión de juego.

Más información

- [GameProperty data type](#)
- [SearchGameSessions\(\) ejemplos](#)
- [UpdateGameSession GameProperties parámetro \(\)](#)

Al utilizar una arquitectura de servicio de cliente sin servidor, el backend puede ver el estado de los tickets de emparejamiento desde una base de datos altamente escalable en lugar de acceder directamente a la API de Amazon GameLift Servers.

En el siguiente diagrama se muestra un backend sin servidor creado con el que los jugadores pueden crear juegos Servicios de AWS que se ejecutan en flotas. Amazon GameLift Servers La siguiente lista proporciona una descripción de cada aviso numerado en el diagrama. Para probar este ejemplo, consulta Hospedaje de juegos [basado en sesiones multijugador](#) en on. AWS GitHub



1. El cliente del juego solicita una identidad de usuario de Amazon Cognito de un grupo de identidades de Amazon Cognito.
2. El cliente del juego recibe credenciales de acceso temporales y solicita una sesión de juego a través de una API de Amazon API Gateway.
3. API Gateway invoca una AWS Lambda función.

4. La función de Lambda solicita los datos del reproductor de una tabla NoSQL de Amazon DynamoDB. La función proporciona la identidad de Amazon Cognito en los datos de contexto de la solicitud.
5. La función de Lambda solicita una coincidencia mediante el emparejamiento de Amazon GameLift Servers FlexMatch.
6. FlexMatchempareja a un grupo de jugadores con la latencia adecuada y, a continuación, solicita la ubicación de una sesión de juego en una Amazon GameLift Servers cola. La cola tiene flotas con una o más Región de AWS ubicaciones.
7. Cuando Amazon GameLift Servers coloca la sesión en una de las ubicaciones de la flota, Amazon GameLift Servers envía una notificación de evento a un tema de Amazon Simple Notification Service (Amazon SNS).
8. Una función de Lambda recibe el evento de Amazon SNS y lo procesa.
9. Si el ticket de emparejamiento es un evento MatchmakingSucceeded, la función de Lambda escribe el resultado, junto con el puerto y la dirección IP del servidor de juegos, en una tabla de DynamoDB.
- 10El cliente del juego envía una solicitud firmada a API Gateway para ver el estado del ticket de emparejamiento en un intervalo específico.
- 11API Gateway utiliza una función de Lambda que comprueba el estado del ticket de emparejamiento.
- 12La función de Lambda comprueba la tabla de DynamoDB para comprobar si el ticket es correcto. Si lo es, la función devuelve al cliente el puerto y la dirección IP del servidor de juegos, junto con el ID de sesión del jugador. Si el ticket no es correcto, la función envía una respuesta verificando que el emparejamiento no está listo aún.
- 13El cliente del juego se conecta al servidor de juegos mediante TCP o UDP mediante el puerto y la dirección IP que proporciona el servicio de backend. A continuación, el cliente del juego envía el ID de sesión del jugador al servidor de juegos, que lo valida mediante el SDK del servidor de Amazon GameLift Servers.

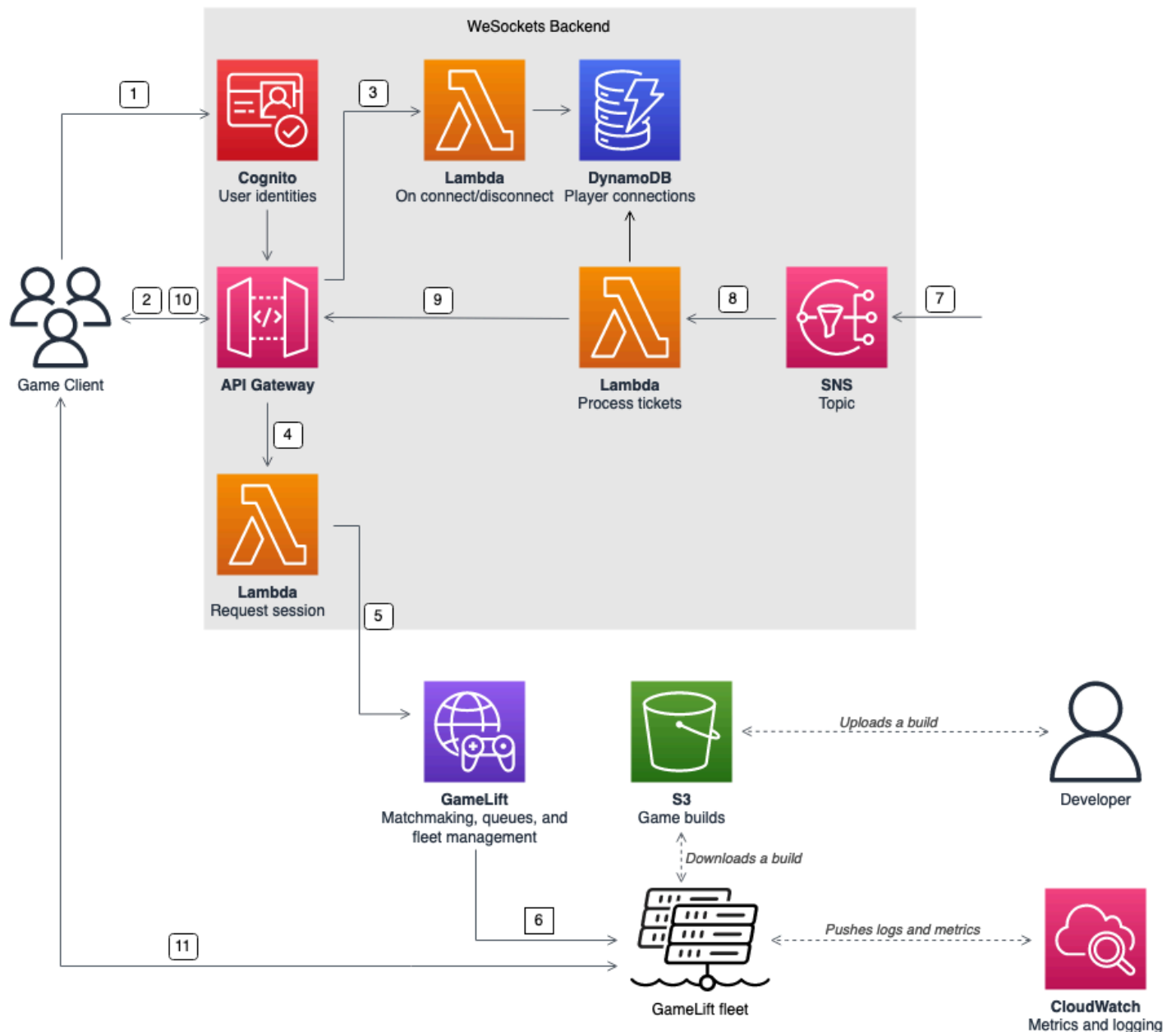
Servidores de sesión de juego independientes con un backend basado WebSocket

Con una arquitectura WebSocket basada en Amazon API Gateway, puede realizar solicitudes de emparejamiento WebSockets y enviar notificaciones push para completar el emparejamiento

mediante mensajes iniciados por el servidor. Esta arquitectura mejora el rendimiento mediante una comunicación bidireccional entre el cliente y el servidor.

Para obtener más información sobre el uso de API Gateway WebSocket APIs, consulte [Trabajar con WebSocket APIs](#).

El siguiente diagrama muestra una arquitectura de backend WebSocket basada en la que se utiliza API Gateway y otros Servicios de AWS para unir a los jugadores en juegos que se ejecutan en Amazon GameLift Servers flotas. La siguiente lista proporciona una descripción de cada aviso numerado en el diagrama.



1. El cliente del juego solicita una identidad de usuario de Amazon Cognito de un grupo de identidades de Amazon Cognito.
2. El cliente del juego firma una WebSocket conexión a una API API Gateway con las credenciales de Amazon Cognito.
3. API Gateway llama a una AWS Lambda función de la conexión. La función almacena la información de conexión en una tabla de Amazon DynamoDB.
4. El cliente del juego envía un mensaje a una función Lambda, a través de la API API Gateway a través de la WebSocket conexión, para solicitar una sesión.
5. Una función de Lambda recibe el mensaje y, a continuación, solicita un emparejamiento mediante el emparejamiento de Amazon GameLift Servers FlexMatch.
6. Tras una FlexMatch partida, un grupo de jugadores FlexMatch solicita que la sesión de juego se sitúe en una Amazon GameLift Servers cola.
7. Cuando Amazon GameLift Servers coloca la sesión en una de las ubicaciones de la flota, Amazon GameLift Servers envía una notificación de evento a un tema de Amazon Simple Notification Service (Amazon SNS).
8. Una función de Lambda recibe el evento de Amazon SNS y lo procesa.
9. Si el ticket de emparejamiento es un evento MatchmakingSucceeded, la función de Lambda solicita a DynamoDB la conexión de jugador correcta. A continuación, la función envía un mensaje al cliente del juego a través de la API API Gateway a través de la WebSocket conexión. En esta arquitectura, el cliente del juego no consulta activamente el estado del emparejamiento.
- 10 El cliente del juego recibe el puerto y la dirección IP del servidor del juego, junto con el ID de sesión del jugador, a través de la WebSocket conexión.
- 11 El cliente del juego se conecta al servidor de juegos mediante TCP o UDP mediante el puerto y la dirección IP que proporciona el servicio de backend. El cliente del juego también envía el ID de sesión del jugador al servidor de juegos, que lo valida mediante el SDK del servidor de Amazon GameLift Servers.

Implementación de flotas de alojamiento para Amazon GameLift Servers

La implementación de los recursos de alojamiento implica crear y configurar la infraestructura informática que ejecutará los servidores de sus juegos. Amazon GameLift Servers ofrece varios tipos de flotas que se adaptan a las diferentes necesidades de alojamiento, desde Nube de AWS recursos totalmente gestionados hasta soluciones híbridas que combinan la infraestructura local y en la nube.

Elija el tipo de flota que mejor se adapte a sus requisitos de costo, control, escalabilidad y distribución geográfica. También puede combinar varios tipos de flotas en una única solución de alojamiento para optimizarla para diferentes escenarios o poblaciones de jugadores.

Características de las flotas

Una flota de Amazon GameLift Servers es un conjunto de recursos de computación que ejecutan servidores de juegos y alojan sesiones de juego para los jugadores. Las flotas pueden variar en el tipo de recursos de computación que utilicen y en su método de administración. El tamaño de una flota (el número de sesiones de juego y jugadores a los que puede ofrecer soporte) depende del número de recursos de computación que le proporcione. Todas las flotas de Amazon GameLift Servers tienen las siguientes características:

- Los procesos del servidor de juegos que se ejecutan en todas las flotas están integrados en el SDK del servidor de Amazon GameLift Servers y se comunican con el servicio de Amazon GameLift Servers de la misma manera. Los servidores de juegos informan de su disponibilidad para alojar sesiones de juego y jugadores, responden a las peticiones para iniciar o detener las sesiones de juego y llevan a cabo otras interacciones.
- Amazon GameLift Servers maneja del mismo modo la ubicación de las sesiones de juego de todas las flotas. Amazon GameLift Servers hace un seguimiento del estado de los servidores de juegos de una flota y elige entre los servidores de juegos disponibles para alojar una nueva sesión de juego. Este proceso se utiliza tanto si el juego ubica las sesiones de juego en una sola flota como si utiliza una [cola de sesiones de juego](#) para equilibrar el alojamiento entre las distintas flotas. Con una cola, también puede personalizar las decisiones sobre la ubicación de tal modo que se tengan en cuenta factores como el costo de los recursos y la latencia.
- Todas las flotas admiten el uso de un emparejador de FlexMatch junto con una cola de ubicación de sesiones de juego. El servicio de Amazon GameLift Servers recibe las solicitudes de emparejamiento de los jugadores, forma los emparejamientos y los pasa a la cola de sesiones de juego para encontrar servidores de juego disponibles.
- Amazon GameLift Servers recopila una amplia variedad de métricas de flota; por ejemplo, las métricas de estado de los procesos de computación y del servidor o las métricas de uso de las sesiones de juego y de la actividad de los jugadores. Si desea ver una lista completa de las métricas disponibles, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

Temas

- [Guía de decisiones: Elección de una opción de alojamiento](#)

- [Amazon GameLift Servers EC2 flotas gestionadas](#)
- [Flotas de contenedores administradas de Amazon GameLift Servers](#)
- [Flotas de Amazon GameLift Servers Anywhere](#)
- [Creación de una solución de alojamiento híbrida](#)

Guía de decisiones: Elección de una opción de alojamiento

Amazon GameLift Servers ofrece varias opciones de alojamiento, cada una diseñada para diferentes escenarios y requisitos de alojamiento. Esta guía le ayuda a elegir el tipo de flota en función de las necesidades técnicas, las preferencias operativas y las limitaciones empresariales del juego. Para obtener descripciones detalladas, consulte [opciones de alojamiento de juegos de Amazon GameLift Servers](#).

Comparación de flotas

En la siguiente tabla se comparan las características clave de cada opción de alojamiento:

Opción de alojamiento	Recomendada para	Nivel de administración	Ventajas principales
Gestionado EC2	Alojamiento de producción con AWS administración completa	Totalmente administrado	Escalado automático, alcance global, sobrecarga operativa mínima
Contenedores administrados	Aplicaciones en contenedores que requieren orquestación	Totalmente administrado	Orquestación de contenedores, eficiencia de los recursos, implementación moderna
Anywhere	Entornos en las instalaciones, híbridos o de desarrollo	Autoadministrada	Control del hardware, requisitos de conformidad, optimización de costos

Factores de decisión clave

Tenga en cuenta estos factores a la hora de elegir una opción de alojamiento:

Complejidad operativa

¿Qué grado de administración de la infraestructura desea asumir? Las flotas administradas implican una sobrecarga operativa mínima, mientras que las flotas de Anywhere le brindan un control total, pero requieren más administración.

Requisitos de escalabilidad

¿Necesita escalado automático en función de la demanda de los jugadores? Las flotas administradas ofrecen escalado automático integrado, mientras que las flotas de Anywhere requieren una administración manual de la capacidad.

Distribución geográfica

¿Dónde están ubicados los jugadores? Las flotas gestionadas se pueden implementar en varias AWS regiones del mundo, mientras que las flotas de Anywhere dependen de las ubicaciones de su propia infraestructura.

Conformidad y seguridad

¿Tiene requisitos de conformidad específicos o necesita mantener los datos en las instalaciones? Las flotas de Anywhere le permiten mantener el control total de la infraestructura y la ubicación de los datos.

Consideraciones sobre costos

¿Cuál es su presupuesto y su estrategia de optimización de costos? A la hora de comparar las opciones, tenga en cuenta tanto los costos de infraestructura como la sobrecarga operativa.

Soluciones híbridas

No tiene que elegir un único tipo de flota para su solución de alojamiento de juegos. Muchos juegos utilizan un enfoque híbrido que combina varios tipos de flota, lo que permite optimizarlos para diferentes escenarios:

- **Desarrollo y prueba:** utilice flotas de Anywhere para los entornos de desarrollo y de prueba y, a continuación, impleméntelas en flotas administradas para los entornos de producción.

- Optimización regional: utilice las flotas gestionadas en las que haya una Regiones de AWS alta densidad de jugadores y las flotas de Anywhere en las regiones en las que ya disponga de una infraestructura.
- Optimización de costos: utilice flotas administradas para obtener capacidad inicial, y flotas de Anywhere para desbordamientos o cargas de trabajo específicas.

Amazon GameLift Servers EC2 flotas gestionadas

Amazon GameLift Servers EC2 Las flotas gestionadas proporcionan recursos basados en la nube para el alojamiento de producción. Con una flota gestionada, obtienes la flexibilidad, la seguridad y la fiabilidad de Nube de AWS los recursos optimizados para el alojamiento de juegos multijugador. Amazon GameLift Servers proporciona herramientas sólidas de administración de hosts.

Una EC2 flota gestionada es un conjunto de instancias de Amazon Elastic Compute Cloud (Amazon EC2) que Amazon GameLift Servers posee y opera en función de su configuración. Estas instancias están ubicadas físicamente en Regiones de AWS o zonas locales compatibles. Al crear una flota, eliges un tipo de EC2 instancia que cumpla con los requisitos de potencia informática, memoria, almacenamiento y capacidades de red del servidor de juegos.

Al iniciar cada instancia de la flota, Amazon GameLift Servers implementa la compilación del servidor de juegos con el entorno de tiempo de ejecución necesario. El entorno de tiempo de ejecución utiliza la versión de la imagen de máquina de Amazon (AMI) más reciente disponible al crear la flota. Todas las instancias de la flota utilizan la misma versión de la AMI.

Note

Como práctica recomendada, te recomendamos reemplazar tus flotas cada 30 días para mantener un entorno seguro y de up-to-date ejecución para tus servidores de juegos alojados. Para ello, es necesario crear una nueva flota y migrar a ella el tráfico de jugadores. Para obtener más orientación, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Tras instalar el entorno de tiempo de ejecución y la compilación del servidor de juegos en una instancia, Amazon GameLift Servers comienza a lanzar los procesos del servidor de juegos. Cada servidor de juegos establece una conexión con el servicio de Amazon GameLift Servers, informa de que está listo para alojar una sesión de juego y empieza a comunicar su estado. Amazon GameLift Servers puede entonces solicitar al proceso del servidor que inicie una sesión de juego.

Además de la implementación de flotas, Amazon GameLift Servers administra automáticamente las siguientes tareas de administración del host por usted:

- Hace un seguimiento del estado de todos los recursos de computación de la flota y reemplaza los recursos obsoletos o en mal estado.
- Gestiona la autenticación para la comunicación entre los procesos del servidor y el servicio de Amazon GameLift Servers.
- Inicia y detiene automáticamente los procesos del servidor de juegos en cada recurso de computación según la configuración del tiempo de ejecución.
- Ofrece herramientas de escalado automático que ajustan la capacidad de la flota de forma dinámica para satisfacer la demanda de los jugadores.
- Reporta las métricas de rendimiento de las EC2 instancias de la flota.

Consulte estos temas sobre cómo configurar y mantener las EC2 flotas gestionadas:

- [Hoja de ruta de desarrollo para el alojamiento con EC2 administrado por Amazon GameLift Servers](#)
- [Cree una EC2 flota Amazon GameLift Servers gestionada](#)
- [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#)
- [Personalizaciones de recursos de alojamiento](#)
- [Actualización de la configuración de una flota de Amazon GameLift Servers](#)

Flujo de trabajo EC2 gestionado para la creación

En el caso de las flotas administradas, Amazon GameLift Servers configura el recurso de flota e implementa un conjunto de recursos de computación con el software del servidor de juegos instalado y en ejecución. Cuando el flujo de trabajo de creación se complete y se complete correctamente, la flota tendrá una EC2 instancia activa en la región de origen de la flota y una en cada una de las ubicaciones remotas de la flota. Todas las instancias de juego estarán listas para alojar sesiones de juego.

1. Amazon GameLift Servers crea un recurso de flota en la región de origen de la flota y establece la capacidad deseada en cada ubicación a una (1) instancia. El estado de la flota y la ubicación se establece en Nuevo.
2. Amazon GameLift Servers comienza a escribir los eventos en el registro de eventos de la flota.

3. Amazon GameLift Servers establece el estado de la flota en Descargando y comienza a preparar el software del servidor de juegos para la implementación.
 - a. Obtiene la compilación del servidor de juegos cargado y extrae los archivos comprimidos.
 - b. Ejecuta scripts de instalación (si se proporcionan).
 - c. Establece el estado de la flota en Validando y comienza a verificar que no haya habido errores al descargar e instalar los archivos de compilación.
4. Amazon GameLift Servers establece el estado de la flota en Construcción, configura el hardware de la flota y asigna una EC2 instancia para cada instancia de la flota.
5. Amazon GameLift Servers establece el estado de la flota en Activando. Inicia un proceso de servidor de juegos en cada instancia (según las instrucciones de tiempo de ejecución de la flota) y prueba la conectividad entre la compilación y el servicio de Amazon GameLift Servers.
6. Cuando los procesos del servidor de juegos de cada instancia establecen una conexión e indican que están preparados para alojar sesiones de juego, Amazon GameLift Servers establece los estados de flota y ubicación en Activo. En este momento, la flota se considera lista para alojar sesiones de juego.

Cree una EC2 flota Amazon GameLift Servers gestionada

En este tema se describe cómo crear una EC2 flota Amazon GameLift Servers gestionada. Las flotas gestionadas utilizan instancias informáticas de Amazon Elastic Compute Cloud (Amazon EC2) optimizadas para el alojamiento de juegos multijugador. Puede crear flotas gestionadas que desplieguen cómputos en todo el mundo Regiones de AWS y en las Zonas Locales compatibles con Amazon GameLift Servers.

Al crear un nuevo recurso de EC2 flota gestionada, se inicia inmediatamente la primera fase de creación de la flota. La creación gestionada de flotas pasa por varias fases, ya que se Amazon GameLift Servers despliega una EC2 instancia, se instala un entorno de ejecución y el servidor de juegos se basa en la instancia, y se inicia el lanzamiento de los servidores de juegos. Según el entorno de ejecución que requiera la compilación del servidor de juegos, se Amazon GameLift Servers implementará la última versión de Amazon Machine Image (AMI) en el momento de crear la flota (y todas las instancias futuras de la flota utilizarán la misma versión). Puedes monitorizar el estado de la flota en la consola o mediante el AWS Command Line Interface (AWS CLI). Cuando la flota está lista para alojar sesiones de juego, el estado cambia a ACTIVE. Para obtener más información sobre la creación de flotas administradas, consulte estos temas:

Note

Como práctica recomendada, te recomendamos sustituir tus flotas cada 30 días para mantener un entorno seguro y de up-to-date ejecución para tus servidores de juegos alojados. Para ello, es necesario crear una nueva flota y migrar a ella el tráfico de jugadores. Para obtener más orientación, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Para crear una flota gestionada EC2

Usa la Amazon GameLift Servers consola o el AWS Command Line Interface (AWS CLI) para crear una EC2 flota gestionada.

Console

En la [Amazon GameLift Servers consola](#), utilice el panel de navegación para abrir la página EC2Flotas gestionadas. Seleccione Crear EC2 flota gestionada para iniciar el flujo de trabajo de creación de flotas.

Paso 1: Defina los detalles de EC2 la flota gestionada

i Para una configuración de flota mínima:

- Proporcione un nombre de flota.
- Elija Tipo binario: compilación. Elija una compilación de la lista de compilaciones de servidor que haya cargado previamente en Amazon GameLift Servers.
- Omita las secciones Detalles adicionales y Etiquetas para usar los ajustes predeterminados.

1. Rellene la sección Detalles de la flota:

- a. Escriba un Nombre para la flota. Los nombres de flota son meramente descriptivos, y son útiles para ordenar o filtrar una lista de flotas. No es necesario que sean únicos.
- b. Proporcione una descripción breve de la flota para incluir cualquier información personalizada que desee registrar sobre la flota.

- c. Seleccione Tipo binario, Compilación y use la lista desplegable para elegir una compilación del servidor de juegos que haya cargado previamente en Amazon GameLift Servers.
2. Defina los Detalles adicionales según sea necesario.
 - a. Si el ejecutable del servidor de juegos necesita acceder a otros AWS recursos de su cuenta, especifique un rol de instancia de IAM con los permisos necesarios. Para obtener más información, incluida la forma de autorizar otras aplicaciones del lado del servidor (como un CloudWatch agente), consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#) Esta configuración no se puede cambiar después de crear la flota.

Debe crear el rol antes de crear una flota que lo utilice. Además, para crear una flota con un rol de instancia, el usuario de AWS debe tener el permiso `PassRole` de IAM (consulte [Ejemplo de permisos de IAM para Amazon GameLift Servers](#)).
 - b. Active la opción Generar un certificado TLS para configurar la autenticación y el cifrado del juego. Los clientes del juego utilizan este certificado para autenticar un servidor de juego al conectarse y cifrar todas las comunicaciones entre el cliente y el servidor. Para cada instancia de una flota habilitada con TLS, Amazon GameLift Servers también crea una nueva entrada de DNS con el certificado. Esta configuración no se puede cambiar después de crear la flota.
 - c. Amazon GameLift Servers emite datos de métricas para cada flota. Si desea combinar los datos de las métricas de varias flotas, especifique un nombre en Grupo de métricas. Utilice el mismo nombre de grupo de métricas para todas las flotas cuyas métricas desea combinar. Se utiliza CloudWatch para ver los datos agregados del grupo de métricas.
3. Añada Etiquetas al recurso de flota. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Asigne etiquetas a AWS los recursos que desee clasificar, por ejemplo, por propósito, propietario o entorno. Para cada etiqueta que desee añadir, elija Añadir etiqueta nueva.
4. Seleccione Siguiente para continuar con el flujo de trabajo.

Paso 2: Defina los detalles de la instancia

En este paso, especifique el tipo de recursos de hardware que va a utilizar para los servidores de juegos y dónde quiere implementarlos. Si elige varias ubicaciones, podrá implementar el

servidor de juegos en una ubicación geográfica más amplia, de modo que esté más cerca de los jugadores y minimice la latencia. Todas las EC2 instancias de una flota deben ser del mismo tipo. No todos los tipos de instancias están disponibles en todas las ubicaciones.

 Para una configuración de flota mínima:

- Empezar por implementar solo en la ubicación de origen. No es necesario añadir ubicaciones remotas para las fases iniciales de desarrollo y prueba.
- Defina el tipo de flota como “Bajo demanda”. Las flotas de spot requieren un trabajo de configuración adicional.
- Establezca el tipo de instancia en “c5.large”. Este tipo de instancia que se usa habitualmente está disponible en todas Regiones de AWS. Puede optimizar el uso del hardware más tarde.

1. En Implementación de instancias, especifique el tipo y las ubicaciones de la flota.
 - a. Seleccione una o más Ubicaciones adicionales en las que desee implementar las instancias de flota. Estas ubicaciones remotas se añaden a la ubicación de origen de la flota (que está preseleccionada), que es Región de AWS donde se va a crear la flota. Puede seleccionar ubicaciones remotas de todas Regiones de AWS las Zonas Locales Amazon GameLift Servers compatibles.

Para obtener más información sobre las ubicaciones compatibles, incluido cómo usar una Región de AWS que no esté habilitada de forma predeterminada, consulta [Ubicaciones de servicio de Amazon GameLift Servers](#) el tema del alojamiento gestionado. Consulte también las [cuotas](#) de Amazon GameLift Servers en las ubicaciones por flota.
 - b. Seleccione si va a usar instancias Bajo demanda o de Spot para esta flota. Para obtener más información sobre los tipos de flotas, consulte [Instancias bajo demanda frente a instancias de spot](#).
2. Elija una configuración de Amazon EC2 Instance que se adapte a sus necesidades y que esté disponible en todas las ubicaciones seleccionadas. Esta lista se filtra en función de su ubicación actual y de las selecciones de tipo de flota. Puede filtrarla aún más por otros factores, como el tipo de instancia, la familia y la arquitectura. No se puede cambiar el tipo de instancia después de crear la flota.

Algunas ubicaciones tienen opciones limitadas de tipos de instancia. Si el tipo de instancia que prefiere no está disponible en todas las ubicaciones, elija el valor de disponibilidad de la ubicación para ver todos los detalles. Para dar cabida a todas las ubicaciones, es posible que necesite crear flotas independientes con distintos tipos de instancias.

Para obtener más información sobre cómo elegir un tipo de instancia, consulte [Tipos de instancias](#). Para obtener más información sobre las arquitecturas de Amazon EC2 Arm, consulte [AWS el procesador Graviton](#) y los tipos de [EC2 instancias de Amazon](#). Para obtener una lista completa de los tipos de instancias compatibles Amazon GameLift Servers, consulta la referencia de la API para [EC2InstanceType\(\)](#)CreateFleet().

 Note

Las instancias de Graviton ARM requieren una compilación de servidor para una AMI de Linux. Se requiere el SDK del servidor 5.1.1 o posterior para C++ y C#. Se requiere el SDK del servidor 5.1.1 o posterior para continuar. Estas instancias no out-of-the-box admiten la instalación de Mono en Amazon Linux 2023 (AL2023) o Amazon Linux 2 (AL2).

3. Seleccione Siguiente para continuar con el flujo de trabajo.

Paso 4: configurar el tiempo de ejecución

En este paso, describe cómo quiere que cada instancia de la flota ejecute el software de su servidor de juegos. Defina una línea de proceso de servidor independiente para cada ejecutable que se ejecute en una instancia y decida cuántos procesos de servidor se ejecutarán simultáneamente. Abra los puertos en cada instancia para que los jugadores puedan conectarse directamente a los servidores del juego. Puede actualizar estos ajustes de flota más tarde.

 Para una configuración de flota mínima:

- Defina un elemento de línea de proceso de servidor único para el archivo ejecutable del servidor de juegos. Si el servidor de juegos requiere que se ejecuten otros procesos, cree también una definición para cada uno de ellos.

- Use el número predeterminado de procesos simultáneos (1) para cada elemento de línea.
- Omita la configuración de activación de la sesión de juego.
- Especifique un número de puerto único.
- Omita la configuración de recursos de la sesión de juego.

1. Cree una configuración del tiempo de ejecución para dar instrucciones a Amazon GameLift Servers sobre cómo ejecutar procesos del servidor en cada una de las instancias de la flota. Puede cambiar la configuración de tiempo de ejecución de una flota en cualquier momento tras la implementación.
 - a. Introduzca la Ruta de lanzamiento de un archivo ejecutable en su compilación. En las instancias de Windows, los ejecutables de servidores de juegos se compilan en la ruta `C:\game`. En las instancias de Linux, los servidores de juegos están diseñados para `/local/game`. Ejemplos: **`C:\game\MyGame\server.exe`** o **`/local/game/MyGame/server.exe`**.
 - b. Introduzca Parámetros de lanzamiento opcionales para pasarlos al ejecutable del juego. Ejemplo: **`+sv_port 33435 +start_lobby`**.
 - c. Especifique el número de Procesos simultáneos que se ejecutarán en cada instancia. En el caso de un ejecutable de servidor de juegos, cada proceso puede alojar una sesión de juego, por lo que los procesos simultáneos determinan el número de sesiones de juego que la instancia puede alojar simultáneamente.

Revise las [cuotas](#) de Amazon GameLift Servers en los procesos de servidor por instancia. Estas cuotas se aplican al total de procesos simultáneos de todas las configuraciones. Si configura la flota de manera que se superen, esta no podrá activarse.

2. Use los valores predeterminados de Activación de la sesión de juego o personalícelos para su juego. Si la configuración del tiempo de ejecución requiere varios procesos de servidor de juego simultáneos por instancia, estos ajustes determinan la rapidez con la que se pueden iniciar nuevas sesiones de juego.
 - a. Establezca el Número máximo de activaciones de sesiones de juego simultáneas para limitar el número de servidores de juego en una instancia que estén preparando una nueva sesión de juego. Esta configuración resulta útil cuando el inicio de varias

sesiones de juego nuevas requiere muchos recursos y puede afectar al rendimiento de otras sesiones de juego en curso.

- b. Establezca el Tiempo de espera de la nueva activación de modo que refleje el tiempo máximo que debe tardar una nueva sesión de juego en completar la activación y en informar a los jugadores de que todo está listo. Amazon GameLift Servers finaliza la activación de una sesión de juego cuando se excede este valor.
3. Abra la configuración de los EC2 puertos para permitir que el tráfico entrante acceda a los procesos del servidor de la flota. Estos ajustes no son necesarios para crear una flota, pero sí es necesario configurarlos para que los jugadores puedan conectarse a las sesiones de juego de la flota.

Para cada configuración de puerto, elija el Tipo de protocolo de transferencia de datos que se utilizará para la comunicación entre el cliente del juego y el servidor. Proporcione un Rango de puertos (en formato `nnnnn[-nnnnn]`) y un Rango de direcciones IP en notación CIDR (como `0.0.0.0/0`, que permite el acceso a cualquier persona).

Si necesita establecer varios rangos no consecutivos, cree varios ajustes de puerto.

- a. Active o desactive la Política de protección para el escalado de juegos en todas las instancias de la flota. Durante un evento de reducción vertical, Amazon GameLift Servers no finalizará las instancias de flota protegidas si alojan sesiones de juego activas.
 - b. Establezca un Límite de creación de recursos máximo si quiere limitar el número de sesiones de juego que puede crear un jugador durante un período de tiempo específico.
5. Seleccione Siguiente para continuar con el flujo de trabajo.

Paso 5: revisar y crear

Antes de crear la flota, revise la configuración. Aunque algunos ajustes se pueden actualizar más adelante (consulte [Actualización de la configuración de una flota de Amazon GameLift Servers](#)), no se permiten cambios en los siguientes ajustes tras crear la flota:

- Tipo de cómputo: no se puede convertir una EC2 flota gestionada en una flota de Anywhere.

- Binario de servidor de juegos: para implementar una actualización en la compilación del servidor de juegos, debe crear una flota nueva.
- Opciones adicionales, como el rol de instancia y la generación de certificados TLS.
- Detalles de la instancia, incluidos el tipo de flota (puntual o bajo demanda) y el tipo de EC2 instancia.

Cuando lo tenga todo listo para implementar la nueva flota, elija Crear. Amazon GameLift Servers comienza inmediatamente el proceso de activación de la flota, para lo cual asigna un ID único y pone la flota en estado NEW. Haga un seguimiento del progreso de la flota desde la página Flotas. Consulte la página de detalles de la flota y vaya a la pestaña Eventos.

Puede actualizar la capacidad de alojamiento de la flota después de que esta haya alcanzado el estado ACTIVO. Amazon GameLift Servers implementa inicialmente una flota con una sola instancia en cada ubicación de la flota, y usted ajusta la capacidad añadiendo instancias a cada ubicación. Para obtener más información, consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).

AWS CLI

Utilice el comando `create-fleet` para crear una flota con el tipo de recurso de computación EC2. Amazon GameLift Servers crea el recurso de flota en su Región de AWS predeterminada actual (también puede añadir una etiqueta `--region` para especificar una Región de AWS distinta).

Creación de una flota administrada de nivel mínimo

El siguiente ejemplo de solicitud crea una nueva flota con los ajustes mínimos necesarios para implementar una flota con servidores de juegos en funcionamiento a los que los clientes de juegos puedan conectarse. La nueva flota tiene las siguientes características:

- Especifica una compilación de servidor de juegos que se ha cargado en Amazon GameLift Servers y se encuentra en estado READY.
- Utiliza instancias bajo demanda `c5.large` con el sistema operativo necesario para la compilación del juego seleccionada.
- Establece la base de la flota Región de AWS `us-west-2` y despliega las instancias únicamente en esa región.
- Según la configuración del tiempo de ejecución, cada recurso de computación de la flota ejecuta un proceso de servidor de juegos, lo que significa que cada recurso solo puede alojar

una sesión de juego a la vez. El tiempo de espera de activación de la sesión de juego está establecido en el valor predeterminado de 300 segundos y no hay un límite en el número de activaciones simultáneas.

- Los jugadores pueden conectarse a los servidores del juego mediante una configuración de puerto único de 33435.
- Todas las demás características están desactivadas o utilizan la configuración predeterminada.

```
aws gamelift create-fleet \  
  --name MinimalFleet123 \  
  --description "A basic test fleet" \  
  --region us-west-2 \  
  --ec2-instance-type c5.large \  
  --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"
```

Creación de una flota administrada completamente configurada

La siguiente solicitud de ejemplo crea una flota de producción con la configuración de todas las características opcionales. La nueva flota tiene las siguientes características:

- Especifica una compilación de servidor de juegos que se ha cargado en Amazon GameLift Servers y se encuentra en estado READY.
- Utiliza instancias bajo demanda c5.large con el sistema operativo necesario para la compilación del juego seleccionada.
- Establece la base de la flota us-west-2 y despliega las instancias en la región de origen y en una ubicación remota. Región de AWS sa-east-1
- Según la configuración del tiempo de ejecución:
 - Cada recurso de computación de la flota ejecuta diez procesos de servidor de juegos con los mismos parámetros de inicio, lo que significa que cada recurso puede alojar hasta diez sesiones de juego simultáneamente.
 - En cada recurso de computación, solo se pueden activar dos sesiones de juego al mismo tiempo. Las sesiones de juego en activación deben estar listas para alojar a jugadores en un plazo de 300 segundos (5 minutos); de lo contrario, finalizarán.

- Los jugadores pueden conectarse a los servidores del juego mediante un puerto del rango 33435 to 33535.
- Genera un certificado TLS para la comunicación cifrada entre el cliente y el servidor del juego.
- Todas las sesiones de juego en la flota tienen la protección de sesión de juego activada.
- Los jugadores individuales pueden crear hasta tres nuevas sesiones de juego en un período de 15 minutos.
- Las métricas de esta flota se incluyen en el grupo de métricas AMERfleets, que, para este ejemplo, agrega las métricas de un grupo de flotas en América del Norte, América Central y del América del Sur.

```
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 solicitud de creación de la flota se realiza correctamente, Amazon GameLift Servers devuelve un conjunto de atributos de flota que incluye los valores de configuración solicitados y un ID de flota nuevo. Amazon GameLift Servers inicia inmediatamente el proceso de activación de la flota y establece el estado de la flota en Nuevo. Puede realizar un seguimiento del estado de la flota y ver más información sobre la flota con estos comandos de la CLI:

- [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](#)

Puede cambiar la capacidad de la flota y otras opciones de configuración según sea necesario mediante estos comandos:

- [update-fleet-attributes](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Flotas de contenedores administradas de Amazon GameLift Servers

Las flotas de contenedores administradas por Amazon GameLift Servers proporcionan recursos basados en la nube para alojar su software de servidor de juegos en contenedores. Con una flota de contenedores gestionada, dispondrás de la flexibilidad, la seguridad y la fiabilidad de Nube de AWS los recursos, optimizados para el alojamiento de juegos multijugador. Amazon GameLift Servers proporciona herramientas sólidas de administración de hosts.

 Acelere la incorporación con estas herramientas para contenedores administrados:

- El [kit básico para contenedores](#) agiliza la integración y la configuración de flotas. Añade características esenciales de administración de sesiones de juego al servidor de juegos y utiliza plantillas preconfiguradas para crear una flota de contenedores y una canalización de implementación automatizada para el servidor de juegos. Tras la implementación, utilice la consola de Amazon GameLift Servers y las herramientas de la API para supervisar el rendimiento de la flota, administrar las sesiones de juego y analizar las métricas.

- Para los desarrolladores de Unreal Engine o Unity, usa los [Amazon GameLift Servers complementos y el servidor del motor de juego SDKs](#) para integrar tu servidor de juegos y crear una flota de contenedores desde el entorno de desarrollo de tu motor de juego. Los flujos de trabajo guiados del complemento ayudan a crear una solución rápida y sencilla con alojamiento en la nube mediante contenedores administrados. Puede tomarlos como base para crear una solución de alojamiento personalizada para su juego.

Una flota de contenedores gestionada es un conjunto de instancias de Amazon Elastic Compute Cloud (Amazon EC2) que ejecutan Linux, que Amazon GameLift Servers posee y opera en función de su configuración. Estas instancias están ubicadas físicamente en Zonas Locales Regionales de AWS o compatibles. Al crear una flota de contenedores, eliges un tipo de EC2 instancia que cumpla con los requisitos de potencia informática, memoria, almacenamiento y capacidades de red del servidor de juegos.

Para obtener una flota de contenedores administrada, almacene imágenes de contenedores basadas en Linux en un repositorio de Amazon Elastic Container Registry (Amazon ECR) y cree una definición de grupo de contenedores que describa su arquitectura de contenedores. Al crear una flota, Amazon GameLift Servers aprovisiona una instancia de flota con la última versión de la imagen de máquina de Amazon (AMI) de Linux y usa la definición del grupo de contenedores para implementar las imágenes de contenedores. Todas las instancias de una flota de contenedores utilizarán la misma versión de AMI, incluso si actualiza la definición de un grupo de contenedores o cambia la imagen de un contenedor.

Note

Como práctica recomendada, te recomendamos sustituir tus flotas cada 30 días para mantener un entorno seguro y de up-to-date ejecución para tus servidores de juegos alojados. Para ello, es necesario crear una nueva flota y migrar a ella el tráfico de jugadores. Para obtener más orientación, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Tras implementar la instancia contenerizada, los contenedores comienzan a lanzar los procesos del servidor del juego. Cada servidor de juegos establece una conexión con el servicio de Amazon GameLift Servers, informa de que está listo para alojar una sesión de juego y empieza a comunicar

su estado. Amazon GameLift Servers puede entonces solicitar al proceso del servidor que inicie una sesión de juego.

Además de la implementación de flotas, Amazon GameLift Servers administra automáticamente las siguientes tareas de administración del host por usted:

- Hace un seguimiento del estado de todos los contenedores de la flota y reemplaza los recursos obsoletos o en mal estado.
- Gestiona la autenticación para la comunicación entre los procesos del servidor y el servicio de Amazon GameLift Servers.
- Ofrece herramientas de escalado automático que ajustan la capacidad de la flota de forma dinámica para satisfacer la demanda de jugadores.
- Reporta las métricas de rendimiento de las EC2 instancias, los contenedores y los procesos de servidor de la flota.

Consulte estos temas sobre la configuración y el mantenimiento de las flotas de contenedores administradas:

- [Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers](#)
- [Creación de una flota de contenedores administrada por Amazon GameLift Servers](#)
- [Personalización de una flota de contenedores de Amazon GameLift Servers](#)
- [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#)
- [Actualización de una flota de contenedores administrada por Amazon GameLift Servers](#)

Cómo funcionan los contenedores en Amazon GameLift Servers

Las flotas de contenedores de Amazon GameLift Servers están diseñadas para ofrecerle flexibilidad a la hora de implementar y escalar sus aplicaciones contenerizadas. Utiliza Amazon Elastic Container Service (Amazon ECS) para administrar la implementación y la ejecución de tareas de sus flotas de Amazon GameLift Servers. En este tema se describen los elementos estructurales básicos para ejecutar contenedores en una flota administrada por Amazon GameLift Servers, se ilustran las arquitecturas más habituales y se describen algunos conceptos básicos.

 Acelere la incorporación con estas herramientas para contenedores administrados:

- El [kit básico para contenedores](#) agiliza la integración y la configuración de flotas. Añade características esenciales de administración de sesiones de juego al servidor de juegos y utiliza plantillas preconfiguradas para crear una flota de contenedores y una canalización de implementación automatizada para el servidor de juegos. Tras la implementación, utilice la consola de Amazon GameLift Servers y las herramientas de la API para supervisar el rendimiento de la flota, administrar las sesiones de juego y analizar las métricas.
- Los desarrolladores de Unreal Engine o Unity pueden utilizar los [complementos de Amazon GameLift Servers](#) para integrar el servidor de juegos y crear una flota de contenedores desde el entorno de desarrollo del motor de juegos. Los flujos de trabajo guiados del complemento ayudan a crear una solución rápida y sencilla con alojamiento en la nube mediante contenedores administrados. Luego tómelos como base para crear una solución de alojamiento personalizada para su juego.

Componentes de una flota de contenedores

Fleet

Una flota de contenedores es un conjunto de instancias de Amazon EC2 para alojar servidores de juegos en contenedores. Amazon GameLift Servers administra estas instancias en su nombre. Al crear una flota, se configura la forma en que se implementa la arquitectura de contenedores con el software del servidor de juegos en cada instancia de la flota. Puede crear una flota de contenedores con instancias en una o múltiples ubicaciones geográficas. Puede usar las herramientas de escalado de Amazon GameLift Servers para escalar automáticamente la capacidad de una flota de contenedores para alojar sesiones de juego y jugadores.

Instancia

Una instancia Amazon EC2 es el servidor virtual que proporciona capacidad de computación para el alojamiento de juegos. Con Amazon GameLift Servers, puede elegir entre varios tipos de instancia. Cada tipo de instancia ofrece una combinación distinta de CPU, memoria, almacenamiento y capacidad de red.

Al crear una flota de contenedores, Amazon GameLift Servers los implementa en función del tipo de instancia que elija y de la configuración de la flota. Cada instancia de flota implementada es idéntica y ejecuta el software de servidor de juegos en contenedores de la misma manera. El

número de instancias de una flota determina el tamaño de la flota y la capacidad de alojamiento de juegos.

Grupo de contenedores

Amazon GameLift Servers utiliza el concepto de grupo de contenedores para describir y administrar un conjunto de contenedores. Un grupo de contenedores es similar a una “tarea” o “pod” de contenedores. Dentro de cada grupo de contenedores, puede definir cómo se comportan los contenedores, configurar las dependencias y compartir los recursos de CPU y memoria disponibles.

Cada instancia de flota puede tener los siguientes tipos de grupos de contenedores:

- Un grupo de contenedores de servidores de juegos administra los contenedores en los que se ejecuta la aplicación del servidor de juegos y el software de soporte. Una flota de contenedores debe tener un grupo de contenedores de este tipo para alojar a los jugadores y a las sesiones de juego. Un grupo de contenedores de servidores de juegos se puede replicar en una instancia de flota. La cantidad de réplicas de grupos de servidores de juegos por instancia de flota depende de los requisitos informáticos del software y de los recursos de computación disponibles en la instancia.
- Un grupo de contenedores por instancia, que es opcional, le permite ejecutar software adicional en cada instancia de la flota. Son útiles para ejecutar servicios en segundo plano o programas de utilidades, por ejemplo, con fines de supervisión. El software del servidor de juegos no depende directamente de los procesos de un grupo por instancia. Solo se implementa una copia de un grupo de contenedores por instancia en cada instancia de la flota.

Cada grupo de contenedores de una flota de contenedores tiene un contenedor que se denomina “esencial”. Un contenedor esencial determina el ciclo de vida de un grupo de contenedores. Si un contenedor esencial produce un error, todo el grupo de contenedores se reinicia.

Container

El contenedor es el elemento más básico de una arquitectura basada en contenedores. Incluye una imagen de contenedor con ejecutables de software y archivos dependientes. Defina un contenedor para configurar la forma en que el software se ejecuta e interactúa con Amazon GameLift Servers.

Amazon GameLift Servers define dos tipos de contenedores:

- Un contenedor del servidor de juegos incluye todo lo que necesita para ejecutar los procesos del servidor de juegos y alojar sesiones de juego para los jugadores. Incluye la compilación del

servidor de juegos y el software dependiente. Defina un contenedor de servidores de juegos para el grupo de contenedores de servidores de juegos de una flota. El contenedor de servidor de juegos se considera automáticamente esencial para el grupo de contenedores.

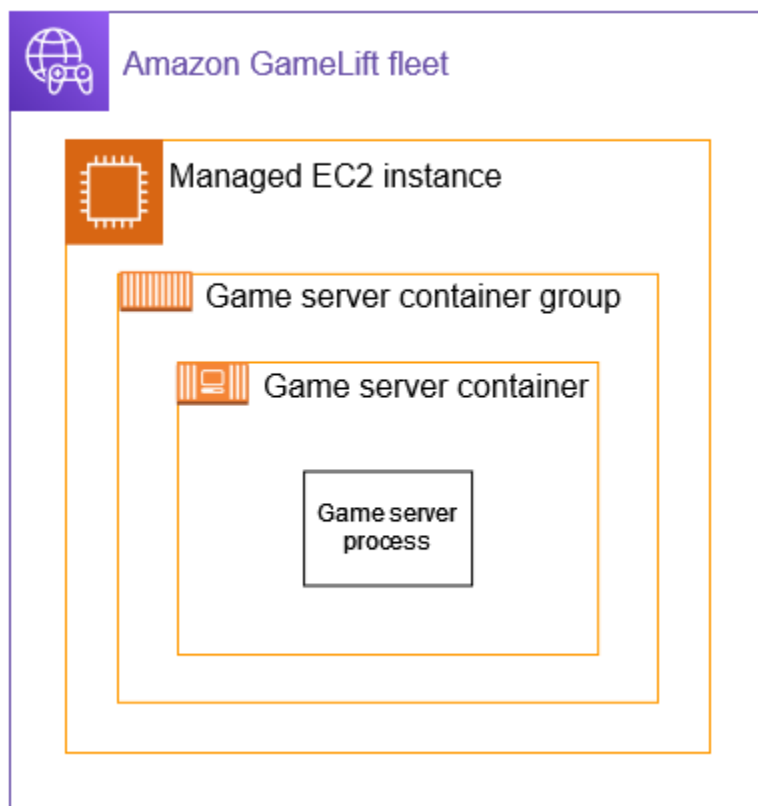
- Un contenedor de soporte ejecuta software adicional para respaldar al servidor de juegos. Es similar al concepto de contenedor lateral. Le da la opción de ejecutar y escalar el software compatible junto con los servidores de juegos, pero administrándolo como contenedores independientes. En un grupo de contenedores de servidor de juegos, puede definir cero o más contenedores de soporte. En un grupo de contenedores por instancia, todos los contenedores son contenedores de soporte. Cualquier contenedor de soporte puede designarse como esencial.

Computación

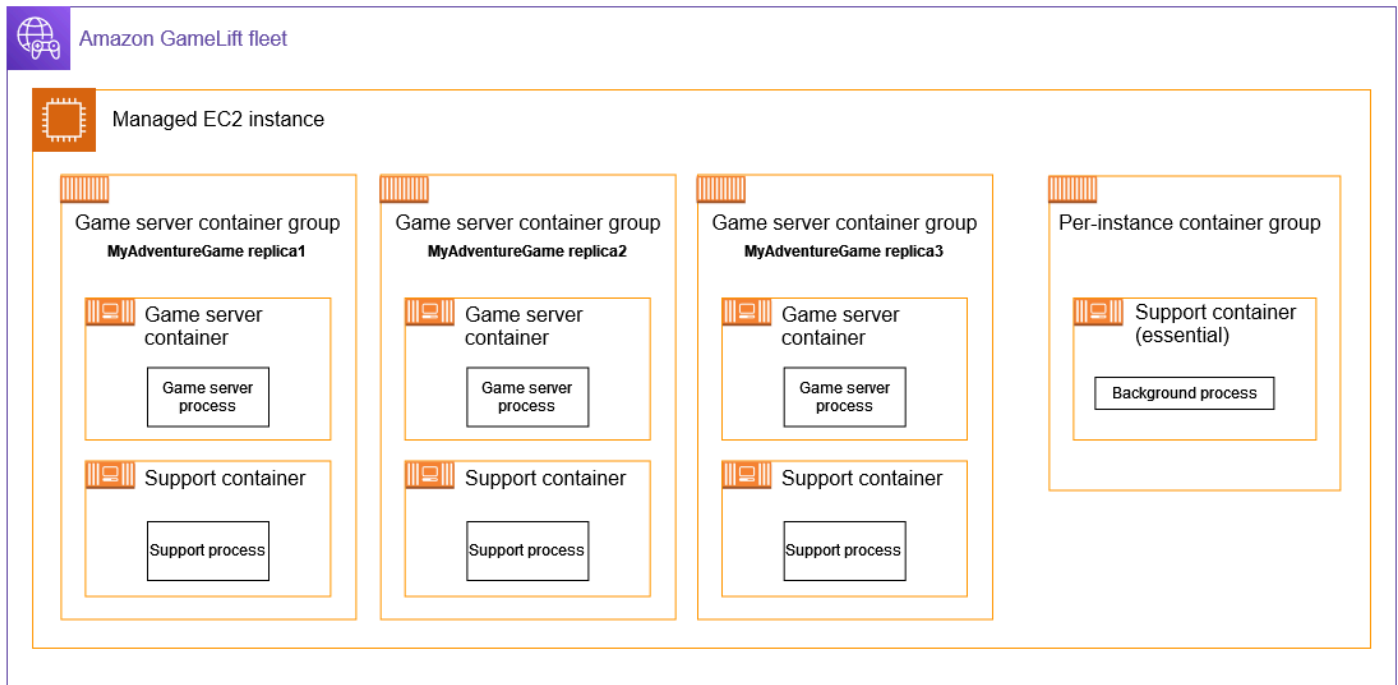
Un recurso de computación constituye una copia de un grupo de contenedores de servidor de juegos en una instancia de flota.

Arquitecturas comunes

En el siguiente diagrama se ilustra la estructura de flota de contenedores más simple. En esta estructura, cada instancia de la flota mantiene una copia del grupo de contenedores de servidor del juego. El grupo de contenedores tiene un único contenedor de servidores de juegos que ejecuta un proceso de servidor de juegos. En este ejemplo, la flota de contenedores está configurada para colocar una copia del grupo de contenedores de servidores de juegos por instancia. Con esta arquitectura, cada instancia ejecuta un proceso de servidor de juegos.



Este segundo diagrama ilustra una arquitectura de flota de contenedores más compleja. En esta estructura, la flota tiene un grupo de contenedores de servidores de juegos y un grupo de contenedores por instancia. El grupo de contenedores de servidores de juegos tiene contenedores separados para el proceso del servidor de juegos y un proceso de soporte. La flota está configurada para colocar tres copias del grupo de contenedores del servidor del juego en cada instancia de la flota. El grupo de contenedores por instancia nunca se replica. En este ejemplo, la flota de contenedores está configurada para colocar tres copias del grupo de contenedores del servidor de juegos por instancia. Con esta arquitectura, cada instancia ejecuta tres procesos del servidor de juegos.



Características principales

En esta sección se resume cómo Amazon GameLift Servers implementa algunos conceptos básicos de contenedor. Para obtener instrucciones sobre cómo trabajar con flotas de contenedores, consulte los temas correspondientes de esta guía.

Actualizaciones de flotas activas

Los contenedores administrados proporcionan soporte avanzado para ayudarle a administrar el ciclo de vida del software alojado y la arquitectura de contenedores. Puede actualizar las definiciones de sus contenedores, incluidas las imágenes de los contenedores, e implementar cambios en sus flotas actuales. Esta característica agiliza y facilita la iteración de los cambios en los contenedores durante el desarrollo. También proporciona características que le ayudan a crear, implementar y realizar un seguimiento de las actualizaciones de las versiones de software a lo largo del tiempo. Estas características incluyen:

- Administre las actualizaciones de las definiciones y el control de versiones de grupos de contenedores. Puede actualizar prácticamente todas las propiedades de una definición de grupo de contenedores, incluidas las imágenes de los contenedores y los ajustes de configuración. Cada vez que actualiza un contenedor, Amazon GameLift Servers asigna automáticamente un número de versión a la actualización y, de forma predeterminada, mantiene todas las versiones. Puede acceder a cualquier versión específica y eliminar las versiones que desee. Al configurar una

flota de contenedores, puede especificar qué versión y definición de un grupo de contenedores implementar.

- Actualice las flotas de contenedores existentes con nuevas definiciones de grupos de contenedores y ajustes de configuración. Puede implementar actualizaciones de contenedores en las flotas que ya estén implementadas en las instancias de la flota. Puede realizar un seguimiento del estado de las implementaciones de actualizaciones en cada ubicación de la flota mediante el Consola de administración de AWS, el AWS SDK y la CLI.
- Configure cómo desea que se implementen las actualizaciones de flota en una flota activa.
 - Protección de la sesión de juego. Opte por proteger las instancias de flota con sesiones de juego activas hasta que finalicen las sesiones de juego (implementación segura). También puede sustituir las instancias de flota independientemente de la actividad de las sesiones de juego (implementación no segura). Utilice las implementaciones no seguras durante las fases de desarrollo y prueba para reducir el tiempo de implementación.
 - Porcentaje de buen estado mínimo. Especifique el porcentaje de tareas en buen estado que desea mantener durante la implementación. Esta característica le permite decidir cuántas instancias de la flota se ven afectadas durante una implementación. Un valor bajo prioriza la velocidad de implementación, mientras que un valor alto garantiza que la disponibilidad del servidor de juegos se mantenga en niveles altos durante toda la implementación.
 - Estrategia en caso de error de implementación. Decida qué acciones se deben adoptar en el caso de que se produzca un error en la implementación. Un error en la implementación significa que algunos de los contenedores actualizados no han superado las comprobaciones de estado y se consideran defectuosos. Puede configurar las implementaciones para que reviertan automáticamente todas las instancias de flota al estado de implementación anterior. Como alternativa, puede optar por mantener algunas de las instancias de flota defectuosas para utilizarlas en la depuración.

La posibilidad de actualizar las flotas activas es muy útil cuando se desea implementar una actualización en el software del servidor de juegos. Una vez que haya creado una nueva imagen de contenedor para el servidor de juegos, su implementación consiste en un proceso de dos pasos: primero, actualice la definición del grupo de contenedores con la nueva imagen y, segundo, actualice la flota de contenedores. Amazon GameLift Servers administra las demás tareas según sea necesario.

Embalaje de contenedores

Al desarrollar la estructura de contenedores para su implementación en una flota de contenedores, un objetivo común es optimizar el uso de la potencia informática disponible. Para lograr este objetivo, debe agrupar tantos grupos de contenedores de servidores de juegos como sea posible en cada instancia de flota.

Amazon GameLift Servers le ayuda a conseguirlo calculando el número máximo de grupos de contenedores de servidores de juegos por instancia, en función de la siguiente información:

- El tipo de instancia de la flota y sus recursos de vCPU y memoria.
- Los requisitos de vCPU y memoria para todos los contenedores del grupo de contenedores del servidor de juegos.

Los requisitos de vCPU y memoria para todos los contenedores de un grupo de contenedores por instancia, si lo hay.

Al crear una flota de contenedores, puede usar el máximo calculado o puede especificar el número deseado. Como práctica recomendada, planifique experimentar con su software de servidor de juegos en contenedores para determinar los requisitos de recursos para un rendimiento óptimo del servidor de juegos.

Escalado de capacidad

La capacidad de flota mide el número de sesiones de juego que una flota puede alojar simultáneamente. También puede medir la capacidad en función del número de jugadores simultáneos que puede soportar una flota. Para aumentar o reducir la capacidad de alojamiento de una flota, debe añadir o eliminar instancias de la flota.

Una flota de contenedores está configurada para ejecutar un número específico de procesos simultáneos del servidor de juegos en cada instancia de la flota. (Puede calcularlo en (1) los grupos de contenedores de servidores de juegos por instancia y en (2) el número de procesos del servidor de juegos que se ejecutan en cada grupo de contenedores.) El número de servidores de juegos simultáneos por instancia indica cuál es el impacto de añadir o eliminar cada instancia de la flota. Por ejemplo, si tu flota de contenedores ejecuta un proceso de servidor de juego en cada grupo de contenedores de servidores de juegos y cada instancia de flota contiene 100 grupos de contenedores de servidores de juegos, aumentas o disminuyes la capacidad de tu flota para albergar sesiones de juego simultáneas en incrementos de 100. Si cada sesión de juego tiene 10 plazas para

jugadores, aumentarás o disminuirás la capacidad de tu flota para alojar jugadores en incrementos de 1000.

Con las flotas de contenedores, puede utilizar cualquiera de los métodos de escalado de capacidad que ofrecen Amazon GameLift Servers. Entre ellos se incluyen:

- Establezca la capacidad de la flota manualmente configurando el recuento de instancias de flota deseado.
- Configure el escalado automático centrándose en el búfer deseado de instancias disponibles (seguimiento de objetivos). Este método mantiene automáticamente una cierta cantidad de recursos de alojamiento inactivos para que los jugadores entrantes puedan acceder rápidamente a las partidas. A medida que la demanda de los jugadores aumenta o disminuye, el tamaño de este búfer se ajusta continuamente.
- Configura el escalado automático con reglas de escalado personalizadas (característica avanzada). Este método te permite escalar en función de las métricas de flota que elijas.

Conexiones de juegos client/server

Con las flotas administradas por Amazon GameLift Servers, los clientes de juegos se conectan directamente a los servidores de juegos alojados en la nube. Cuando un cliente de juego solicita unirse a una partida, Amazon GameLift Servers busca una sesión de juego y proporciona información de conexión (IP y puerto) al cliente de juego. Puede controlar el acceso externo a las instancias de la flota abriendo ciertos rangos de puertos (permisos de entrada) para la flota. Los permisos de entrada determinan qué puertos están abiertos al tráfico entrante. Puede cerrar rápidamente todos los puertos, limitarlos a unos pocos o abrir todos los puertos.

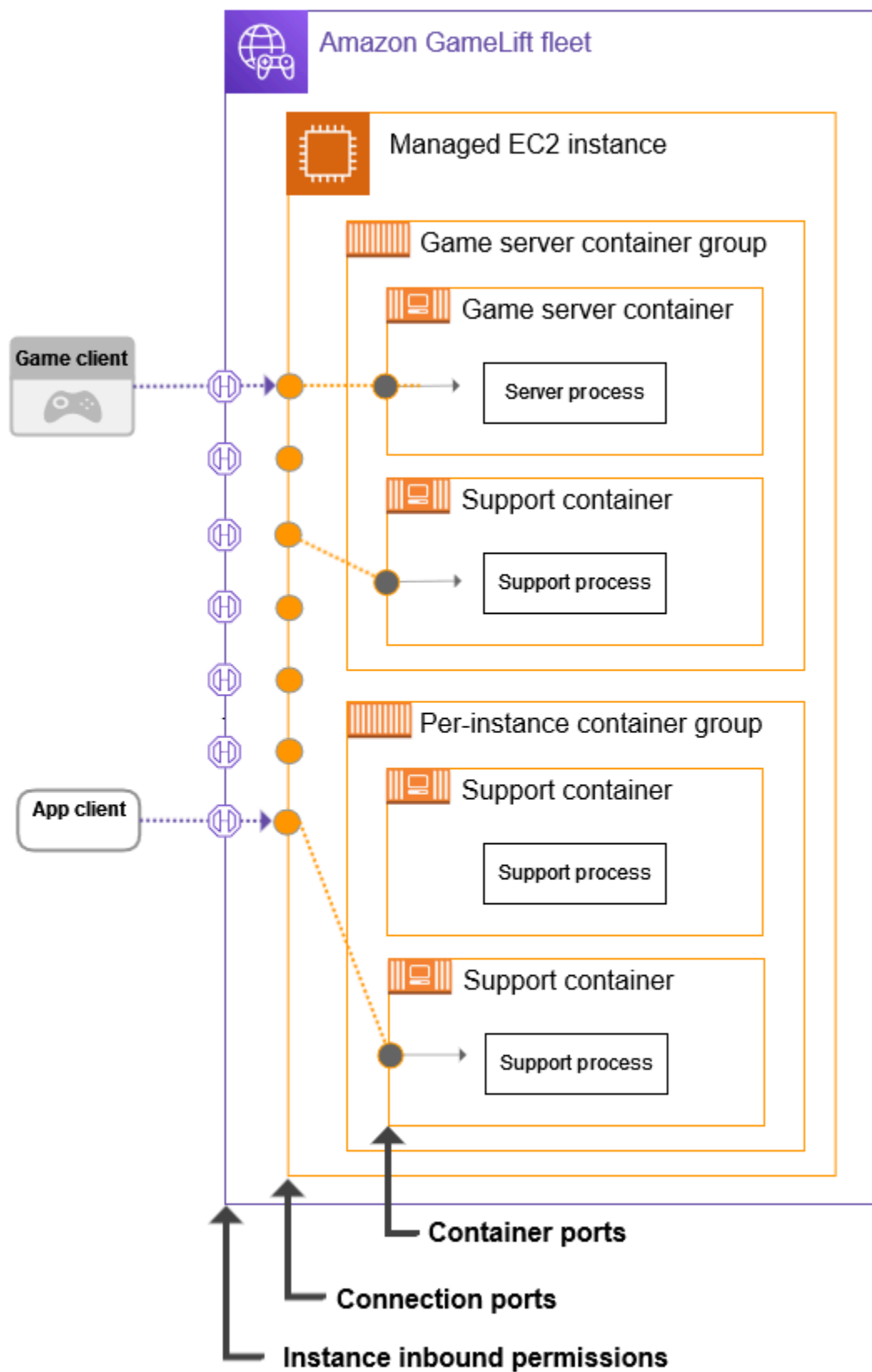
Las flotas de contenedores administradas requieren una configuración adicional que permita el acceso a los procesos que se ejecutan en un contenedor. Al crear una definición de contenedor, se especifica un conjunto de puertos, uno para cada proceso que requiere una conexión. Esto incluye:

- Todos los procesos del servidor de juegos que se ejecutarán simultáneamente en el contenedor del servidor de juegos. Todos los procesos del servidor de juegos deben permitir que los clientes del juego se conecten una sesión de juego.
- Cualquier proceso de un contenedor de soporte al que deba conectarse una fuente externa. Por ejemplo, puede conectarse de forma remota a una aplicación de prueba.

Cuando establece los ajustes de puerto del contenedor orientado al interior, Amazon GameLift Servers los utiliza para calcular los permisos de entrada externos a los que se pueden conectar los clientes de juego y otras aplicaciones. Amazon GameLift Servers también administra el mapeo entre los permisos de entrada y los puertos de contenedores individuales, lo que permite a los jugadores acceder a una sesión de juego en un contenedor. Este mapeo interno proporciona un nivel de seguridad al proteger los servidores de juegos del acceso directo a los puertos de los contenedores. Tienes la opción de personalizar la configuración de los puertos externos de la flota según sea necesario. Para obtener más información sobre cómo configurar manualmente los puertos de la flota de contenedores, consulte [Configuración de conexiones de red](#).

Puede modificar la configuración del puerto de una flota de contenedores en cualquier momento. Este cambio requiere la implementación de una actualización de la flota.

En el siguiente diagrama, se ilustra el papel de las conexiones de puertos en una flota de contenedores. Como se muestra, se configuran los puertos en contenedores individuales, y Amazon GameLift Servers utiliza esta información para configurar suficientes puertos en la instancia de la flota como para asignarlos a cada puerto de contenedores. Tanto los permisos de entrada de las instancias externas como los puertos de conexión internos de la flota los calcula Amazon GameLift Servers, a menos que elija configurarlos manualmente.



Registro del contenedor

En las flotas de contenedores administradas, se capturan los flujos de salida estándar (y de error estándar) de todos los contenedores. Esto incluye los registros de las sesiones de juego del servidor de juegos. Puede configurar una flota de contenedores para utilizar una de las siguientes opciones para administrar los flujos de salida:

- Guarda la salida del contenedor como un flujo de CloudWatch registro de Amazon. Cada flujo de registro hace referencia al identificador de la flota y al contenedor. Si elige esta opción de registro para la flota, especifica un grupo de CloudWatch registros que organiza todos los flujos de registros de la flota. A continuación, puede utilizar CloudWatch las funciones para buscar y analizar los datos de registro según sea necesario.
- Almacene la salida del contenedor en un bucket de Amazon Simple Storage Service (Amazon S3). Puede ver, compartir o descargar el contenido según lo necesite.
- Desactive el registro. En este escenario, la salida del contenedor no se guarda.

Amazon GameLift Servers envía los datos de registro de las flotas de contenedores gestionadas a los CloudWatch servicios de Amazon S3 de su AWS cuenta. Para ver sus datos, utilice estas Consola de administración de AWS u otras herramientas iniciando sesión en su AWS cuenta y trabajando con los servicios individuales. Para ampliar el acceso limitado a Amazon GameLift Servers para emprender estas acciones, puede crear un rol de servicio para sus flotas de contenedores.

Puede modificar la configuración de registro de una flota de contenedores en cualquier momento. Este cambio requiere la implementación de una actualización de la flota.

Las flotas de contenedores y el agente Amazon GameLift Servers

Una arquitectura de contenedores de uso común ejecuta un único proceso en cada contenedor. En una flota de contenedores de Amazon GameLift Servers, el grupo de contenedores de servidores de juegos tiene un contenedor de servidores de juegos, que ejecuta un proceso de servidor de juegos. Con esta arquitectura, Amazon GameLift Servers administra el ciclo de vida del proceso de un único servidor de juego en cada grupo de contenedores de servidores de juegos de una instancia de flota.

Si opta por crear una arquitectura de contenedores que ejecute varios procesos de servidores de juegos en cada grupo de contenedores de servidores de juegos, necesitará una forma de administrar el ciclo de vida de todos los procesos. Esto incluye tareas como iniciar, cerrar y reemplazar los procesos según sea necesario, administrar el número deseado de procesos para que se ejecuten simultáneamente y gestionar los estados de fallo.

Puede optar por utilizar el agente de Amazon GameLift Servers para estas tareas. Para una flota de contenedores, el agente implementa instrucciones de tiempo de ejecución que especifican qué ejecutables se deben ejecutar (y cuántos), proporcionan los parámetros de lanzamiento y establecen reglas sobre la activación del servidor del juego. Por ejemplo, las instrucciones de tiempo de ejecución pueden indicar al agente que mantenga diez procesos del servidor de juegos para su uso en producción y un proceso del servidor de juegos con parámetros de lanzamiento especiales para las pruebas.

Para usar el agente con sus flotas de contenedores, añada el agente a la imagen del contenedor e incluya un conjunto de instrucciones de tiempo de ejecución. Para obtener más información sobre el agente, consulte [Uso el agente de Amazon GameLift Servers](#).

Creación de una flota de contenedores administrada por Amazon GameLift Servers

Cree una flota de contenedores administrada por Amazon GameLift Servers para implementar y alojar su servidor de juegos en contenedores en la nube de AWS. Al crear una flota de contenedores, especifique definiciones de grupos de contenedores que especifiquen una o más imágenes de contenedor (al menos una que incluya la compilación del servidor de juegos) y los ajustes de configuración.

Al crear un nuevo recurso de flota de contenedores administrada, se inicia inmediatamente la primera fase de creación de la flota. La creación de flotas administradas pasa por varias fases a medida que Amazon GameLift Servers implementa una instancia de EC2, instala un entorno de tiempo de ejecución, implementa los grupos de contenedores en la instancia e inicia el lanzamiento de los procesos del servidor de juegos. Según el entorno de tiempo de ejecución que requiera la compilación del servidor de juegos, Amazon GameLift Servers implementa la versión más reciente de la imagen de máquina de Amazon (AMI) en el momento de crear la flota (y todas las instancias futuras de la flota utilizarán la misma versión). Puede supervisar el estado de una flota en la consola o mediante la AWS Command Line Interface (AWS CLI). Cuando la flota está lista para alojar sesiones de juego, el estado cambia a ACTIVE. Para obtener ayuda sobre problemas de creación de flotas, consulte [Depuración de problemas relacionados con flotas de Amazon GameLift Servers](#).

Puede optar por crear una flota de contenedores vacía y, posteriormente, añadir o actualizar las definiciones de grupos de contenedores de la flota. Si crea una flota sin una definición de grupo de contenedores, la flota no alcanzará el estado activo.

 Note

Como práctica recomendada, es aconsejable reemplazar las flotas cada 30 días para mantener un entorno de tiempo de ejecución seguro y actualizado para los servidores de juegos alojados. Para ello, es necesario crear una nueva flota y migrar a ella el tráfico de jugadores. Para obtener más orientación, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Utilice la [consola de Amazon GameLift Servers](#) o la AWS Command Line Interface (AWS CLI) para crear una flota de contenedores.

Console

En la [consola de Amazon GameLift Servers](#), seleccione la Región de AWS en la que desea crear la flota. Las definiciones de grupos de contenedores deben residir en la misma región en la que se desea crear la flota.

Abra la barra de navegación izquierda de la consola y elija Contenedores administrados: flotas. En la página Flotas, elija Crear flota de contenedores.

Paso 1: Definir los detalles de la flota de contenedores administrada

1. En la sección Detalles de la flota de contenedores, introduzca una descripción de flota.
2. Especifique un rol de IAM para la flota. Este rol incluye los permisos que debe tener Amazon GameLift Servers para administrar la flota de contenedores en su nombre. Para obtener ayuda para crear el rol de servicio necesario, consulte [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).
3. En Configuración de registro, elija una opción. De forma predeterminada, la opción seleccionada es CloudWatch. Proporcione la información necesaria en función de la opción seleccionada.
4. Añada grupos de contenedores a la flota. Se trata de un paso opcional. Puede optar por crear una flota sin un grupo de contenedores con previsiones de añadirlos más adelante. Una flota sin grupos de contenedores no implementará ninguna instancia de flota ni podrá alojar ningún juego aún, pero se creará el recurso de flota.

- Seleccione una definición de grupo de contenedores del servidor de juegos. Si lo desea, especifique la versión de la definición que desea implementar. Si no especifica un número de versión, Amazon GameLift Servers utiliza automáticamente la versión más reciente.
 - Si lo desea, añada una versión y una definición de grupo de contenedores por instancia. Si no especifica un número de versión, Amazon GameLift Servers utiliza automáticamente la versión más reciente.
5. En la sección Detalles adicionales, puede configurar algunas personalizaciones opcionales. Ninguno de estos ajustes es necesario para crear la flota de contenedores.

Paso 2: Definir los detalles de la instancia

1. En Implementación de instancias, seleccione una o más ubicaciones remotas en las que implementar instancias. La región de origen se selecciona automáticamente (es la región en la que va a crear la flota). Si selecciona ubicaciones adicionales, las instancias de flota también se implementarán en estas ubicaciones.

Important

Para utilizar las regiones que no estén habilitadas de forma predeterminada, actívelas en su Cuenta de AWS.

- Las flotas con regiones que no están habilitadas, y que haya creado antes del 28 de febrero de 2022, no se verán afectadas por este requisito.
- Para crear nuevas flotas con múltiples ubicaciones o actualizar las existentes, habilite primero las regiones o zonas locales que desee utilizar.

Para obtener más información sobre las regiones que no están habilitadas de forma predeterminada y cómo habilitarlas, consulte [Administración de Regiones de AWS](#) en la Referencia general de AWS. Consulte [Introducción al uso de zonas locales](#) en la Guía del usuario de zonas locales de AWS.

2. Seleccione una Configuración de instancia para la flota. La consola calcula automáticamente la vCPU y la memoria mínimas necesarias (en función de los límites totales que establezca para cada grupo de contenedores). Filtra la lista completa de tipos de instancia disponibles en función de los requisitos de recursos y de las ubicaciones que hayas introducido. Puede añadir filtros adicionales según sea necesario.

Para obtener más información sobre cómo elegir un tipo de instancia, consulte [Configure una flota de contenedores](#). El tamaño del tipo de instancia que elija afectará a la forma en que se empaquetan los grupos de contenedores del servidor de juegos en cada instancia de la flota. En función de su selección, considere la posibilidad de revisar la configuración de los grupos de contenedores de servidor de juegos que desee por instancia.

Paso 4: Revisar y crear

- Revise los ajustes de configuración de flota.

Puede actualizar los metadatos de la flota y la configuración en cualquier momento, independientemente del estado de la flota. Para obtener más información, consulte [Actualización de la configuración de una flota de Amazon GameLift Servers](#). Puede actualizar la capacidad de la flota después de que la flota haya alcanzado el estado ACTIVO. Para obtener más información, consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#). También puede añadir ubicaciones remotas o eliminarlas.

Cuando haya terminado, elija Crear.

Si la solicitud se realiza correctamente, la consola muestra la página de detalles del nuevo recurso de flota. Inicialmente, el estado es NEW, ya que Amazon GameLift Servers comienza el proceso de creación de la flota. Puede hacer un seguimiento del estado de la flota nueva en la página Flotas. La flota estará lista para alojar sesiones de juego cuando alcance el estado ACTIVE.

AWS CLI

Para crear una flota de contenedores con la AWS CLI, abra una ventana de línea de comandos y utilice el comando `create-container-fleet`. Para obtener más información sobre este comando, consulte [create-container-fleet](#) en la Referencia de comandos de la AWS CLI.

La solicitud `create-container-fleet` de ejemplo mostrada a continuación crea una nueva flota de contenedores con las características siguientes:

- `ContainerGroupsConfiguration` especifica únicamente una definición de grupo de contenedores de servidor de juegos: `MyAdventureGameContainerGroup`. El número de grupos de contenedores de servidor de juegos que se implementará en cada instancia de flota lo calcula Amazon GameLift Servers.

- La flota utiliza instancias bajo demanda c5.large de forma predeterminada.
- También de forma predeterminada, la flota abre un conjunto de puertos de conexión y puertos de permisos de entrada calculados por Amazon GameLift Servers. Implementa grupos de contenedores en las siguientes ubicaciones:

```
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" \  

```

Si la solicitud de creación de la flota se realiza correctamente, Amazon GameLift Servers devuelve un conjunto de atributos de flota que incluye los valores de configuración solicitados y un nuevo ID de flota de contenedores. Amazon GameLift Servers establece entonces el estado de la flota y los estados de las ubicaciones en Nuevo e inicia el proceso de activación de la flota. Puede realizar un seguimiento del estado de la flota y ver más información sobre la flota con estos comandos de la CLI:

- [describe-fleet-events](#)
- [describe-container-fleet](#)
- [describe-fleet-capacity](#)
- [describe-fleet-port-settings](#)
- [describe-fleet-utilization](#)
- [describe-fleet-location-capacity](#)
- [describe-fleet-location-utilization](#)

Puede cambiar la capacidad de la flota y otras opciones de configuración según sea necesario mediante estos comandos:

- [update-container-fleet](#)
- [update-fleet-capacity](#)
- [update-fleet-port-settings](#)
- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Creación de una definición de grupo de contenedores para una flota de contenedores de Amazon GameLift Servers

Una definición de grupo de contenedores describe cómo implementar las aplicaciones de servidor de juegos en contenedores en una flota de contenedores. Se trata de un esquema que indica a Amazon GameLift Servers qué imágenes de contenedor deben implementarse en la flota y cómo utilizarlas. Cuando se crea una flota de contenedores, se especifican las definiciones de grupo de contenedores que se van a implementar en la flota. Para obtener más información sobre los grupos de contenedores, consulte [Componentes de una flota de contenedores](#).

Antes de comenzar

Consejos sobre qué hacer antes de empezar a crear una definición de grupo de contenedores:

- Finalice las imágenes de sus contenedores y envíelas a un repositorio de Amazon Elastic Container Registry (Amazon ECR) en el mismo Región de AWS lugar en el que planea crear el grupo de contenedores. Amazon GameLift Servers captura una instantánea de cada imagen al crear la definición del grupo de contenedores y la utiliza al desplegarla en una flota de contenedores. Consulte [Creación de una imagen de contenedor para Amazon GameLift Servers](#).
- Cree sus definiciones de contenedor como archivos JSON. Una definición de grupo de contenedores incluye una o más definiciones de contenedores. Puede usar los archivos JSON si crea una definición de grupo de contenedores con AWS CLI for Amazon GameLift Servers.
- Compruebe que el AWS usuario tiene permisos de IAM para acceder al repositorio de Amazon ECR. Consulte [Ejemplo de permisos de IAM para Amazon GameLift Servers](#).

Creación de una definición de grupo de contenedores de servidor de juegos

Un grupo de contenedores de servidores de juegos ejecuta su software de servidor de juegos. Un grupo de contenedores de servidores de juegos tiene un contenedor de servidores de juegos, que ejecuta el ejecutable del servidor de juegos. También puede tener uno o más contenedores de soporte para ejecutar software adicional que sirva de soporte al servidor de juegos. (A veces se los denomina contenedores laterales).

En este tema se describe cómo crear una definición sencilla de grupo de contenedores de servidores de juegos mediante la Amazon GameLift Servers consola o las herramientas AWS CLI. Para obtener información más detallada sobre las características opcionales, consulte [Personalización de una flota de contenedores de Amazon GameLift Servers](#).

 Note

Puede cambiar la mayoría de las definiciones de grupos de contenedores y los ajustes de definición de contenedores después de crearlos. Si realiza cambios en la definición de un contenedor, Amazon GameLift Servers captura una nueva instantánea de las imágenes del contenedor actualizadas.

Para crear una definición sencilla de grupo de contenedores de servidor de juegos:

En las siguientes instrucciones, se describe cómo crear una definición de grupo de contenedores con los parámetros mínimos requeridos y mediante los valores por defecto de Amazon GameLift Servers.

Console

En la [Amazon GameLift Servers consola](#), seleccione el Región de AWS lugar en el que desee crear el grupo de contenedores.

Abra la barra de navegación izquierda de la consola y elija Contenedores administrados: definiciones de grupos. En la página de definición de grupos de contenedores, elija Crear definición de grupo.

Paso 1: Definición de los detalles de la definición del grupo de contenedores

1. Introduzca un nombre de definición de grupo de contenedores. El nombre debe ser único en la región y la Cuenta de AWS .
2. Seleccione el tipo de grupo de contenedores Servidor de juegos.
3. En Límite total de memoria, introduzca los recursos de memoria máximos que estarán disponibles para todos los contenedores del grupo de contenedores. Si necesita ayuda para calcular este valor, consulte [Establecimiento de los límites de recursos](#).
4. En Límite total de vCPU, introduzca la potencia de cálculo máxima que estará disponible para todos los contenedores del grupo de contenedores. Si necesita ayuda para calcular este valor, consulte [Establecimiento de los límites de recursos](#).

Paso 2: Adición de definiciones de contenedor

Como mínimo, un grupo de contenedores de servidores de juegos tiene un contenedor de servidores de juegos. En la consola, la primera definición de contenedor que se crea es el

contenedor del servidor de juegos. En este paso se describe cómo definir la configuración mínima requerida para una definición de contenedor de servidor de juegos.

1. Introduzca un nombre para la definición del grupo de contenedores. Cada contenedor definido para el grupo debe tener un valor de nombre único.
2. Enlace a una imagen de contenedor con la compilación del servidor de juegos. Introduzca la URI de imagen de Amazon ECR para una imagen de contenedor en un repositorio público o privado. Puede usar cualquiera de los siguientes formatos:
 - Solo URI de imagen: [Cuenta de AWS].dkr.ecr.[Región de AWS].amazonaws.com/[repository ID]
 - URI de imagen + resumen: [Cuenta de AWS].dkr.ecr.[Región de AWS].amazonaws.com/[repository ID]@[digest]
 - URI de imagen + etiqueta: [Cuenta de AWS].dkr.ecr.[Región de AWS].amazonaws.com/[repository ID]:[tag]
3. Especifica la versión del SDK del servidor de Amazon GameLift Servers que utiliza la compilación del servidor del juego. Para una flota de contenedores, este valor debe ser 5.2.0 o superior.
4. En Rango de puertos de contenedores interno, configure el protocolo y defina un rango de puertos. El tamaño del rango debe ser superior al número de procesos simultáneos del servidor de juegos que se ejecutarán en este contenedor. Si el contenedor del servidor de juegos ejecuta solo un proceso de servidor por contenedor, este rango de puertos solo necesita unos pocos puertos. Para obtener más información, consulte [Configuración de conexiones de red](#).
5. Añada más contenedores según sea necesario para ejecutar software de soporte adicional. Los contenedores adicionales se designan automáticamente como contenedores de soporte. Un grupo de contenedores de servidores de juegos solo puede tener un contenedor de servidores de juegos y hasta ocho contenedores de soporte. Proporcione los siguientes ajustes mínimos necesarios:
 - Nombre de la definición de contenedor
 - URI de la imagen de ECR.
 - Puertos de contenedores internos (inclúyalos solo si el contenedor tiene procesos que necesiten acceso a la red).

Paso 3: Configuración de las dependencias

- Si la definición de su grupo de contenedores tiene más de un contenedor, si lo desea, puede establecer dependencias entre los contenedores. Para obtener más información, consulte [Establecimiento de dependencias de contenedores](#).

Paso 3: Revisar y crear

1. Revise todos los ajustes de la definición de grupo de contenedores. Utilice Editar para realizar cambios en cualquier sección, incluidas todas las definiciones de contenedores del grupo.
2. Cuando haya terminado, elija Crear.

Si la solicitud se realiza correctamente, la consola muestra la página de detalles del nuevo recurso de definición de grupo de contenedores. Inicialmente el estado es COPYING, cuando Amazon GameLift Servers comienza a tomar instantáneas de todas las imágenes de contenedor del grupo. Cuando se completa esta fase, el estado de la definición del grupo de contenedores cambia a READY. Una definición de grupo de contenedores debe tener el estado READY para poder crear una flota de contenedores con ella.

AWS CLI

Cuando utilice la AWS CLI para crear una definición de grupo de contenedores, mantenga las configuraciones de definición de contenedores en un JSON archivo independiente. Puede hacer referencia al archivo en el comando de la CLI. Consulte [Creación de un archivo JSON de definición de contenedor](#) para ver esquemas de ejemplo.

Creación de una definición de grupo de contenedores

Para crear una nueva definición de grupo de contenedores, utilice el comando `create-container-group-definition` de la CLI. Para obtener más información sobre este comando, consulte la Referencia [create-container-group-definition](#) de comandos de la AWS CLI.

Este ejemplo ilustra una solicitud de definición de grupo de contenedores de servidor de juegos. Presupone que se ha creado un archivo JSON con las definiciones de contenedor de este grupo.

```
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
```

Creación de un archivo **JSON** de definición de contenedor

Al crear una definición de grupo de contenedores, también define los contenedores del grupo. Una definición de contenedor especifica el repositorio de Amazon ECR en el que se almacena la imagen del contenedor y las configuraciones opcionales para los puertos de red, los límites de uso de CPU y memoria y otros ajustes. Recomendamos crear un único archivo JSON con las configuraciones de todos los contenedores de un grupo de contenedores. Mantener un archivo es útil para almacenar, compartir y realizar un seguimiento de las versiones de estas configuraciones críticas. Si usa la AWS CLI para crear sus definiciones de grupos de contenedores, puede hacer referencia al archivo en el comando.

Para crear una definición de contenedor

1. Cree y abra un nuevo archivo .JSON. Por ejemplo:

```
[~/work/glc]$ vim SimpleServer.json
```

2. Cree una definición de contenedor independiente para cada uno de los contenedores del grupo. Copie el siguiente contenido de ejemplo y modifíquelo según sea necesario para sus contenedores. Para obtener más información sobre la sintaxis de una definición de contenedor, consulte [ContainerDefinitionInput](#) la referencia de la Amazon GameLift Servers API.
3. Guarde el archivo localmente para poder consultarlo en un comando de AWS CLI.

Ejemplo: Definición de contenedor de servidor de juegos

Example

En este ejemplo se describe el contenedor esencial para el grupo de contenedores de servidores de juegos. El contenedor de réplicas esencial incluye la aplicación de servidor de juegos, el agente de Amazon GameLift Servers, y puede incluir otro software auxiliar para el alojamiento del juego. La definición debe incluir un nombre, un URI de imagen y una configuración de puertos. Este ejemplo también establece algunos límites de recursos específicos del contenedor.

```
{
  "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"
}
```

Flotas de Amazon GameLift Servers Anywhere

Utilice las flotas de Anywhere cuando desee utilizar las características de Amazon GameLift Servers con sus propios recursos de alojamiento. Las flotas de Anywhere se utilizan habitualmente como entornos de prueba para el desarrollo iterativo, o junto con flotas administradas en una solución de alojamiento híbrida.

Una flota de Anywhere consiste en un conjunto de recursos de computación (virtuales o físicos) que usted suministra y administra. Los recursos de computación pueden residir en cualquier ubicación geográfica con conectividad, desde un portátil local hasta outposts remotos. Al configurar una flota de Anywhere, añada recursos de computación a la flota registrándolos a través de Amazon GameLift Servers. Cada recurso de computación se registra con su dirección IP (o nombre DNS) para que Amazon GameLift Servers pueda establecer una conexión con él.

Para implementar el software de un servidor de juegos en una flota de Anywhere, debe instalarlo en cada recurso de computación e iniciar los procesos del servidor de juegos. Cada proceso de servidor de juegos que se inicia establece una conexión con el servicio de Amazon GameLift Servers e indica que está listo para alojar una sesión de juego. Puede usar las herramientas de implementación y administración de la configuración existentes para gestionar las tareas iniciales de implementación y administración del alojamiento. Hay algunas tareas adicionales necesarias para el uso con Amazon GameLift Servers, como las siguientes:

- Registrar y anular el registro de recursos de computación para añadirlos a la flota o para quitarlos de esta.

- Mantenga los identificadores de up-to-date autenticación en todos los ordenadores. Los procesos de servidor del recurso de computación los utilizan para conectarse al servicio de Amazon GameLift Servers.

 Note

Si lo desea, puede implementar su flota de Anywhere con el agente de Amazon GameLift Servers para automatizar estas tareas clave de administración. Consulte [Uso el agente de Amazon GameLift Servers](#).

Consulte estos temas sobre la configuración y el mantenimiento de las flotas de Anywhere:

- [Hoja de ruta de desarrollo para el alojamiento con Amazon GameLift Servers Anywhere](#)
- [Hoja de ruta de desarrollo para el alojamiento híbrido con Amazon GameLift Servers](#)
- [Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere](#)
- [Creación de una flota de Amazon GameLift Servers Anywhere](#)
- [Actualización de la configuración de una flota de Amazon GameLift Servers](#)

Flujo de trabajo de creación de una flota de Anywhere

En el caso de las flotas de Anywhere, Amazon GameLift Servers configura únicamente el recurso de flota. Deberá configurar y registrar los recursos de computación en la flota, así como instalar el software del servidor de juegos e iniciar los procesos del servidor de juegos para alojar las sesiones de juego.

1. Amazon GameLift Servers crea un recurso de flota en la región de origen de la flota. El estado de la flota y el estado de la ubicación personalizada se establecen en Nuevo.
2. Amazon GameLift Servers comienza a escribir los eventos en el registro de eventos de la flota.
3. Una vez creado el recurso de flota, Amazon GameLift Servers establece el estado de la flota en Activa. En este momento, ya puede registrar nuevos recursos de computación en la flota.

Creación de una flota de Amazon GameLift Servers Anywhere

En este tema se explica cómo crear una flota de Amazon GameLift Servers Anywhere. Con una flota de Anywhere, puede utilizar las características principales de administración de sesiones de juego

de Amazon GameLift Servers y, al mismo tiempo, alojar sesiones de juego con sus propios recursos de computación. Cree una flota de Anywhere para su hardware en las instalaciones u otros recursos basados en la nube.

Normalmente, las flotas de Anywhere se utilizan junto con las flotas administradas de Amazon GameLift Servers en una solución de alojamiento híbrida. También proporcionan entornos de prueba útiles a la hora de desarrollar un juego para alojarlo en Amazon GameLift Servers. Consulte estos temas para obtener más información sobre cuándo y cómo incorporar flotas de Amazon GameLift Servers Anywhere a una solución de alojamiento de juegos:

- [Alojamiento híbrido](#)
- [Implementación de flotas de alojamiento para Amazon GameLift Servers](#)
- [Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere](#)

Como las flotas de Anywhere están autoadministradas, la configuración de una flota requiere algo de trabajo adicional. A fin de que una flota de Anywhere esté lista para alojar jugadores y sesiones de juego, deberá completar las siguientes tareas:

Temas

- [Antes de comenzar](#)
- [Creación de una ubicación personalizada](#)
- [Creación de una flota de Anywhere](#)
- [Añadir un recurso de computación a la flota](#)
- [Iniciar un servidor de juegos](#)

Antes de comenzar

Antes de crear una flota de Anywhere, lleve a cabo las siguientes tareas. Para obtener una guía más detallada, consulte [Hoja de ruta de desarrollo para el alojamiento con Amazon GameLift Servers Anywhere](#) o [Hoja de ruta de desarrollo para el alojamiento híbrido con Amazon GameLift Servers](#).

- Integre el código del servidor de juegos con la versión 5.x (o superior) del SDK del servidor de Amazon GameLift Servers. No necesita completar todas las tareas de integración del juego, solo las necesarias para crear una compilación de juegos. Un procedimiento habitual es configurar su equipo local como una flota de Anywhere y utilizar una interfaz de la línea de comandos para

probar la integración del servidor de juegos (consulte [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#)). Puede incorporar componentes adicionales (como un cliente de juegos compatible con Amazon GameLift Servers) a medida que los desarrolle.

- Empaquete el software del servidor de juegos para instalarlo en los recursos de computación de su flota de Anywhere. El paquete debe incluir la compilación integrada del servidor de juegos y todo el software de soporte necesario para ejecutarlo.
- Decida si desea utilizar el agente de Amazon GameLift Servers con la flota de Anywhere. El agente es una herramienta de administración de procesos en los recursos de computación que automatiza algunas de las tareas clave relacionadas con la administración de los procesos y los cálculos del servidor para el uso con Amazon GameLift Servers. Para obtener más información, consulte [Uso el agente de Amazon GameLift Servers](#).

Creación de una ubicación personalizada

Cree una ubicación personalizada para representar la ubicación física de los recursos de computación. Al crear una flota de Anywhere, debe tener al menos una ubicación personalizada ya definida. Puede crear ubicaciones personalizadas adicionales y añadirlas a una flota existente en cualquier momento.

Creación de una ubicación personalizada

Utilice la consola de Amazon GameLift Servers o la AWS Command Line Interface (AWS CLI) para crear una ubicación personalizada.

Console

En la [consola de Amazon GameLift Servers](#), use el panel de navegación para abrir la página Ubicaciones. Elija Crear ubicación para abrir el cuadro de diálogo Crear.

1. En el cuadro de diálogo, escriba un Nombre de ubicación. Una práctica recomendada consiste en utilizar un nombre que describa una ubicación significativa para un conjunto de recursos de computación. Pueden ser ubicaciones geográficas, el nombre de un centro de datos u otro identificador de ubicación. Amazon GameLift Servers añade al nombre de la ubicación personalizada el prefijo custom-.
2. (Opcional) Añada etiquetas a la ubicación personalizada. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Asigne etiquetas a los recursos de AWS que desee categorizar y hágalo de un modo que resulte útil; por ejemplo, por

finalidad, propietario o entorno. Para cada etiqueta que desee añadir, elija Añadir etiqueta nueva.

3. Seleccione Crear.

AWS CLI

Cree una ubicación personalizada mediante el comando [create-location](#). Proporcione un valor de `location-name`, que deberá empezar por `custom-`. Una práctica recomendada consiste en utilizar un nombre que describa una ubicación significativa para un conjunto de recursos de computación. Pueden ser ubicaciones geográficas, el nombre de un centro de datos u otro identificador de ubicación.

```
aws gamelift create-location \
  --location-name custom-location-1
```

Output

```
{
  "Location": {
    "LocationName": "custom-location-1",
    "LocationArn": "arn:aws:gamelift:us-east-1:111122223333:location/custom-
location-1"
  }
}
```

Creación de una flota de Anywhere

Cree una flota de Anywhere para un conjunto de recursos de computación de su propiedad. Una nueva flota de Anywhere comienza en blanco; para añadir recursos de computación, debe registrarlos.

Al crear una nueva flota de Anywhere, esta cambia rápidamente de estado de flota, de NEW a ACTIVE. Puede añadir recursos de computación a la flota tras llegar a ACTIVE.

Creación de una flota de Anywhere

Utilice la consola de Amazon GameLift Servers o la AWS Command Line Interface (AWS CLI) para crear una flota de Anywhere.

Console

En la [consola de Amazon GameLift Servers](#), use el panel de navegación para abrir la página Flotas. Seleccione Crear flota para iniciar el flujo de trabajo de creación de flotas.

Paso 1: elegir el tipo de recurso de computación

Seleccione la opción Anywhere y elija Siguiente.

Paso 2: definir los detalles de la flota

En este paso, especifique algunos ajustes importantes para toda la flota.

1. Rellene la sección Detalles de la flota:
 - a. Escriba un Nombre para la flota. Es recomendable utilizar un patrón de nomenclatura de flotas, a fin de identificar fácilmente los tipos de flota al ver las listas de flotas.
 - b. Introduzca una breve Descripción de la flota.
2. Configure estos Detalles adicionales opcionales según sea necesario. Puede actualizar estos ajustes de flota más tarde.
 - a. Al crear una flota para pruebas de producción o preproducción, use esta configuración para especificar un valor de Costo por hora para los recursos de computación de la flota. Amazon GameLift Servers puede usar esta información durante el proceso de ubicación de las sesiones de juego para seleccionar los recursos de alojamiento en función del costo.
 - b. Si desea combinar los datos de las métricas de esta flota y de otras, especifique un nombre de Grupo de métricas. Utilice el mismo nombre de grupo de métricas para todas las flotas que desee combinar. Consulte las métricas del grupo de métricas para ver los datos agregados.
3. Añada etiquetas a la ubicación personalizada. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Asigne etiquetas a los recursos de AWS que desee categorizar y hágalo de un modo que resulte útil; por ejemplo, por finalidad, propietario o entorno. Para cada etiqueta que desee añadir, elija Añadir etiqueta nueva.
4. Seleccione Siguiente para continuar con el flujo de trabajo.

Paso 3: seleccionar ubicaciones personalizadas

En este paso, identifique la ubicación física de los recursos de computación que planea añadir a la flota. Puede especificar una o más ubicaciones ahora y añadirlas o eliminarlas más tarde, según sea necesario.

1. En Ubicaciones personalizadas, seleccione una o más ubicaciones para los recursos de computación de la flota. La lista incluye todas las ubicaciones personalizadas que se han definido en la Región de AWS seleccionada actualmente. Para definir una nueva ubicación personalizada que desee añadir a la flota, elija Crear ubicación.
2. Seleccione Siguiente para continuar con el flujo de trabajo.

Paso 4: revisar y crear

Antes de crear la flota, revise la configuración.

Cuando esté listo para implementar la nueva flota, elija Crear. Amazon GameLift Servers comienza inmediatamente el proceso de activación de la flota, para lo cual asigna un ID único y pone la flota en estado NEW. Puede hacer un seguimiento del progreso de la flota en la página Flotas.

AWS CLI

Utilice el comando `create-fleet` para crear una flota del tipo de recurso de computación ANYWHERE. Proporcione un nombre y al menos una ubicación personalizada. Amazon GameLift Servers crea el recurso de flota de Anywhere en su Región de AWS predeterminada actual (también puede añadir una etiqueta `--region` para especificar una Región de AWS distinta).

El siguiente ejemplo de solicitud crea una nueva flota con la configuración mínima obligatoria. Sustituya *FleetName* y *custom-location* con su propia información.

```
aws gamelift create-fleet \  
--name FleetName \  
--compute-type ANYWHERE \  
--locations "Location=custom-location"
```

Respuesta de ejemplo

```
{
```

```
"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"
}
```

Al crear una nueva flota de Anywhere, esta pasa rápidamente al estado de flota ACTIVE. Puede añadir recursos de computación a la flota tras llegar a ACTIVE.

Tenga en cuenta que la respuesta no incluye las ubicaciones de la flota. Puede obtener todos los detalles de la flota llamando a [describe-fleet-attributes](#) y [describe-fleet-location-attributes](#).

Añadir un recurso de computación a la flota

Para añadir un recurso de computación a una flota y prepararla para alojar sesiones de juego, lleve a cabo las siguientes tareas:

- Registre el recurso de computación en la flota. El registro le dice a Amazon GameLift Servers cuáles son los recursos de alojamiento físicos que forman parte de la flota.
- Solicite un token de autenticación para el recurso de computación. Todos los servidores de juegos que se ejecutan en el recurso de computación necesitan este token para conectarse al servicio Amazon GameLift Servers. Los tokens de autenticación son temporales y deben actualizarse periódicamente.

Note

Si va a implementar el software de servidor de juegos con el agente de Amazon GameLift Servers, puede omitir este paso. El agente registra automáticamente cada recurso de

computación y mantiene un token de autenticación válido para el recurso. Consulte [Uso el agente de Amazon GameLift Servers](#).

Puede registrar un recurso de computación y solicitar un token de autenticación mediante la CLI de AWS o haciendo llamadas programáticas al SDK de AWS para Amazon GameLift Servers. Estas acciones no están disponibles en la consola de Amazon GameLift Servers.

La práctica recomendada consiste en automatizar estas dos tareas añadiendo un script de arranque en cada recurso de computación. El script de arranque llama automáticamente a los comandos `register-compute` y `get-compute-auth-token`. También puede automatizar las tareas para actualizar periódicamente el token de autenticación durante toda la vida útil del recurso de computación y anular el registro de este al apagarlo.

Todas las acciones de arranque devuelven valores específicos del recurso de computación que deberá almacenar en este. Cuando un proceso de servidor de juegos se inicia en el recurso de computación, debe pasar estos valores como parámetros del servidor al inicializar una conexión con el servicio Amazon GameLift Servers (consulte [ServerParameters](#) en la referencia del SDK del servidor). Le recomendamos que establezca estos valores específicos del recurso de computación (o sus ubicaciones almacenadas) como variables de entorno. Si utiliza el agente de Amazon GameLift Servers, esta tarea se realizará de forma automática. Los valores específicos del recurso de computación son los siguientes:

- `register-compute` devuelve un valor para `GameLiftServiceSdkEndpoint`. Establezca este valor en el parámetro del servidor `WebSocketUrl`.
- `compute-auth-token` devuelve el token de autenticación. Establezca este valor en el parámetro del servidor `authToken`.

AWS CLI

Las siguientes instrucciones describen cómo enviar manualmente cada solicitud mediante la CLI de AWS.

Registro de un recurso de computación

Llame a [register-compute](#) para registrar un recurso de computación. Determine el identificador de la flota a la que desea añadir el recurso de computación. Proporcione la siguiente información sobre el recurso de computación: un nombre significativo, una dirección IP y una ubicación. La ubicación del recurso de computación debe ser una ubicación personalizada que ya

esté asociada a la flota. Si desea utilizar una ubicación personalizada distinta, utilice la consola de Amazon GameLift Servers para actualizar la flota o llame al comando [create-fleet-locations](#) de la CLI de AWS para añadir una ubicación personalizada a la flota.

En el siguiente ejemplo, reemplace los valores de marcador de posición para su flota y recurso de computación. El valor `fleet-id` se devuelve al crear una flota de Anywhere. Puede obtener todos los detalles de la flota llamando a [describe-fleet-attributes](#) y [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
```

Ejemplo de resultado

```
{  
  "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"  
  }  
}
```

Solicitud de un token de autenticación

Llame a [get-compute-auth-token](#) para solicitar un token de autenticación válido. Registre un recurso de computación. Determine el identificador de la flota y el nombre del recurso de computación.

En el siguiente ejemplo, reemplace los valores de marcador de posición para su flota y recurso de computación. El valor `fleet-id` se devuelve al crear una flota de Anywhere. Puede obtener

todos los detalles de la flota llamando a [describe-fleet-attributes](#). Para obtener más información sobre el recurso de computación, llame a [list-compute](#) con el identificador de la flota para ver todos los recursos de computación registrados en esta.

```
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
```

Ejemplo de resultado

```
{  
  "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"  
}
```

Iniciar un servidor de juegos

Tras crear una flota de Anywhere y añadirle uno o más recursos de computación, ya puede empezar a ejecutar los servidores de juegos.

Paso 1: instalar el software del servidor de juegos

Instale la compilación del servidor de juegos y todo el software dependiente en todos los recursos de computación de la flota de Anywhere. La compilación del servidor de juegos debe estar integrada en la versión 5.x (o superior) del SDK del servidor de Amazon GameLift Servers, y debe tener la funcionalidad mínima requerida para comunicarse con el servicio de Amazon GameLift Servers.

Paso 2: preparar los recursos de computación para ejecutar un servidor de juegos

Compruebe que todos los recursos de computación estén registrados y tengan un token de autenticación válido. Si utiliza scripts para administrar estas tareas, compruebe que estos se ejecutan en cada recurso de computación antes de iniciar cualquier proceso del servidor de juegos.

Si ha implementado el agente de Amazon GameLift Servers con el software de su servidor de juegos, compruebe que se inicie el ejecutable del agente.

Paso 3: iniciar un proceso de servidor de juegos

Ejecute una instancia del ejecutable del servidor de juegos en un recurso de computación. Si la compilación del servidor de juegos está integrada correctamente, el proceso del servidor de juegos llamará a la acción `InitSDK()` del SDK del servidor con un conjunto de parámetros de servidor válidos. Cuando el proceso esté listo para alojar una sesión de juego, llamará a `ProcessReady()`.

Note

Si va a implementar el software del servidor de juegos con el agente de Amazon GameLift Servers, puede omitir este paso. El agente inicia automáticamente los procesos del servidor de juegos en función de las instrucciones de tiempo de ejecución que usted proporcione.

Puede supervisar el progreso consultando las métricas de los procesos del servidor para los procesos de servidor activos y con la activación en curso. Consulte [Métricas de flotas de Amazon GameLift Servers](#). Si el proceso del servidor de juegos no se inicializa, compruebe si el proceso está recuperando los valores de parámetros de servidor correctos para el recurso de computación en el que se están ejecutando.

Creación de una solución de alojamiento híbrida

Una solución de alojamiento híbrida combina varias fuentes de recursos de alojamiento de juegos para alojar tu juego, incluidas las flotas Amazon GameLift Servers gestionadas que se ejecutan en las instalaciones Nube de AWS y los recursos que tú suministres y gesticiones por tu cuenta. En este tema se describen algunos patrones comunes para crear una solución híbrida y se proporcionan consejos sobre cómo combinar correctamente el alojamiento de juegos autoadministrado con el alojamiento de juegos en la nube administrado por Amazon GameLift Servers.

Patrones híbridos comunes

- Optimización de costos: utilice el tipo de flota más rentable para cada escenario; por ejemplo, flotas de Anywhere para obtener capacidad inicial y flotas administradas para satisfacer los picos de demanda.

- Flexibilidad geográfica: despliegue flotas gestionadas en flotas de alto tráfico Regiones de AWS y en cualquier lugar donde disponga de una infraestructura existente o de requisitos de conformidad específicos.
- Mitigación de riesgos: reduzca la dependencia en todo enfoque de alojamiento único distribuyendo la carga entre varios tipos de flotas y proveedores.
- Migración gradual: pase de las soluciones de alojamiento locales o de otro tipo a las soluciones de alojamiento de AWS forma gradual y, al mismo tiempo, mantenga la continuidad del servicio.

Separación del desarrollo y la producción

Utilice diferentes tipos de flotas para los entornos de desarrollo y los de producción.

- Desarrollo: flotas de Anywhere para entornos de desarrollo y pruebas rentables
- Producción: Amazon gestionó flotas de contenedores EC2 o Amazon para un alojamiento de producción escalable y confiable

Optimización regional

Optimice los tipos de flota en función de las Regiones de AWS características:

- Alto tráfico Regiones de AWS: flotas gestionadas con autoscaling para una demanda variable
- Especializado Regiones de AWS: flotas en cualquier lugar para garantizar el cumplimiento, la soberanía de los datos o la infraestructura existente

Organización de la capacidad en niveles

Utilice diferentes tipos de flota para distintos niveles de capacidad:

- Capacidad inicial: flotas de Anywhere o instancias reservadas para carga predecible
- Capacidad de ampliación: flotas administradas con escalado automático para los picos de demanda
- Capacidad de desbordamiento: instancias puntuales o adicionales Regiones de AWS para picos extremos

Consideraciones de implementación

Al crear una solución híbrida, tenga en cuenta estos factores clave:

Configuración de las colas de sesiones de juego

Configure las colas de sesiones de juego para incluir todos los tipos de flota y establezca prioridades y preferencias de latencia adecuadas para garantizar una ubicación óptima en su infraestructura híbrida.

Supervisión y observabilidad

Implemente una supervisión integral en todos los tipos de flotas para mantener la visibilidad del rendimiento, la capacidad y los costos de su solución híbrida.

Complejidad operativa

Tenga en cuenta el aumento de la complejidad operativa que supone administrar varios tipos de flotas, incluidos los diferentes procesos de implementación, herramientas de supervisión y procedimientos de solución de problemas.

Conectividad de red

Asegure una conectividad de red fiable entre los distintos entornos de alojamiento, especialmente en el caso de las flotas de Anywhere que puedan residir en las instalaciones o en distintos proveedores de servicios en la nube.

Introducción al alojamiento híbrido

Para implementar una solución de alojamiento híbrida:

1. Comience de forma sencilla: comience con un solo tipo de flota y añada otros gradualmente a medida que los requisitos se vayan haciendo más evidentes.
2. Planifique la arquitectura: diseñe la arquitectura híbrida en función de sus requisitos específicos de costo, rendimiento, conformidad y complejidad operativa.
3. Configure las colas: configure colas de sesiones de juego que acomoden los distintos tipos de flota con las prioridades y estrategias de ubicación adecuadas.
4. Realice pruebas exhaustivas: pruebe la ubicación de las sesiones de juego y los escenarios de conmutación por error en toda la infraestructura híbrida antes de pasar a producción.
5. Supervise y optimice: supervise de forma continua el rendimiento y los costos de todos los tipos de flota y ajuste la configuración según sea necesario.

Consejos

- Utilice los mismos componentes de cliente y servidor de juegos, tanto con recursos de alojamiento administrados como autoadministrados. Proporcione una experiencia de jugador unificada en todos los recursos de alojamiento.
- Use los mismos emparejadores de FlexMatch para ubicar los emparejamientos en todos los recursos de alojamiento.
- Administre de forma centralizada los recursos de alojamiento híbrido de forma conjunta mientras los implementa por todo el mundo.
- A medida que la demanda de jugadores fluctúa, administre fácilmente las cargas de las sesiones de juego entre los recursos administrados y autoadministrados.
- Con el agente de Amazon GameLift Servers, puede usar las mismas herramientas para administrar los ciclos de vida de los servidores de juego en todos los tipos de recursos de alojamiento.
- Recopile métricas y registros de juegos y jugadores en todos los recursos de alojamiento. Aproveche las Amazon GameLift Servers funciones y otros AWS servicios para combinar datos y desarrollar soluciones de observabilidad cohesivas.

Configuración de la ubicación de las sesiones de juego

La ubicación de sesiones de juego es el proceso de encontrar servidores de juegos disponibles para alojar nuevas sesiones de juego. Amazon GameLift Servers utiliza las colas de sesiones de juego para ubicar las sesiones de juego de forma inteligente en sus flotas, teniendo en cuenta factores como la latencia de los jugadores, la capacidad de la flota y la optimización de los costos.

Una cola de sesiones de juego bien configurada garantiza que los jugadores se conecten al mejor servidor de juegos disponible, lo que proporciona un rendimiento óptimo y, al mismo tiempo, hace un uso eficiente de los recursos de alojamiento. Las colas pueden abarcar varias flotas y AWS regiones para ofrecer cobertura global y redundancia.

Una cola de sesiones de juego es el mecanismo principal que utiliza Amazon GameLift Servers para buscar servidores de juegos disponibles y elegirlos para alojar las nuevas sesiones de juego. Las colas ofrecen una forma mucho más eficiente de procesar un gran número de solicitudes de sesiones de juego y encontrar ubicaciones para ellas en varias flotas de recursos de alojamiento. Si la solución de alojamiento utiliza más de una flota y está procesando un gran volumen de solicitudes, es probable que necesite una cola.

Cuando el juego quiere iniciar una nueva sesión de juego para los jugadores, envía una solicitud de ubicación al servicio de Amazon GameLift Servers, que la canaliza a la cola. La configuración de la

cola determina cuándo y cómo se procesan las solicitudes. Al procesar una solicitud de ubicación, Amazon GameLift Servers busca en un conjunto de flotas un servidor de juegos para alojar la sesión de juego. La ubicación se realiza correctamente cuando Amazon GameLift Servers encuentra un servidor de juegos disponible y le pide que inicie una sesión de juego.

Temas

- [Características de una cola](#)
- [Prácticas recomendadas para las colas de sesiones de juego de Amazon GameLift Servers](#)
- [Creación de una cola de sesión de juego](#)
- [Configuración de la notificación de eventos para la ubicación de sesiones de juego.](#)

Características de una cola

Una cola de sesiones de Amazon GameLift Servers juego es un recurso en la nube. AWS Puede crear una cola en cualquier Región de AWS admitida por Amazon GameLift Servers (consulte [Ubicaciones de servicio de Amazon GameLift Servers](#)). Las solicitudes de ubicación de sesiones de juego se envían a esa ubicación, donde se procesan.

Automatizar la ubicación de las sesiones de juego ofrece importantes ventajas tanto para los desarrolladores de juegos como para los jugadores. Entre ellos se incluyen:

- Ofrecen la mejor ubicación posible. Al procesar las solicitudes de ubicación de las sesiones de juego, una cola utiliza el algoritmo de Amazon GameLift Servers FleetIQ para priorizar las ubicaciones en función de un conjunto de preferencias definidas, como el costo, la ubicación y la latencia de los jugadores.
- Admiten flotas de spot para contribuir a reducir los costos del alojamiento de juegos. Puedes configurar tus colas con las flotas de AWS Spot, que suelen ofrecer unos costes de alojamiento considerablemente más bajos, así como con las flotas bajo demanda. Dado que el bajo costo es uno de los criterios clave para las ubicaciones, las colas siempre pueden aprovechar las diferencias de costo.
- Pueden ubicar las nuevas partidas más rápido cuando hay una alta demanda. Al configurar una cola con múltiples flotas, se ofrecen opciones más flexibles para ubicar las sesiones de juego. Sin embargo, las flotas adicionales también proporcionan la capacidad de respaldo necesaria cuando aumenta la demanda. Para cualquier solicitud de ubicación, cuando Amazon GameLift Servers no pueda ubicar una sesión de juego en la ubicación preferida, pasará automáticamente a evaluar otras ubicaciones.

- Aumentan la resiliencia de la disponibilidad del servidor de juegos. Se pueden producir cortes de energía. Cuando se utiliza una cola de múltiples flotas, las ralentizaciones o interrupciones no tienen por qué afectar al acceso de los jugadores al juego. Al configurar tu cola con flotas con capacidad en diferentes zonas Regiones de AWS y zonas de disponibilidad, te asegurarás de que los jugadores siempre encuentren una sesión de juego a la que unirse.
- Obtención de métricas de las ubicaciones de sesiones de juego y del rendimiento de las colas. Amazon GameLift Servers emite métricas específicas de las colas, que incluyen estadísticas sobre éxitos y errores de ubicación, el número de solicitudes en la cola y el tiempo medio que las solicitudes pasan en la cola. Puede ver estas métricas en la consola de Amazon GameLift Servers o en CloudWatch.

Para empezar a crear una cola de inicio básica, consulte [Creación de una cola de sesión de juego](#).

Temas

- [Prácticas recomendadas para las colas de sesiones de juego de Amazon GameLift Servers](#)
- [Creación de una cola de sesión de juego](#)
- [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).

Prácticas recomendadas para las colas de sesiones de juego de Amazon GameLift Servers

Una cola de sesiones de juego contiene una lista de flotas en las que Amazon GameLift Servers puede ubicar nuevas sesiones de juego. Cada flota puede tener recursos de alojamiento implementados en varias ubicaciones geográficas. Al elegir una ubicación, la cola selecciona una flota y una ubicación de flota, de acuerdo con un conjunto de prioridades que haya establecido para la flota.

Tenga en cuenta las siguientes directrices y prácticas recomendadas:

- Añada flotas en ubicaciones que incluyan a sus jugadores. Puede añadir flotas y alias en cualquier ubicación disponible. La ubicación es importante si realiza ubicaciones en función de la latencia informada de los jugadores.
- Utilice alias para todas las flotas. Asigne un alias a cada flota en una cola y utilice los nombres de los alias al configurar los destinos en la cola.

- Utilice las mismas compilaciones del juego o scripts, o similares, para todas las flotas. La cola puede incluir a los jugadores en sesiones de juego de cualquier flota en la cola. Los jugadores deben poder jugar en cualquier sesión de juego de cualquier flota.
- Cree flotas en al menos dos ubicaciones. Al tener los servidores de juegos alojados en al menos otra ubicación, se reducirá el impacto de las interrupciones regionales en sus jugadores. Puede mantener sus flotas de respaldo con una reducción vertical y usar el escalado automático para aumentar la capacidad si el uso aumenta.
- Priorice la ubicación de las sesiones de juego. Una cola prioriza las opciones de ubicación en función de varios elementos, incluido el orden de la lista de destinos.
- Cree la cola en la misma ubicación que el servicio de cliente. Al colocar la cola en una ubicación cercana al servicio de cliente, puede minimizar la latencia de la comunicación.
- Utilice flotas con múltiples ubicaciones. Utilice la configuración del filtro de colas para evitar que la cola coloque las sesiones de juego en ubicaciones específicas. Puede utilizar al menos dos flotas con múltiples ubicaciones con diferentes ubicaciones de origen para reducir el impacto de las ubicaciones de los juegos durante una interrupción regional.
- Utilice la misma configuración de certificado TLS para todas las flotas. Los clientes de juego que se conectan a las sesiones de juego de sus flotas deben tener protocolos de comunicación compatibles.

Creación de una cola de sesión de juego

Las colas se utilizan para ubicar nuevas sesiones de juego en varias flotas y ubicaciones. El juego inicia nuevas sesiones de juego enviando las solicitudes de ubicación a una cola. Se configura una cola con instrucciones sobre cómo procesar las solicitudes. Para obtener más información sobre la ubicación de las sesiones de juego, consulte [Creación de sesiones de juego](#).

Creación de una cola de sesiones de juego

Estas instrucciones ilustran cómo crear una cola de trabajo sencilla con ajustes de configuración mínimos y ajustes predeterminados. Existen varias opciones para personalizar la configuración de una cola. Estas opciones le ayudan a obtener las mejores ubicaciones posibles en función de las necesidades del juego. Para obtener más información sobre cómo personalizar las colas para el juego, consulte [Personalización de colas de sesiones de juego](#). Puede actualizar la mayoría de los ajustes de configuración en cualquier momento.

Puede crear una cola de sesiones de juego mediante la consola de Amazon GameLift Servers o la CLI de AWS .

Console

En la [Amazon GameLift Servers consola](#), selecciona una AWS región en la que trabajar. Abra la barra de navegación izquierda de la consola y elija Colas.

1. En la página Colas, elija Crear cola para iniciar el flujo de trabajo.
2. En Ajustes de cola, introduzca lo siguiente:
 - a. Introduzca un nombre de cola. Este nombre debe ser exclusivo de la zona en la Región de AWS que vas a crear la cola.
 - b. Mantenga el ajuste predeterminado de Tiempo de espera, que es de 600 segundos (o 10 minutos). Este valor controla durante cuánto tiempo intenta Amazon GameLift Servers iniciar una nueva sesión de juego antes de detenerla. Amazon GameLift Servers busca recursos disponibles hasta que se agota el tiempo de espera de la solicitud. Puede actualizar el ajuste de tiempo de espera de una cola en cualquier momento.
 - c. Omita la sección Políticas de latencia de jugadores. Una cola usa políticas de latencia solo cuando recibe solicitudes de ubicación que incluyen datos de latencia de los jugadores. Puede añadir políticas de latencia a una cola en cualquier momento. Para obtener más información sobre cómo crear políticas de latencia, consulte [Creación de una política de latencia de jugadores](#).
3. Omita la sección Ubicaciones de las sesiones de juego para usar el ajuste predeterminado de Todas las ubicaciones. Este ajuste le permite crear una lista de ubicaciones permitidas en las que la cola puede realizar ubicaciones (esto también se denomina “configuración de filtro”). Para obtener más información sobre la priorización por ubicación y las configuraciones de filtro, consulte [Prioriza las ubicaciones por ubicación](#).
4. En Orden de destino, añada una o varias flotas a la cola. Puede identificar las flotas mediante una flota IDs o ARNs un alias de flota. Si va a añadir varias flotas, tenga en cuenta que todas deberían estar ejecutando compilaciones del juego similares y ser compatibles con cualquier cliente de juego que utilice esta cola. Además, todas las flotas de la cola deben tener la misma configuración de certificado.
 - a. Seleccione la región en la que se creó la flota o el alias. En el caso de las flotas con múltiples ubicaciones, esta es la región de “inicio”.
 - b. En el tipo de destino, seleccione una flota o un alias.
 - c. Las selecciones de región y tipo se utilizan para rellenar una lista desplegable de flotas o alias existentes. Seleccione un elemento de la lista para designarlo como destino de la cola.

- d. Para especificar otra flota o alias para la cola, elija Añadir destino y repita los pasos anteriores.
 - e. Una vez que hayas agregado una lista de destinos, usa la drag-and-drop función para reordenar los destinos. Amazon GameLift Servers usa este orden al priorizar las ubicaciones por destino.
5. Omita la sección Prioridad de ubicación de las sesiones de juego para mantener el orden de prioridad predeterminado. Este ajuste le permite personalizar la forma en la que Amazon GameLift Servers elige dónde buscar recursos de alojamiento disponibles para nuevas ubicaciones de las sesiones de juego. Para obtener más información sobre la priorización de ubicaciones, consulte [Priorización de la ubicación de las sesiones de juego](#). Puede actualizar las prioridades de ubicación de una cola en cualquier momento.
 6. En Orden de ubicación, conserve los valores predeterminados. Este ajuste permite priorizar por ubicación de la flota. Indica el orden de ubicación que se va a utilizar. Cuando se utiliza la configuración de prioridad predeterminada, la ubicación se utiliza como factor de desempate cuando el destino preferido es una flota con múltiples ubicaciones.
 7. Omita la sección opcional Configuración de notificaciones de eventos. Las notificaciones de eventos son necesarias para las colas que procesan un gran volumen de solicitudes de ubicación. En el caso de las colas que procesan volúmenes reducidos, por ejemplo, con fines de desarrollo o pruebas, puede realizar un seguimiento del estado de las solicitudes de colocación mediante una encuesta. [DescribeGameSessionPlacement](#) Para obtener más información, consulte [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#). Puede actualizar la configuración de notificaciones de eventos de una cola en cualquier momento.
 8. Elija Crear para generar una nueva cola con una personalización mínima.

AWS CLI

Example Creación de una cola

En el ejemplo siguiente se crea una cola de sesión de juego con estas configuraciones:

- Un tiempo de espera de cinco minutos.
- Dos destinos de flota.
- Filtre para permitir solo las siguientes ubicaciones: us-east-1, us-east-2, us-west-2 y ca-central-1.

- El orden de prioridad se basa en el costo y, después, en las ubicaciones en un orden específico.

```
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

Puede obtener los valores del ARN de la flota y del alias llamando a uno de los dos [describe-fleet-attributes](#) o [describiendo el alias con el identificador de flota o alias](#).

Si la `create-game-session-queue` solicitud se realiza correctamente, Amazon GameLift Servers devuelve un [GameSessionQueue](#) objeto con la nueva configuración de cola. Ahora puede enviar solicitudes a la cola utilizando. [StartGameSessionPlacement](#)

Example Creación de una cola con las políticas de latencia de los jugadores

En el ejemplo siguiente se crea una cola de sesión de juego con estas configuraciones:

- Un tiempo de espera de diez minutos
- Tres destinos de flota
- Un conjunto de políticas de latencia de los jugadores

```
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` solicitud se realiza correctamente, Amazon GameLift Servers devuelve un [GameSessionQueue](#) objeto con la nueva configuración de cola.

Configuración de la notificación de eventos para la ubicación de sesiones de juego.

Puede utilizar notificaciones de eventos para supervisar el estado de las solicitudes de ubicación individuales. Le recomendamos configurar las notificaciones de eventos para todos los juegos con un gran volumen de actividad de ubicación.

Existen dos opciones para configurar las notificaciones de eventos.

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

Para obtener una lista de los eventos de ubicación de sesiones de juego emitidos por Amazon GameLift Servers, consulte [Eventos de ubicación de sesión de juego](#).

Important

Para los sistemas de ubicación de gran volumen, recomendamos utilizar temas estándar (no FIFO) de Amazon SNS en lugar de temas FIFO. Los temas FIFO tienen límites de publicación más bajos que los temas estándar, lo que puede provocar excepciones de limitación en periodos de mucha carga. Si experimenta limitación con los temas FIFO, es posible que pierdas las notificaciones de ubicación en colas.

Configuración de un tema de SNS

Para que Amazon GameLift Servers publique todos los eventos generados por una cola de sesiones de juego en un tema, defina el campo de destino de notificaciones en él.

Para configurar un tema de SNS para la notificación de eventos de Amazon GameLift Servers

1. [Inicie sesión en la consola Amazon SNS en la v3/home Consola de administración de AWS y ábrala. https://console.aws.amazon.com/sns/](https://console.aws.amazon.com/sns/)
2. En la página Temas de SNS, elija Crear tema y siga las instrucciones para crear el tema.
3. En Política de acceso, haga lo siguiente:
 - a. Elija el método Avanzado.
 - b. Añada la siguiente sección en negrita del objeto JSON a la política existente.

JSON

```
{
  "Version": "2012-10-17",
  "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. Añada un control de acceso adicional al tema agregando condiciones a la política de recursos (opcional).
4. Seleccione Crear tema.
5. Una vez que haya creado su tema de SNS, añádalo a las colas durante la creación de las colas o edite una cola existente para agregarlo.

Configuración de un tema de SNS con cifrado del servidor

Con el cifrado del servidor (SSE), puede almacenar información confidencial en temas cifrados. SSE protege el contenido de los mensajes en temas de Amazon SNS mediante claves que se administran en AWS Key Management Service (AWS KMS). Para obtener más información sobre el cifrado del servidor con Amazon SNS, consulte [Cifrado en reposo](#) en la Guía para desarrolladores de Amazon Simple Notification Service.

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

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

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

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

Configure EventBridge

Amazon GameLift Servers publica automáticamente todos los eventos de ubicación de las sesiones de juego en EventBridge. Con EventBridge ella puedes configurar reglas para que los eventos se envíen a los objetivos para su procesamiento. Por ejemplo, puedes establecer una regla para dirigir el evento `PlacementFulfilled` a una AWS Lambda función que se encargue de las tareas previas a la conexión a una sesión de juego. Para obtener más información EventBridge, consulta [¿Qué es Amazon EventBridge?](#) en la Guía del EventBridge usuario de Amazon.

Los siguientes son algunos ejemplos de EventBridge reglas que se pueden usar con las Amazon GameLift Servers colas:

Empareja eventos de todas las colas de Amazon GameLift Servers

```
{
  "source": [
    "aws.gamelift"
  ],
```

```
"detail-type": [
    "GameLift Queue Placement Event"
]
}
```

Empareja eventos de una cola específica

```
{
  "source": [
    "aws.gamelift"
  ],
  "detail-type": [
    "GameLift Queue Placement Event"
  ],
  "resources": [
    "arn:aws:gamelift:your_region:your_account:gamesessionqueue/your_queue_name"
  ]
}
```

Haga referencia a la implementación con AWS CDK

Para obtener una implementación de referencia completa de la ubicación de sesiones de juego basada en eventos mediante Amazon SNS y Amazon DynamoDB AWS Lambda, consulte la guía de ubicación de [sesiones de juego basada en eventos en el](#) Amazon Toolkit. GameLift Esta guía incluye una AWS CDK plantilla para implementar un sistema completo de ubicación de sesiones de juego basado en eventos con una AWS Lambda función para procesar los eventos de ubicación de sesiones de juego, una tabla de Amazon DynamoDB para rastrear el estado de ubicación de las sesiones de juego y prácticas recomendadas para la configuración de colas. Amazon GameLift Servers

Personalización de su solución de alojamiento de juegos

Si tiene implantada una solución de alojamiento de juegos básica, consulte los siguientes temas para personalizarla y realzarla con el fin de mejorar la experiencia de los jugadores, optimizar los costos y añadir funciones avanzadas. En esta sección se describen varias opciones de personalización, organizadas por el componente al que afectan principalmente. Elija las personalizaciones que mejor se adapten a los requisitos del juego y a su base de jugadores.

Temas

- [Personalizaciones de compilaciones de servidores de juegos](#)
 - [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#)
 - [Habilitación del acceso del servidor de juegos a los datos de la flota de Amazon GameLift Servers](#)
 - [Configuración del emparejamiento de VPC para Amazon GameLift Servers](#)
- [Personalizaciones de sesiones de jugador y emparejamientos](#)
 - [Generación de ID de jugador](#)
 - [Incorporación del emparejamiento de FlexMatch a Amazon GameLift Servers](#)
- [Personalizaciones de ubicaciones de sesión de juego](#)
 - [Personalización de colas de sesiones de juego](#)
 - [Priorización de la ubicación de las sesiones de juego.](#)
 - [Creación de una cola para instancias de spot](#)
- [Personalizaciones de recursos de alojamiento](#)
 - [Elección de los recursos de computación para una flota administrada](#)
 - [Personalización de una flota de contenedores de Amazon GameLift Servers](#)
 - [Reducción de los costos de alojamiento de juegos con flotas de spot](#)
 - [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#)
 - [Uso el agente de Amazon GameLift Servers](#)
 - [Abstracción de una designación de flota de Amazon GameLift Servers con un alias](#)

Personalizaciones de compilaciones de servidores de juegos

Las personalizaciones de la versión del servidor de juegos ofrecen oportunidades para mejorar la funcionalidad del servidor de juegos, incluso para aprovechar otros AWS servicios. Estas personalizaciones amplían las capacidades del servidor de juegos más allá del alojamiento básico para admitir características e integraciones avanzadas.

Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos

Al crear una compilación de servidor de juegos para implementarla en las flotas de Amazon GameLift Servers, es posible que desee que las aplicaciones de la compilación de juegos se comuniquen de

forma directa y segura con otros recursos de AWS de su propiedad. Puesto que Amazon GameLift Servers administra sus flotas de alojamiento de juegos, debe conceder a Amazon GameLift Servers un acceso limitado a estos recursos y servicios.

Entre los ejemplos de escenarios se incluyen los siguientes:

- Utilice un CloudWatch agente de Amazon para recopilar métricas, registros y trazas de las flotas de EC2 gestionadas y de las flotas de Anywhere.
- Envía los datos del registro de instancias a Amazon CloudWatch Logs.
- Obtención de archivos de juegos almacenados en un bucket de Amazon Simple Storage Service (Amazon S3).
- Lectura y escritura de datos de juegos (como el inventario o los modos de juego) almacenados en una base de datos de Amazon DynamoDB u otro servicio de almacenamiento de datos.
- Envío de señales directamente a una instancia mediante Amazon Simple Queue Service (Amazon SQS).
- Acceso a recursos personalizados implementados y que se ejecutan en Amazon Elastic Compute Cloud (Amazon EC2).

Amazon GameLift Servers admite los siguientes métodos para establecer el acceso:

- [Acceda a AWS los recursos con un rol de IAM](#)
- [Acceda a AWS los recursos con la interconexión de VPC](#)

Acceda a AWS los recursos con un rol de IAM

Utilice un rol de IAM para especificar quién puede acceder a sus recursos y establecer límites a ese acceso. Las partes de confianza pueden “asumir” un rol y obtener credenciales de seguridad temporales que les autoricen a interactuar con los recursos. Cuando las partes realizan solicitudes de la API relacionadas con el recurso, deben incluir las credenciales.

Para configurar el acceso controlado por un rol de IAM, lleve a cabo las siguientes tareas:

1. [Creación del rol de IAM](#)
2. [Modificación de las aplicaciones para adquirir credenciales](#)
3. [Asociación del rol de IAM a una flota](#)

Creación del rol de IAM

En este paso, crea un rol de IAM, con un conjunto de permisos para controlar el acceso a sus AWS recursos y una política de confianza que otorga Amazon GameLift Servers derechos para usar los permisos del rol.

Para obtener instrucciones sobre cómo configurar el rol de IAM, consulte [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#). Al crear la política de permisos, elija servicios, recursos y acciones específicos con los que deben trabajar sus aplicaciones. Como práctica recomendada, limite el ámbito de los permisos tanto como sea posible.

Después de crear el rol, anote el nombre de recurso de Amazon (ARN) del rol. El ARN del rol será necesario durante la creación de la flota.

Modificación de las aplicaciones para adquirir credenciales

En este paso, configura las aplicaciones para que adquieran credenciales de seguridad para el rol de IAM y las usen al interactuar con sus AWS recursos. Consulte la siguiente tabla para determinar cómo modificar las aplicaciones en función de (1) el tipo de aplicación y (2) la versión del SDK del servidor que utilice el juego para comunicarse con Amazon GameLift Servers.

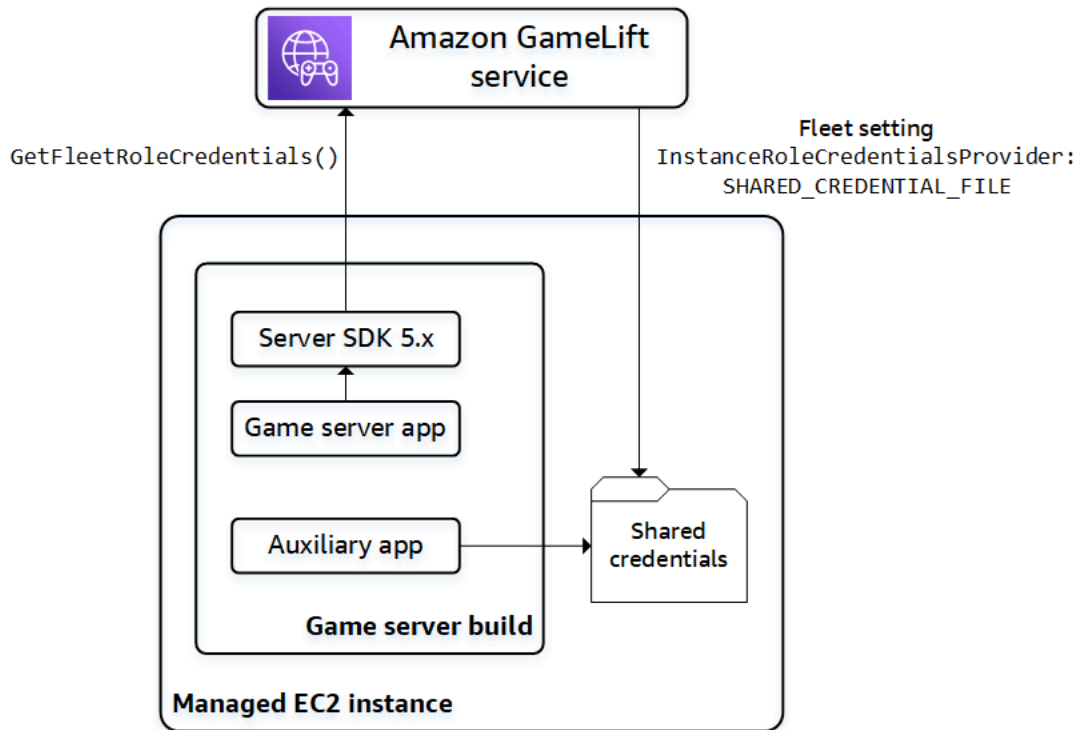
	Aplicaciones del servidor de juegos	Otras aplicaciones
Uso del SDK del servidor, versión 5.x	Llame al método del SDK del servidor <code>GetFleetRoleCredentials()</code> desde el código del servidor de juegos.	Añada código a la aplicación para obtener las credenciales de un archivo compartido en la instancia de la flota.
Uso de la versión 4 o anterior del SDK del servidor	Call AWS Security Token Service (AWS STS) AssumeRole con el rol ARN.	Call AWS Security Token Service (AWS STS) AssumeRole con el rol ARN.

Note

En el caso de las flotas de contenedores, `FleetRoleArn` las credenciales se insertan en cada contenedor. Sus aplicaciones pueden usar el proveedor de AWS credenciales predeterminado para acceder a estas credenciales. Puede seguir

llamando `GetFleetRoleCredentials()`, lo que devuelve las mismas credenciales. Solo se puede acceder a estas credenciales de rol de flota dentro de los contenedores.

En el caso de los juegos integrados con el SDK 5.x del servidor, este diagrama muestra cómo las aplicaciones de la compilación del juego implementada pueden adquirir credenciales para el rol de IAM.



Llamada a `GetFleetRoleCredentials()` (SDK del servidor 5.x)

En el código del servidor de juegos, que ya debería estar integrado con el SDK del servidor de Amazon GameLift Servers 5.x, llame a `GetFleetRoleCredentials` ([C++](#)) ([C#](#)) ([Unreal](#)) ([Go](#)) para recuperar un conjunto de credenciales temporales. Cuando las credenciales caduquen, puede actualizarlas con otra llamada a `GetFleetRoleCredentials`.

Utilización de credenciales compartidas (SDK del servidor 5.x)

En el caso de las aplicaciones que no son de servidor y que se implementan con compilaciones de servidores de juegos que utilizan el SDK del servidor 5.x, añada código para obtener y utilizar las credenciales almacenadas en un archivo compartido. Amazon GameLift Servers genera un perfil de credenciales para cada instancia de flota. Las credenciales están disponibles para que las

utilicen todas las aplicaciones de la instancia. Amazon GameLift Servers actualiza continuamente las credenciales temporales.

Debe configurar una flota para generar el archivo de credenciales compartidas al crear la flota.

En cada aplicación que necesite utilizar el archivo de credenciales compartido, especifique la ubicación del archivo y el nombre del perfil de la siguiente manera:

Windows:

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "C:\\Credentials\\credentials"
```

Linux:

```
[credentials]
shared_credential_profile= "FleetRoleCredentials"
shared_credential_file= "/local/credentials/credentials"
```

Ejemplo: configurar un CloudWatch agente para recopilar métricas para las instancias de Amazon GameLift Servers flota

Si quieres utilizar un CloudWatch agente de Amazon para recopilar métricas, registros y rastreos de tus Amazon GameLift Servers flotas, utiliza este método para autorizar al agente a emitir los datos a tu cuenta. En este escenario, realice los pasos siguientes:

1. Recupera o escribe el `config.json` archivo del CloudWatch agente.
2. Actualice el archivo `common-config.toml` del agente para identificar el nombre del archivo de credenciales y el nombre del perfil, tal y como se describió previamente.
3. Configure el script de instalación del servidor de juegos para instalar e iniciar el CloudWatch agente.

Utilización de **AssumeRole()** (SDK del servidor 4)

Añada código a sus aplicaciones para asumir la función de IAM y obtener credenciales para interactuar con sus AWS recursos. Cualquier aplicación que se ejecute en una instancia de flota de Amazon GameLift Servers con un SDK del servidor correspondiente a la versión 4, o a una versión anterior, puede asumir el rol de IAM.

En el código de la aplicación, antes de acceder a un AWS recurso, la aplicación debe llamar a la operación [AssumeRole](#) API AWS Security Token Service (AWS STS) y especificar el ARN del rol. Esta operación devuelve un conjunto de credenciales temporales que autorizan a la aplicación a acceder al AWS recurso. Para obtener más información, consulte [Uso de credenciales temporales con AWS recursos](#) en la Guía del usuario de IAM.

Asociación del rol de IAM a una flota

Una vez que haya creado el rol de IAM y actualizado las aplicaciones de la compilación del servidor de juegos para obtener y usar las credenciales de acceso, podrá implementar una flota. Al configurar la nueva flota, defina los siguientes parámetros.

Para flotas de contenedores:

- [FleetRoleArn](#)— Establezca este parámetro en el ARN del rol de IAM.

Para otros tipos de flota:

- [InstanceRoleArn](#)— Establezca este parámetro en el ARN del rol de IAM.
- [InstanceRoleCredentialsProvider](#)— Amazon GameLift Servers Para solicitar la generación de un archivo de credenciales compartido para cada instancia de flota, defina este parámetro en. `SHARED_CREDENTIAL_FILE`

Debe establecer esos valores al crear la flota. No se podrán actualizar más tarde.

Acceda a AWS los recursos con la interconexión de VPC

Puede utilizar la interconexión de Amazon Virtual Private Cloud (Amazon VPC) para comunicarse entre las aplicaciones que se ejecutan en una Amazon GameLift Servers instancia y otro recurso. AWS Una VPC es una red privada virtual que usted define e incluye un conjunto de recursos administrados a través de su. Cuenta de AWS Cada flota de Amazon GameLift Servers tiene su propia VPC. Con la interconexión de VPC, puede establecer una conexión de red directa entre la VPC de su flota y el resto de sus recursos. AWS

Amazon GameLift Servers simplifica el proceso de configuración de la interconexión de VPC para los servidores de juegos. Gestiona las solicitudes de interconexión, actualiza la tablas de ruteo y configura las conexiones según sea necesario. Para obtener instrucciones sobre cómo establecer el emparejamiento de VPC para los servidores de juegos, consulte [Configuración del emparejamiento de VPC para Amazon GameLift Servers](#).

 Note

Las flotas de contenedores no admiten el emparejamiento de VPC.

Habilitación del acceso del servidor de juegos a los datos de la flota de Amazon GameLift Servers

Existen determinadas situaciones en las que una compilación de juego personalizada o un script de Amazon GameLift Servers Realtime pueden requerir información sobre la flota de Amazon GameLift Servers. Por ejemplo, la compilación o el script del juego pueden incluir código para realizar las siguientes acciones:

- Supervisar la actividad en función de los datos de la flota.
- Reunir las métricas para realizar un seguimiento de la actividad según los datos de la flota. (Muchos juegos utilizan estos datos para LiveOps realizar actividades).
- Proporcionar datos relevantes a los servicios de juegos personalizados, por ejemplo, para el emparejamiento, la escalación de la capacidad adicional o la realización de pruebas.

La información de la flota se encuentra disponible como un archivo JSON en cada instancia en las siguientes ubicaciones:

- Windows: C:\GameMetadata\gamelift-metadata.json
- Linux: /local/gamemetadata/gamelift-metadata.json

El `gamelift-metadata.json` archivo incluye los [atributos de un recurso de Amazon GameLift Servers flota](#).

Ejemplo de archivo 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"  
}
```

Configuración del emparejamiento de VPC para Amazon GameLift Servers

En este tema, se ofrecen directrices sobre cómo configurar una interconexión de VPC entre sus servidores de juegos alojados en Amazon GameLift Servers y otros recursos que no pertenecen a Amazon GameLift Servers. Utilice las conexiones de emparejamiento de Amazon Virtual Private Cloud (VPC) para permitir que sus servidores de juegos se comuniquen de forma directa y privada con sus otros AWS recursos, como un servicio web o un repositorio. Puede establecer el emparejamiento de VPC con cualquier recurso que se ejecute en una AWS cuenta a la que tenga acceso AWS y que esté gestionado por ella.

Note

La interconexión de VPC es una característica avanzada. Para obtener más información sobre las opciones preferidas para permitir que tus servidores de juegos se comuniquen de forma directa y privada con tus otros AWS recursos, consulta [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#)

Si ya estás familiarizado con Amazon VPCs y el peering de VPC, entiende que configurar el peering con los servidores de Amazon GameLift Servers juegos es algo diferente. No dispone de acceso a la VPC que contiene los servidores de juegos (que se controla mediante el servicio de Amazon GameLift Servers), por lo que no puede solicitar directamente el emparejamiento de VPC. En lugar de ello, primero debe preautorizar la VPC con sus recursos que no pertenecen a Amazon GameLift Servers para aceptar una solicitud de interconexión del servicio Amazon GameLift Servers. A continuación, activa Amazon GameLift Servers para solicitar la interconexión de VPC que acaba de autorizar. Amazon GameLift Servers administra las tareas de creación de la interconexión, configuración de las tablas de enrutamiento y configuración de la conexión.

Para configurar la interconexión de VPC para una flota existente

1. Obtén los ID de AWS cuenta y las credenciales.

Necesitas un ID y credenciales de inicio de sesión para las siguientes AWS cuentas. Para encontrar la AWS cuenta, inicia sesión [Consola de administración de AWS](#) y consulta la configuración de la cuenta. Para obtener las credenciales, vaya a la consola de IAM.

- AWS cuenta que utilizas para gestionar tus servidores de Amazon GameLift Servers juegos.
- AWS cuenta que utilizas para gestionar tus recursos no relacionados con los Amazon GameLift Servers recursos.

Si utiliza la misma cuenta para recursos de Amazon GameLift Servers y otros que no sean de Amazon GameLift Servers, necesita el ID y las credenciales de esa cuenta solo.

2. Obtenga los identificadores de cada VPC.

Obtenga la siguiente información para VPCs comparar ambas opciones:

- VPC para sus servidores de juegos de Amazon GameLift Servers: es decir, el ID de flota de Amazon GameLift Servers. Los servidores de juegos están implementados en Amazon GameLift Servers en una flota de instancias EC2. Cada flota se sitúa automáticamente en su propia VPC, que se administra mediante el servicio Amazon GameLift Servers. No dispone de acceso directo a la VPC, por lo que se identifica mediante el ID de la flota.
- VPC para sus recursos no relacionados con Amazon GameLift Servers AWS los recursos: puede establecer un emparejamiento de VPC con cualquier recurso que se ejecute AWS y esté administrado por una AWS cuenta a la que tenga acceso. Si todavía no ha creado una VPC para esos recursos, consulte [Introducción a Amazon VPC](#). Una vez que haya creado una VPC, podrá encontrar el ID de la VPC iniciando sesión en la VPC de [Consola de administración de AWS](#) Amazon y viendo la suya. VPCs

Note

Al configurar una interconexión, ambas VPCs deben existir en la misma región. La VPC de los servidores de juegos de la flota de Amazon GameLift Servers está en la misma región que la flota.

3. Autorizar una interconexión de VPC.

En este paso, preautorizará una solicitud futura de Amazon GameLift Servers para conectar la VPC con los servidores de juegos con la VPC para recursos que no pertenecen a Amazon GameLift Servers. Esta acción actualiza el grupo de seguridad de la VPC.

Para autorizar el emparejamiento de VPC, llame a la API de servicio [CreateVpcPeeringAuthorization\(\)](#) o utilice el comando CLI AWS `. create-vpc-peering-authorization` Realice esta llamada mediante la cuenta que administra los recursos que no pertenecen a Amazon GameLift Servers. Proporcione la siguiente información:

- ID de VPC de par: es para la VPC con los recursos que no pertenecen a Amazon GameLift Servers.
- Amazon GameLift Servers AWS ID de cuenta: es la cuenta que utilizas para gestionar tu Amazon GameLift Servers flota.

Una vez que haya autorizado la interconexión de VPC, la autorización sigue siendo válida durante 24 horas, a menos que se revoque. Puede administrar las autorizaciones de interconexión de VPC mediante las siguientes operaciones:

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).

4. Solicite una interconexión.

Con una autorización válida, puede solicitar que Amazon GameLift Servers establezca una interconexión.

Para solicitar un emparejamiento de VPC, llama a la API de servicio [CreateVpcPeeringConnection\(\)](#) o usa el comando CLI AWS `. create-vpc-peering-connection` Realice esta llamada con la cuenta que administra los servidores de juego de Amazon GameLift Servers. Usa la siguiente información para identificar las dos VPCs que quieres emparejar:

- ID de VPC del mismo nivel e ID de AWS cuenta: esta es la VPC para sus recursos no relacionados con Amazon GameLift Servers los recursos y la cuenta que utiliza para administrarlos. El ID de la VPC debe coincidir con el ID de una autorización de interconexión válida.

- ID de la flota: identifica la VPC de los servidores de juegos de Amazon GameLift Servers.
5. Realice un seguimiento del estado de la interconexión.

La solicitud de una interconexión de VPC es una operación asíncrona. Para realizar el seguimiento del estado de una solicitud de interconexión y gestionar los casos de éxito o error, utilice una de las siguientes opciones:

- Sondee continuamente con `DescribeVpcPeeringConnections()`. Esta operación recupera el registro de la interconexión de VPC, incluido el estado de la solicitud. Si se crea correctamente una interconexión, el registro de conexión también contiene un bloque de CIDR de direcciones IP privadas que se asigna a la VPC.
- Gestione los eventos de la flota asociados a las conexiones de emparejamiento de VPC con [DescribeFleetEvents\(\)](#), incluidos los eventos de éxito y fracaso.

Una vez establecida la interconexión, puede administrarla mediante las siguientes operaciones:

- [DescribeVpcPeeringConnections\(\)](#) (AWS CLI `describe-vpc-peering-connections`).
- [DeleteVpcPeeringConnection\(\)](#) (AWS CLI `delete-vpc-peering-connection`).

Para configurar la interconexión de VPC con una nueva flota

Puede crear una nueva flota de Amazon GameLift Servers y solicitar una interconexión de VPC al mismo tiempo.

1. Obtenga los ID de AWS cuenta y las credenciales.

Necesitas un ID y credenciales de inicio de sesión para las dos AWS cuentas siguientes. Para encontrar la AWS cuenta, IDs inicia sesión [Consola de administración de AWS](#) y consulta la configuración de la cuenta. Para obtener las credenciales, vaya a la consola de IAM.

- AWS cuenta que utilizas para gestionar tus servidores de Amazon GameLift Servers juegos.
- AWS cuenta que utilizas para gestionar tus recursos no relacionados con los Amazon GameLift Servers recursos.

Si utiliza la misma cuenta para recursos de Amazon GameLift Servers y otros que no sean de Amazon GameLift Servers, necesita el ID y las credenciales de esa cuenta solo.

2. Obtenga el ID de VPC para sus recursos no relacionados con recursos. Amazon GameLift Servers AWS

Si todavía no ha creado una VPC para estos recursos, hágalo ahora (consulte [Introducción a Amazon VPC](#)). Asegúrese de crear la nueva VPC en la misma región en la que piensa crear su nueva flota. Si tus Amazon GameLift Servers recursos no relacionados se administran en una AWS cuenta o user/user grupo diferente al que utilizas Amazon GameLift Servers, tendrás que usar estas credenciales de cuenta cuando solicites la autorización en el siguiente paso.

Una vez que haya creado una VPC, podrá localizar el ID de la VPC en la consola de Amazon VPC consultando su. VPCs

3. Autorice una interconexión de VPC con los recursos que no pertenecen a Amazon GameLift Servers.

Cuando Amazon GameLift Servers crea la nueva flota y la VPC correspondiente, también envía una solicitud para interconectar la VPC para los recursos que no pertenecen a Amazon GameLift Servers. Debe para preautorizar dicha solicitud. Este paso actualiza el grupo de seguridad de la VPC.

Con las credenciales de la cuenta que administran sus recursos no relacionados con Amazon GameLift Servers los recursos, llame a la API de servicio [CreateVpcPeeringAuthorization\(\)](#) o utilice el comando AWS CLI `create-vpc-peering-authorization`. Proporcione la siguiente información:

- ID de VPC de par: ID de la VPC con los recursos que no pertenecen a Amazon GameLift Servers.
- Amazon GameLift Servers AWS ID de cuenta: ID de la cuenta que utilizas para gestionar tu Amazon GameLift Servers flota.

Una vez que haya autorizado la interconexión de VPC, la autorización sigue siendo válida durante 24 horas, a menos que se revoque. Puede administrar las autorizaciones de interconexión de VPC mediante las siguientes operaciones:

- [DescribeVpcPeeringAuthorizations\(\)](#) (AWS CLI `describe-vpc-peering-authorizations`).
- [DeleteVpcPeeringAuthorization\(\)](#) (AWS CLI `delete-vpc-peering-authorization`).

4. Siga las instrucciones para [crear una nueva flota mediante la AWS CLI](#). Incluya los siguientes parámetros adicionales:
 - `peer-vpc-aws-account-id`: ID de la cuenta que utiliza para administrar la VPC con sus recursos no Amazon GameLift Servers relacionados con los recursos.
 - `peer-vpc-id`— ID de la VPC con la que no es una cuenta. Amazon GameLift Servers

Una llamada correcta a [create-fleet](#) con los parámetros de la interconexión de VPC genera una flota nueva y una nueva solicitud de interconexión de VPC. El estado de la flota se establece en `New` y se inicia el proceso de activación de la flota. El estado de la solicitud de interconexión se establece en `initiating-request`. Puede realizar un seguimiento del éxito o el fracaso de la solicitud de intercambio de pares llamando. [describe-vpc-peering-connections](#)

Cuando se solicita una nueva flota y una interconexión de VPC, las dos acciones se realizarán correctamente o producirán un error. Si se produce un error en una flota durante el proceso de creación, la interconexión de VPC no se establecerá. Del mismo modo, si falla una interconexión de VPC por cualquier motivo, la flota nueva no podrá pasar del estado `Activating` al estado `Active`.

 Note

La nueva interconexión de VPC no se completará hasta que la flota esté lista para activarse. Esto significa que la conexión no está disponible y no se puede usar durante el proceso de instalación de la compilación del servidor de juegos.

En el siguiente ejemplo se crea una nueva flota y una interconexión entre una VPC previamente establecida y la VPC de la nueva flota. La VPC preestablecida se identifica de forma exclusiva mediante la combinación de su ID ajeno a la cuenta Amazon GameLift Servers AWS y el ID de 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"

```

Versión 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"

```

Solución de problemas de interconexión de VPC

Si tiene problemas para establecer una interconexión de VPC para sus servidores de juegos de Amazon GameLift Servers, tenga en cuenta estas causas principales comunes:

- No se ha encontrado una autorización para la conexión solicitada:
 - compruebe el estado de una autorización de VPC para la VPC no perteneciente a Amazon GameLift Servers. Es posible que no exista o que haya caducado.
 - Comprueba las regiones de las dos VPCs que estás intentando emparejar. Si no están en la misma región, no se pueden interconectar.
- Los bloques CIDR (consulte Configuraciones de [conexión de emparejamiento de VPC no válidas](#)) de sus VPCs dos bloques se superponen. Los bloques IPv4 CIDR que están asignados a los pares no se pueden superponer. VPCs El bloque de CIDR de la VPC de su flota de Amazon GameLift

Servers se asigna automáticamente y no se puede cambiar, por lo que tendrá que cambiar el bloque de CIDR de la VPC para los recursos no pertenecientes a Amazon GameLift Servers. Para resolver este problema, siga estos pasos:

- Busque este bloque de CIDR para su flota de Amazon GameLift Servers llamando a `DescribeVpcPeeringConnections()`.
- Vaya a la consola de Amazon VPC, busque la VPC para los recursos no pertenecientes a Amazon GameLift Servers y cambie el bloque de CIDR para que no se superpongan.
- La nueva flota no se activó (al solicitar la interconexión de VPC con una nueva flota). Si la nueva flota no pudo avanzar hasta el estado Activo no hay ninguna VPC con la que interconectar, por lo que no se puede realizar la interconexión.

Personalizaciones de sesiones de jugador y emparejamientos

Las personalizaciones de sesiones de jugador y emparejamientos le brindan la oportunidad de desarrollar sofisticados flujos de trabajo de administración de jugadores, incluido un sistema de emparejamiento detallado que le ayuda a ofrecer experiencias multijugador equilibradas y atractivas.

Generación de ID de jugador

Amazon GameLift Servers utiliza una sesión de jugador como representación de cada jugador conectado a una sesión de juego. Amazon GameLift Servers crea una sesión de jugador cada vez que un jugador se conecta a una sesión de juego utilizando un cliente de juego integrado con Amazon GameLift Servers. Cuando un jugador sale del juego, la sesión de jugador finaliza. Amazon GameLift Servers no reutiliza las sesiones de jugador.

Important

Cuando se está utilizando el emparejamiento de FlexMatch y se crea una nueva solicitud de emparejamiento que contiene un ID de jugador que ya está incluido en una solicitud de emparejamiento existente, esta última se cancela automáticamente. Sin embargo, no se envía ningún evento `MatchmakingCancelled` para la solicitud cancelada. Para supervisar el estado de las solicitudes de emparejamiento existentes, utilice [DescribeMatchmaking](#) para sondear el estado de las solicitudes a intervalos poco frecuentes (30-60 segundos). La solicitud cancelada mostrará el estado `CANCELLED` con el motivo `Cancelled due to duplicate player`.

El siguiente código de ejemplo genera ID de jugador únicos de forma aleatoria:

```
bool includeBrackets = false;
bool includeDashes = true;
string playerId = AZ::Uuid::CreateRandom().ToString<string>(includeBrackets,
    includeDashes);
```

Para obtener más información sobre las sesiones de jugador, consulte [Sesiones de juego y de jugador en la consola de Amazon GameLift Servers](#).

Incorporación del emparejamiento de FlexMatch a Amazon GameLift Servers

Utilice Amazon GameLift Servers FlexMatch para incorporar la funcionalidad de emparejamiento de jugadores a los juegos alojados en Amazon GameLift Servers. Puede utilizar FlexMatch con servidores de juegos personalizados o con Amazon GameLift Servers Realtime.

FlexMatch combina el servicio de emparejamiento con un motor de reglas personalizable. Diseñe la forma de emparejar a los jugadores en función de los atributos de los jugadores y los modos de juego que mejor se adapten a su juego. FlexMatch administra los aspectos prácticos de evaluar a los jugadores que buscan un juego, y crea emparejamientos con uno o más equipos e inicia sesiones de juego para alojar los emparejamientos.

Para utilizar el servicio de FlexMatch al completo, debe tener los recursos de alojamiento configurados con colas. Amazon GameLift Servers utiliza colas para localizar las mejores ubicaciones de alojamiento posibles para los juegos en diversas regiones y tipos de computación. En concreto, las colas de Amazon GameLift Servers pueden usar los datos de latencia, cuando los clientes de juego los proporcionen, para ubicar las sesiones de juego de forma que los jugadores experimenten la latencia más baja posible al jugar.

Para obtener más información sobre FlexMatch, incluida ayuda detallada para integrar el emparejamiento en los juegos, consulte estos temas de la [Guía para desarrolladores de Amazon GameLift Servers FlexMatch](#):

- [Cómo funciona Amazon GameLift Servers FlexMatch](#)
- [Pasos de integración de FlexMatch](#)

Personalizaciones de ubicaciones de sesión de juego

Las personalizaciones de ubicaciones de sesión de juego le brindan la oportunidad de refinar su sistema de ubicación con el fin de ofrecer a su base de jugadores una experiencia de juego óptima. Puede optimizar la ubicación en función de la compatibilidad y las preferencias de los jugadores, así como teniendo en cuenta aspectos operativos como la rentabilidad, la distribución geográfica y la resiliencia contra las interrupciones del servicio.

Personalización de colas de sesiones de juego

En este tema se describe cómo personalizar las colas de sesiones de juego para que se tomen decisiones óptimas en cuanto a la ubicación de las sesiones de juego. Para obtener más información sobre las colas de sesiones de juego y su funcionamiento, consulte [Configuración de la ubicación de las sesiones de juego](#).

Estas características de Amazon GameLift Servers requieren colas:

- [Emparejamientos con FlexMatch](#)
- [Creación de una cola para instancias de spot](#)

Temas

- [Definición del ámbito de una cola](#)
- [Creación una cola con múltiples ubicaciones](#)
- [Evaluación de métricas de cola](#)

Definición del ámbito de una cola

Es posible que la población de jugadores del juego tenga grupos de jugadores que no deberían jugar juntos. Por ejemplo, si publica el juego en dos idiomas, cada idioma debería tener sus propios servidores de juegos.

Para configurar la ubicación de las sesiones de juego para su población de jugadores, cree una cola independiente para cada segmento de jugadores. Revise cada cola para ubicar a los jugadores en los servidores de juegos correctos. Algunas formas habituales de determinar el ámbito de las colas son las siguientes:

- Por ubicaciones geográficas. Al implementar los servidores de juegos en varias áreas geográficas, puede crear colas para los jugadores en cada ubicación con el fin de reducir la latencia de los jugadores.
- Por compilación o variaciones del script. Si tiene más de una variante del servidor de juegos, es posible que esté ofreciendo soporte a grupos de jugadores que no puedan jugar en las mismas sesiones de juego. Por ejemplo, es posible que las compilaciones o scripts de los servidores de juegos admitan distintos idiomas o tipos de dispositivos.
- Por tipos de eventos. Puede crear una cola especial para administrar los juegos de los participantes en torneos u otros eventos especiales.

Diseño de varias colas

En función del juego y de los jugadores, es posible que desee crear más de una cola de sesión de juego. Cuando su servicio de cliente de juegos solicite una nueva sesión de juego, especifica qué cola de sesión de juego utilizar. Para ayudarle a determinar si debe usar varias colas, tenga en cuenta los siguientes aspectos:

- Variaciones del servidor de juegos. Puede crear una cola independiente para cada variación del servidor de juegos. Todas las flotas de una cola deben implementar servidores de juegos compatibles. Esto se debe a que los jugadores que utilicen la cola para unirse a los juegos deben poder jugar en cualquiera de los servidores de juegos de la cola.
- Diferentes grupos de jugadores. Puede personalizar la forma en que Amazon GameLift Servers ubica las sesiones de juego en función del grupo de jugadores. Por ejemplo, es posible que necesite personalizar las colas para determinados modos de juego que requieren un tipo de instancia especial o una configuración de tiempo de ejecución. O bien, puede que le interese una cola especial para administrar las posiciones de un torneo u otro evento.
- Métricas de las colas de las sesiones de juego. Puede configurar las colas en función de cómo desee recopilar las métricas de ubicación de las sesiones de juego. Para obtener más información, consulte [Métricas de Amazon GameLift Servers para colas](#).

Creación una cola con múltiples ubicaciones

Recomendamos un diseño con múltiples ubicaciones para todas las colas. Este diseño puede mejorar la velocidad de ubicación y la resiliencia del alojamiento. Se requiere un diseño con múltiples ubicaciones para utilizar los datos de latencia de los jugadores con el fin de que puedan participar en las sesiones de juego con una latencia mínima. Si va a crear colas en múltiples ubicaciones que

utilizan flotas de instancias de spot, siga las instrucciones que se indican en [Reducción de los costos de alojamiento de juegos con flotas de spot](#).

Una forma de crear una cola con múltiples ubicaciones consiste en añadir una [flota con múltiples ubicaciones](#) a una cola. De esa forma, la cola puede colocar las sesiones de juego en cualquiera de las ubicaciones de la flota. También puede añadir otras flotas con diferentes configuraciones o ubicaciones de origen para aumentar la redundancia. Si utiliza una flota de instancias de spot con múltiples ubicaciones, siga las prácticas recomendadas e incluya una flota de instancias bajo demanda con las mismas ubicaciones.

En el siguiente ejemplo, se describe el proceso de diseño de una cola básica con múltiples ubicaciones. En este ejemplo, utilizamos dos flotas: una flota de instancias de spot y una flota de instancias bajo demanda. Cada flota tiene los siguientes lugares Regiones de AWS de ubicación: us-east-1, us-east-2ca-central-1, y us-west-2.

Para crear una cola básica multiubicación con flotas con múltiples ubicaciones, realice el siguiente procedimiento:

1. Elija una ubicación para crear la cola. Puede minimizar la latencia de las solicitudes colocando la cola en una ubicación cercana a donde implementó el servicio de cliente. En este ejemplo, crearemos la cola en us-east-1.
2. Cree una cola nueva y añada nuestras flotas con múltiples ubicaciones como destinos de la cola. El orden de destino determina el modo en que Amazon GameLift Servers ubica las sesiones de juego. En este ejemplo, incluimos primero la flota de instancias de spot y, en segundo lugar, la flota de instancias bajo demanda.
3. Defina el orden de prioridad de la ubicación de las sesiones de juego de la cola. Este orden determina dónde busca primero la cola un servidor de juegos disponible. En este ejemplo, utilizaremos el orden de prioridad predeterminado.
4. Defina el orden de ubicación. Si no define el orden de las ubicaciones, Amazon GameLift Servers utilizará las ubicaciones en orden alfabético.

Game session placement locations

Locations where the queue can place new game sessions.

Locations

Choose locations ▼

ca-central-1
Canada (Central) ✕

us-west-2
US West (Oregon) ✕

us-east-2
US East (Ohio) ✕

us-east-1
US East (N. Virginia) ✕

Destination order

An ordered list of fleets and aliases that the queue can use for game session placement.

	Region	Type	Name	
⋮	us-east-1 ▼	Fleet ▼	TestFleet-SPOT ▼	Remove
⋮	us-east-1 ▼	Fleet ▼	TestFleet-ONDEMAND ▼	Remove
Add Destination				

Game session placement priority

The values that GameLift uses to prioritize game session placement. The default order is latency, cost, destination, and location.



Latency

Prioritize locations with the lowest average player latency.



Cost

Prioritize destinations with the lowest current hosting cost.



Destination

Prioritize based on the defined destination order.



Location

Prioritize based on the defined location order.

▼ Location order

An ordered list of locations that the queue can use for game session placement.

Location

ca-central-1



Remove

us-east-1



Remove

us-east-2



Remove

us-west-2



Remove

Add locations

Evaluación de métricas de cola

Utilice las métricas para evaluar el rendimiento de las colas. Puede ver las métricas relativas a las colas en la [consola de Amazon GameLift Servers](#) o en Amazon CloudWatch. Para ver una lista y descripciones de métricas de colas, consulte [Métricas de Amazon GameLift Servers para colas](#).

Las métricas de colas pueden proporcionar información sobre los siguientes aspectos:

- Rendimiento general de la cola: las métricas de la cola indican el grado de éxito con el que una cola responde a las solicitudes de ubicación. Esas métricas también pueden ayudarle a identificar cuándo y por qué fallan las ubicaciones. En el caso de las colas con flotas con escalado manual, las métricas `AverageWaitTime` y `QueueDepth` pueden indicar cuándo debe ajustar la capacidad de una cola.
- Rendimiento del algoritmo de FleetIQ: en el caso de las solicitudes de ubicación que utilizan el algoritmo de FleetIQ, las métricas muestran la frecuencia con la que el algoritmo encuentra la ubicación idónea para las sesiones de juego. La ubicación puede priorizar el uso de los recursos con la menor latencia de jugadores o de los recursos con el menor costo. También hay métricas de error que identifican los motivos más comunes por los que Amazon GameLift Servers no puede encontrar una ubicación idónea. Para obtener más información sobre las métricas, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).
- Ubicaciones específicas por ubicación: en el caso de las colas con múltiples ubicaciones, las métricas muestran las ubicaciones correctas por ubicación. En el caso de las colas que utilizan el algoritmo de FleetIQ, estos datos proporcionan información útil sobre de dónde se produce la actividad de los jugadores.

Cuando evalúe las métricas de rendimiento del algoritmo de FleetIQ, tenga en cuenta las siguientes recomendaciones:

- Para realizar un seguimiento de la tasa de búsqueda de una ubicación idónea en la cola, utilice la métrica `PlacementsSucceeded` en combinación con las métricas de FleetIQ para obtener la latencia más baja y el precio más bajo.
- Para aumentar la tasa de búsqueda de una ubicación idónea en una cola, revise las siguientes métricas de error:
 - Si el valor de `FirstChoiceOutOfCapacity` es alto, ajuste la escala de capacidad para las flotas de la cola.
 - Si la métrica de error `FirstChoiceNotViable` es alta, compruebe las flotas de instancias de spot. Las flotas de instancias de spot no se consideran viables cuando la tasa de interrupción es muy alta en un determinado tipo de instancia. Para solucionar este problema, cambie la cola para que utilice flotas de instancias de spot con otros tipos de instancias. Le recomendamos que incluya flotas de instancias de spot con diferentes tipos de instancias en cada ubicación.

Priorización de la ubicación de las sesiones de juego.

Amazon GameLift Servers utiliza un algoritmo para determinar cómo priorizar los destinos de una cola y determinar dónde ubicar una nueva sesión de juego. El algoritmo se basa en un conjunto ordenado de criterios. Puede utilizar el orden de prioridad predeterminado o personalizar el orden. Puede editar el orden de prioridad de una cola en cualquier momento.

Orden de prioridad predeterminado

1. **Latencia:** si la solicitud de ubicación de la sesión de juego incluye datos de latencia específicos de cada ubicación para los jugadores, Amazon GameLift Servers calcula la latencia media de los jugadores en cada ubicación e intenta colocar una sesión de juego en la ubicación de la flota con la media más baja.
2. **Costo:** si una solicitud no incluye datos de latencia o si varias flotas tienen la misma latencia, Amazon GameLift Servers evalúa el costo de alojamiento de cada flota. El costo de alojamiento de una flota varía en función del tipo de flota (spot o bajo demanda), el tipo de instancia y la ubicación.
3. **Destino:** si varias flotas tienen una latencia y unos costos idénticos, Amazon GameLift Servers las prioriza basándose en el orden de destino, según se especifique en la configuración de la cola.
4. **Ubicación:** en el caso de las colas con flotas con múltiples ubicaciones, si todos los demás criterios son iguales, Amazon GameLift Servers prioriza las ubicaciones de la flota en función del orden alfabético.

Personaliza la forma en que una cola prioriza las ubicaciones de las sesiones de juego

Puede elegir personalizar la forma en que una cola prioriza los criterios de ubicación. La cola aplica la priorización personalizada a todas las solicitudes de ubicación de las sesiones de juego que recibe.

Note

Si creáis una configuración de prioridades personalizada y no incluís los cuatro criterios, Amazon GameLift Servers añadirá automáticamente los criterios que falten en el orden predeterminado.

Para personalizar la configuración de prioridades de una cola

Utilice la [consola de Amazon GameLift Servers](#) o la AWS Command Line Interface (AWS CLI) para crear una configuración de prioridad personalizada.

Console

En la [consola de Amazon GameLift Servers](#), puede personalizar las prioridades de una cola al crear una nueva cola o actualizar una cola existente. Seleccione una AWS región en la que trabajar.

Abra la barra de navegación izquierda de la consola y elija Colas. En la página Colas, seleccione una cola existente y elija Editar.

1. Vaya a la sección Prioridad de ubicación de las sesiones de juego. Arrástrelo y suelte cada criterio de prioridad para crear el orden que desee.
2. Vaya a la sección Orden de ubicación. Añada las ubicaciones que quiera priorizar. Esta lista resulta útil cuando la cola tiene flotas con múltiples ubicaciones. Como mínimo, tiene que especificar una ubicación. Primero se priorizan las ubicaciones que especifique aquí, seguidas de todas las demás ubicaciones de los destinos de la cola.
3. Seleccione Save changes (Guardar cambios).

AWS CLI

Utilice el [update-game-session-queue](#) comando con la `--priority-configuration` opción de personalizar el orden de prioridad de una cola. Amazon GameLift Servers actualiza una cola en la AWS región predeterminada actual, o bien, puede añadir una `--region` etiqueta para especificar una región diferente AWS .

El siguiente ejemplo de solicitud agrega o actualiza la configuración de prioridad de una cola específica

```
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" \  

```

Prioriza las ubicaciones según la latencia de los jugadores

Si desea ofrecer a los jugadores la mejor experiencia de juego posible y garantizar una latencia mínima, siga estos pasos para configurar el sistema de ubicación de las sesiones de juego:

- Configura tu cola para que priorice la latencia a la hora de elegir dónde colocar las sesiones de juego. La latencia aparece en la parte superior de la lista de prioridades de forma predeterminada. También puede personalizar la configuración de prioridades de la cola y elegir dónde colocar la latencia en orden de prioridad.
- Configura las políticas de latencia de los jugadores para tu cola. Las políticas de latencia te permiten establecer límites estrictos en cuanto a la cantidad de latencia permitida en la ubicación de una sesión de juego. Si Amazon GameLift Servers no puede completar una sesión de juego sin superar los límites, se agotará el tiempo de espera de la solicitud de colocación y no será válida. Puede configurar una política de latencia única o crear una serie de políticas que reduzcan gradualmente el límite de latencia a lo largo del tiempo. Con una serie de políticas, puede especificar límites de latencia iniciales muy bajos y, aun así, dar cabida a los jugadores con latencias más altas tras un breve retraso. Para obtener más información sobre las políticas de latencia, consulte [Creación de una política de latencia de jugadores](#).
- Cuando realices solicitudes de ubicación de sesiones de juego (consulta [StartGameSessionPlacement](#)), incluye los datos de latencia de cada jugador. Los datos de latencia de los jugadores incluyen un valor para cada ubicación posible en la que se pueda ubicar una sesión de juego. Por ejemplo, en el caso de una cola que coloca las sesiones de juego en Regiones de AWS us-east-2 y ca-central-1, los datos de latencia podrían tener el siguiente aspecto:

```
"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" }  
]
```

Para obtener mediciones de latencia precisas, utilice señalizadores de pings de UDP de Amazon GameLift Servers. Estos puntos de conexión le permiten medir la latencia real de la red UDP

entre los dispositivos de los jugadores y cada una de las posibles ubicaciones de alojamiento, de forma que pueda tomar decisiones de ubicación más precisas que utilizando pings de ICMP. Para obtener más información sobre el uso de señalizadores de pings de UDP para medir la latencia, consulte [Señalizadores de pings de UDP](#).

Prioriza las ubicaciones por ubicación

Puede configurar una cola para que las ubicaciones de las sesiones de juego se basen en una lista priorizada de ubicaciones geográficas. La ubicación es uno de los criterios que determinan la forma en que una cola elige dónde colocar una nueva sesión de juego. De forma predeterminada, la ubicación se prioriza en cuarto lugar, después de la latencia, el costo y el destino.

En el caso de la ubicación de la sesión de juego, el destino y la ubicación tienen significados algo diferentes:

- El término destino hace referencia a una flota específica e incluye todos los recursos de alojamiento de la flota, estén donde estén desplegados. Al priorizar por destino, Amazon GameLift Servers puede hacer una ubicación en cualquier ubicación de la flota. Las flotas administradas con múltiples ubicaciones y las flotas Anywhere pueden tener recursos de alojamiento que se implementen en una o más ubicaciones.
- La ubicación hace referencia a una posición geográfica específica en la que se implementan los recursos de alojamiento de una flota. Una flota puede tener varias ubicaciones, que pueden incluir Regiones de AWS Zonas Locales o ubicaciones personalizadas (para una flota de Anywhere). Una flota administrada con una sola ubicación tiene una ubicación y siempre es una Región de AWS. Una flota administrada con múltiples ubicaciones tiene una región de origen y una o más ubicaciones remotas. Una flota de Anywhere tiene una o más ubicaciones personalizadas.

Al priorizar las ubicaciones por ubicación, Amazon GameLift Servers busca los destinos de cola que incluyan la ubicación prioritaria y busca en ellos un recurso de alojamiento disponible. Si hay varios destinos con la ubicación prioritaria, Amazon GameLift Servers pasa al siguiente criterio de prioridad (costo, latencia, destino).

Puede influir en la priorización de las ubicaciones de una cola.

- Configura la forma en que la cola gestiona todas las solicitudes de ubicación de las sesiones de juego:

- Añada una configuración de prioridad a la cola. La configuración de prioridad de una cola incluye una lista ordenada de ubicaciones. Puede especificar una o varias ubicaciones para priorizar. Esta lista no excluye ninguna ubicación, simplemente indica a Amazon GameLift Servers dónde buscar primero un recurso de alojamiento disponible. Un uso común de una lista de ubicaciones ordenadas es cuando se quiere canalizar la mayor parte del tráfico a una o más ubicaciones geográficas específicas y utilizar ubicaciones adicionales como capacidad de respaldo. Para añadir una configuración prioritaria, llame [UpdateGameSessionQueue](#).
- Añada una configuración de filtro a la cola. Una configuración de filtro es una lista de permitidos para la cola. Indica a Amazon GameLift Servers que debe ignorar las ubicaciones que no estén en la lista cuando se busca un recurso de alojamiento disponible. Hay dos usos comunes para la configuración de un filtro. En primer lugar, en el caso de las flotas con múltiples ubicaciones, puede utilizar un filtro para excluir algunas de las ubicaciones de la flota. En segundo lugar, es posible que desee prohibir temporalmente las colocaciones en una ubicación determinada; por ejemplo, una ubicación podría estar experimentando problemas transitorios. Como puede actualizar la configuración de filtros de una cola en cualquier momento, puede añadir y eliminar fácilmente ubicaciones según sea necesario. Agregue una configuración de filtro mediante una llamada [UpdateGameSessionQueue](#).
- Utilice instrucciones especiales para las solicitudes de ubicación individuales:
 - Incluya una lista de anulaciones de prioridad en una solicitud de ubicación de sesión de juego. Puede proporcionar una lista de ubicaciones prioritarias alternativa con cualquier [StartGameSessionPlacement](#) solicitud. De hecho, esta lista reemplaza la priorización configurada en la cola para las ubicaciones solo para esa solicitud. No afecta a ninguna otra solicitud. Esta función de anulación tiene algunos requisitos:
 - Utilice una lista de anulación solo con una cola que tenga una configuración de prioridad con LOCATION como máxima prioridad.
 - No incluyas los datos de latencia de los jugadores en la misma solicitud de ubicación. Incluir datos de latencia genera conflictos a la hora de priorizar ubicaciones que Amazon GameLift Servers no puede resolver.
 - Decida cómo desea que proceda Amazon GameLift Servers si no encuentra un recurso disponible en la lista de anulaciones de prioridades. Elige entre volver a las otras ubicaciones de la cola o limitar las ubicaciones a la lista de anulaciones. De forma predeterminada, Amazon GameLift Servers recurre para intentar situarse en las demás ubicaciones de la cola.
 - Actualice la configuración del filtro de la cola según sea necesario, por ejemplo, añadiendo ubicaciones a la lista de anulaciones. La lista de anulaciones no invalida la lista de filtros.

Creación de una política de latencia de jugadores

Si las solicitudes de ubicación incluyen datos de latencia de los jugadores, Amazon GameLift Servers busca las sesiones de juego en las ubicaciones con la latencia media más baja de todos los jugadores. Ubicar las sesiones de juego en función de la latencia media de los jugadores impide que Amazon GameLift Servers ubique a la mayoría de los jugadores en juegos con una latencia alta. Sin embargo, Amazon GameLift Servers sigue ubicando a los jugadores con una latencia extrema. Para adaptarse a estos jugadores, cree políticas de latencia para los jugadores.

Una política de latencia de jugadores impide que Amazon GameLift Servers ubique una sesión de juego solicitada en cualquier lugar en el que los jugadores de la solicitud experimenten una latencia superior al valor máximo. Las políticas de latencia de los jugadores también pueden impedir que Amazon GameLift Servers haga coincidir las solicitudes de sesión de juego con las de los jugadores con una latencia más alta.

Tip

Para administrar reglas específicas de latencia, como exigir una latencia similar a todos los jugadores de un grupo, puede utilizar [Amazon GameLift Servers FlexMatch](#) para crear reglas de emparejamiento basadas en la latencia.

Por ejemplo, considere esta cola con un tiempo de espera de 5 minutos y las siguientes políticas de latencia de los jugadores:

1. Si se pasa 120 segundos buscando una ubicación donde todas las latencias de los jugadores son de menos de 50 milisegundos.
2. Si se pasa 120 segundos buscando una ubicación donde todas las latencias de los jugadores son de menos de 100 milisegundos.
3. Si se emplea el tiempo de la cola restante en buscar una ubicación donde todas las latencias de los jugadores son de menos de 200 milisegundos.

Create queue

Queue settings

Name

The name must be unique and have 1-128 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen).

Timeout

Specify how long GameLift tries to place a game session before stopping.

 seconds

Must be 10-600 seconds.

 We recommend setting player latency policies, unless you're using GameLift FlexMatch.



Player latency policies - *optional*

Add policies to help place players into games with lower latency. Use multiple policies to reduce latency requirements per policy so that each player eventually finds a match.

0 seconds left to allocate

100%

Period start

Seconds

Period end

Seconds

Max player latency

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Seconds

Seconds

Milliseconds

Remove

Add policy

Creación de una cola para instancias de spot

Puede conseguir ahorros significativos en los costos de alojamiento utilizando flotas de spot.

Para obtener más información sobre las flotas de spot y cómo utilizarlas, consulte [Instancias bajo demanda frente a instancias de spot](#).

Si su solución de alojamiento de juegos incluye flotas de spot, debe usar una cola de ubicación de las sesiones de juego. Amazon GameLift Servers usa colas para buscar en varios recursos de alojamiento de juegos y seleccionar el mejor disponible para alojar una nueva sesión de juego. Con las flotas de Spot, las colas son especialmente importantes para minimizar los costos de alojamiento y evitar posibles interrupciones de Spot. Este tema te ayuda a configurar una cola flexible que pueda seguir albergando partidas para los jugadores incluso en caso de interrupciones, ralentizaciones o cortes de servicio. Puede personalizar la forma en que la cola prioriza los recursos de alojamiento disponibles en función de varios factores, incluido el costo del alojamiento.

¿Está utilizando FlexMatch para realizar el emparejamiento? Puede usar una cola con flotas de spot para realizar ubicaciones de sesiones de juego para sus emparejamientos.

Tareas de implementación para las flotas de Spot

Al crear o actualizar su solución de alojamiento de juegos para usar las flotas de spot, complete las siguientes tareas. Para obtener información más detallada sobre cómo crear colas que optimice la disponibilidad y la resiliencia de spot, consulte [Reducción de los costos de alojamiento de juegos con flotas de spot](#).

1. Elija y cree un conjunto de destinos de flota para la cola de sesiones de juego.

Empiece por decidir dónde desea que la cola ubique las sesiones de juego. Una cola puede buscar en varias flotas para encontrar la mejor ubicación posible. Cada flota tiene un tipo de instancia, pero puede tener múltiples ubicaciones geográficas. Las colas con flotas que ofrecen variedad tanto en ubicación como en tipo de instancia tienen más probabilidades de tener éxito en las ubicaciones. Consulte estas prácticas recomendadas para diseñar una cola optimizada para Spot eficaz y resiliente.

2. Cree su cola de sesiones de juego optimizada para spot.

Cree una cola y configúrela para sus flotas de spot. Consulte [Creación de una cola de sesión de juego](#) para obtener ayuda sobre la creación y la configuración de la nueva cola. Puede usar la Amazon GameLift Servers consola o la AWS CLI para crear o editar una cola.

- Añada los destinos de la flota del paso 1.
- Priorice el pedido de destino según corresponda. De forma predeterminada, Amazon GameLift Servers prioriza por costo antes que por destino, por lo que el orden de destino se usa solo cuando los costos más bajos entre destinos son iguales.

- Si desea priorizar el costo del alojamiento del juego antes que la latencia de los jugadores, establezca una prioridad de ubicación personalizada. Consulte [Priorización de la ubicación de las sesiones de juego](#).

3. Actualice otros componentes de la solución para usar la nueva cola.

Si la solución utiliza una cola optimizada para spot para iniciar nuevas sesiones de juego, la cola evita automáticamente que las sesiones de juego tengan flotas con una alta probabilidad de interrupción. En su lugar, busca en todas las flotas viables recursos que coincidan con tus prioridades definidas, como la latencia de los jugadores, los costos de alojamiento y el orden de destino.

- Si no utiliza FlexMatch: actualice el servicio de backend para especificar la nueva cola optimizada para spot en las solicitudes de sesión de juego. El servicio de backend envía solicitudes de API a Amazon GameLift Servers en nombre del cliente del juego (mediante `StartGameSessionPlacement()`) y cada solicitud debe especificar un nombre de cola. Para contribuir a la implementación de ubicaciones de sesión de juego en su cliente de juego, consulte [Creación de sesiones de juego](#).
- Si utiliza FlexMatch: actualice la configuración de emparejamiento para enviar las solicitudes de sesión de juego a la nueva cola optimizada para spot. Cuando el sistema de emparejamiento forma una pareja de jugadores, envía una solicitud de ubicación de la sesión de juego a la cola designada para iniciar una nueva sesión de juego para la partida. Solo las configuraciones de emparejamiento con el modo FlexMatch administrado pueden designar una cola de ubicación. Puede actualizar una configuración de emparejamiento mediante la AWS CLI o la Amazon GameLift Servers consola (consulte [Editar una configuración de emparejamiento](#)).

4. Evalúe el rendimiento de las colas y las flotas de spot.

Consulta Amazon GameLift Servers las métricas en la Amazon GameLift Servers consola o en Amazon CloudWatch para revisar el rendimiento. Para obtener más información acerca de las métricas de Amazon GameLift Servers, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#). Entre las métricas principales se incluyen:

- Tasa de interrupción: utilice las métricas `InstanceInterruptions` y `GameSessionInterruptions` para realizar un seguimiento del número y frecuencia de las interrupciones relacionadas con spot para instancias y sesiones de juego. Las sesiones de juego en instancias reclamadas tienen un estado de `TERMINATED` y un motivo de estado de `INTERRUPTED`.

- Efectividad de las colas: realice un seguimiento de las tasas de éxito en las ubicaciones, el tiempo medio de espera y la profundidad de las colas para comprobar que las flotas de spot no afectan al rendimiento de las colas.
- Uso de la flota: supervise los datos sobre las instancias, las sesiones de juego y las sesiones de los jugadores. El uso de las flotas bajo demanda puede ser un indicador de que las colas no están utilizando ubicaciones en las flotas de spot para evitar interrupciones.

Prácticas recomendadas para colas con flotas de spot

Siga las siguientes prácticas recomendadas al crear flotas y colas para instancias de spot.

- Amplíe la cobertura geográfica de la cola. Incluso si tus jugadores están agrupados en una sola Región de AWS, añada ubicaciones adyacentes a tu flota de Spot. Este enfoque mejora la capacidad de la cola para mantener su capacidad durante las ralentizaciones regionales, los cortes de suministro y las interrupciones de spot. Las flotas con múltiples ubicaciones funcionan tanto con instancias de spot como con instancias bajo demanda.
- Diversifique la cobertura de tipos de instancias de la cola. Amazon GameLift Servers evalúa la viabilidad de Spot en función del tipo de instancia, por lo que tener flotas de Spot con varios tipos de instancias reduce la posibilidad de que varias flotas de Spot no sean viables al mismo tiempo. Incluye al menos dos flotas de Spot con distintos tipos de instancias en cada ubicación.

Note

Los precios se basan en las instancias que utilice, no en la cantidad de flotas. Gestionar cinco flotas con 10 instancias cada una equivale a ejecutar una flota con 50 instancias con un costo similar. Los precios varían según el tipo de instancia, el tamaño y la ubicación.

Consejos para agrupar los tipos de instancias de spot:

- Use tipos de instancias de la misma familia, como `m6g.medium`, `m6g.large` y `m6g.xlarge`. Los tipos de instancias más grandes cuestan más, pero también pueden albergar más sesiones de juego a la vez.
- Seleccione tipos de instancias ampliamente disponibles. Por lo general, las familias de generaciones anteriores (como C5, M5 y R5) y las de tamaño común (como `.large`, `.xlarge` y `.2xlarge`) tienen una mejor disponibilidad.

- Consulte el historial de precios de 30 a 90 días en la consola de Amazon GameLift Servers. Busque tipos de instancias con patrones de disponibilidad consistentes.
- Use la consola de Amazon GameLift Servers, la herramienta de creación de flotas, para explorar la cobertura de ubicaciones para los tipos de instancia.
- Añada flotas bajo demanda para obtener capacidad de respaldo. El alojamiento de juegos puede cambiar a flotas bajo demanda siempre que las flotas de spot no estén disponibles. Coloque al menos una flota bajo demanda en cada ubicación para mantener una baja latencia de jugadores. Añada el escalado automático a las flotas bajo demanda de respaldo para mantener las flotas bajo demanda reducidas verticalmente hasta que las necesite.
- Asigne alias a todos los destinos de la flota. Crea alias para cada uno de los destinos de tu cola. Los alias hacen que sea más fácil y eficiente siempre que necesites sustituir flotas.
- Aplique una estrategia de priorización de colas. Puede personalizar la forma en que una cola prioriza dónde ubicar las sesiones de juego (consulte [Priorización de la ubicación de las sesiones de juego](#) para obtener más información). En el caso de las colas optimizadas para spot, la priorización por costo garantiza que se utilicen flotas de spot de bajo costo siempre que sea posible.

También puede priorizar determinadas flotas especificando un orden de destino. Por ejemplo, algunos usuarios designan un conjunto de flotas principales para su uso habitual y también un conjunto de flotas secundarias como respaldo. En este escenario, defina el orden de destino de la cola para que aparezcan primero las flotas principales. A continuación, configure el orden de prioridad de la cola con el destino seguido del costo.

Personalizaciones de recursos de alojamiento

En esta sección se proporcionan opciones avanzadas para configurar y administrar la infraestructura de Amazon GameLift Servers con el fin de cumplir requisitos operativos, de costos y de rendimiento específicos. En concreto, los temas de esta sección describen cómo personalizar los recursos de alojamiento administrados de Amazon GameLift Servers para que se adapten mejor a su juego y a sus jugadores.

Estas son algunas de las preguntas que deberá formularse:

- ¿Dónde va implementar los recursos de alojamiento para los jugadores? La latencia de juego es un factor importante a la hora de seleccionar las ubicaciones geográficas de la flota, pero hay otros

factores que varían en función de la ubicación, como la disponibilidad y el costo de cada tipo de recurso.

- ¿Qué tipos de EC2 instancias se adaptarán mejor a tu juego? De entre los tipos de instancias disponibles, elija aquellos que ofrezcan la mejor combinación posible de arquitectura de computación, memoria, almacenamiento y capacidad de red.
- ¿Qué tamaño de tipo de instancia necesito? Deberá elegir el tamaño de tipo de instancia en función de los requisitos de recursos (memoria y CPU) del software del servidor de juegos, así como en función de otros factores.
- ¿Debe la flota utilizar instancias bajo demanda o instancias de spot? Piense en si puede aprovechar los precios de spot más bajos y si mitiga Amazon GameLift Servers suficientemente la posibilidad de interrupciones en las sesiones de juego.
- ¿Cómo quiero que se ejecute el software del servidor de juegos en cada instancia de la flota? La configuración del tiempo de ejecución le dice a Amazon GameLift Servers qué software de servidor debe ejecutar y cómo hacerlo.
- En el caso de las flotas de contenedores, ¿es válida la configuración predeterminada para mi juego? Amazon GameLift Servers hace gran parte del trabajo para optimizar las configuraciones de su flota de contenedores, pero puede personalizar la mayoría de los ajustes de configuración.

Elección de los recursos de computación para una flota administrada

En el caso del alojamiento administrado por Amazon GameLift Servers, que incluye el EC2 gestionado y los contenedores administrados, el servicio implementa los servidores de juegos en las flotas de recursos de computación en la Nube de AWS. Cuando crea una flota administrada, le conviene configurar los recursos de alojamiento para que se adapten mejor al juego. En este tema se analizan los puntos de decisión clave a la hora de elegir y configurar tus flotas de alojamiento de juegos.

Note

Si está creando una solución híbrida con flotas de Anywhere y administradas por Amazon GameLift Servers, utilice estos temas para diseñar flotas administradas que complementen sus propios recursos autoadministrados. Consulte [Implementación de flotas de alojamiento para Amazon GameLift Servers](#).

Temas

- [Ubicaciones geográficas](#)
- [Sistemas operativos](#)
- [Tipos de instancias](#)
- [Instancias bajo demanda frente a instancias de spot](#)
- [Cuotas de servicio](#)

Ubicaciones geográficas

Tenga en cuenta dónde desea implementar los servidores de juegos. En general, querrás colocar tus servidores de juegos lo más cerca posible de tus jugadores para ofrecer la mejor experiencia de juego posible. Para el alojamiento Amazon GameLift Servers gestionado, puedes elegir colocar los servidores de juegos en cualquiera de las Zonas Locales Regiones de AWS y compatibles. Si está creando una solución híbrida, considera cómo la implementación de flotas administradas puede complementar las ubicaciones de sus flotas autoadministradas de Amazon GameLift Servers Anywhere.

Para la mayoría de los escenarios de desarrollo y pruebas, la implementación en una única ubicación tiene sentido. A medida que se prepara para el lanzamiento y más adelante, hay muchos motivos para realizar la implementación en varias ubicaciones geográficas. Estas incluyen dar soporte a un grupo amplio de jugadores y mejorar la resiliencia y la fiabilidad generales del alojamiento de juegos. Varias ubicaciones también pueden mejorar la experiencia de los jugadores, ya que aceleran la ubicación de las sesiones de juego y ofrecen más opciones a la hora de optimizar las ubicaciones en función de la latencia y el costo.

Para ver una lista de las ubicaciones admitidas Amazon GameLift Servers y más información sobre las ubicaciones de todos los tipos de flota, consulte [Ubicaciones de servicio de Amazon GameLift Servers](#).

Flotas con múltiples ubicaciones

Una flota administrada única puede implementar recursos en varias ubicaciones. Puede configurar la capacidad de cada ubicación de una flota con múltiples ubicaciones.

Ventajas de utilizar una flota con múltiples ubicaciones:

- Implementación y administración simplificadas de la flota: usted suministra el software del servidor del juego y la configuración de la flota, y Amazon GameLift Servers lo implementa en las instancias

de la flota con múltiples ubicaciones (se crea una vez y se implementa en cualquier lugar). En una flota de producción, puede ver y gestionar todas las ubicaciones de una flota en lugar de tener que administrar varias flotas, cada una ubicada en una región diferente.

- Disponibilidad de zona local: si desea utilizar una zona local, debe crear una flota de múltiples ubicaciones con una ubicación Región de AWS local y zonas locales como ubicaciones remotas. Las Zonas Locales son extensiones Regiones de AWS que pueden ofrecer una latencia aún menor a las áreas y los clientes que la necesitan. Puede añadir una zona local a cualquier flota con múltiples ubicaciones; no es necesario incluir la Región de AWS principal de la zona local.
- Compatibilidad con las colas de las sesiones de juego: puede crear colas de ubicación de las sesiones de juego con una o más flotas con múltiples ubicaciones. Este enfoque proporciona flexibilidad a la cola a la hora de priorizar y elegir ubicaciones en las que alojar una nueva sesión de juego.
- Utilización eficiente de los recursos: con el escalado automático activado, Amazon GameLift Servers puede optimizar mejor el escalado de la capacidad en todas las ubicaciones de una flota.

Consejos para usar flotas con múltiples ubicaciones:

- Comprueba las cuotas en cuanto al número de ubicaciones por Región de AWS flota. Consulte [cuotas de Amazon GameLift Servers servicio](#).
- No todos los tipos de instancias están disponibles en todas las ubicaciones. Según las ubicaciones que elija, es posible que tenga opciones limitadas de tipos de instancia. La Amazon GameLift Servers consola proporciona herramientas útiles que le ayudarán a encontrar el equilibrio adecuado entre ubicaciones y tipos de instancias.
- Considere utilizarla [Señalizadores de pings de UDP](#) para recopilar datos sobre la latencia de los jugadores en todas las ubicaciones de su flota. Amazon GameLift Servers puede usar estos datos para optimizar las sesiones de juego con una latencia baja y evitar que los jugadores se unan a sesiones con una latencia inaceptablemente alta. Estos puntos de conexión especiales aceptan mensajes UDP en lugar de los pings de ICMP tradicionales, lo que proporciona mediciones de latencia precisas para ayudarlo a seleccionar las ubicaciones óptimas de la flota.

Sistemas operativos

Todas las instancias de una flota administrada se implementan con una imagen de máquina de Amazon (AMI) que proporciona un entorno de tiempo de ejecución completo para el software del servidor de juegos. En el caso de las flotas de EC2 administradas, debe especificar el sistema operativo de la compilación del servidor de juegos al cargarla en Amazon GameLift Servers. En el

caso de las flotas de contenedores administradas, especifique el sistema operativo en la definición del grupo de contenedores. Para obtener más información sobre las versiones más recientes de la AMI, consulte [Versiones de AMI de Amazon GameLift Servers](#).

Las versiones de AMI se actualizan periódicamente. Cuando crees una flota nueva, Amazon GameLift Servers asigna la última versión disponible de la AMI que hayas seleccionado para la versión del juego. Todas las instancias que se implementan en esa flota utilizan la misma versión. Para mantener su versión de AMI actualizada con las últimas actualizaciones de seguridad y software, debe reemplazar sus flotas periódicamente. Como práctica recomendada, recomendamos reemplazar las flotas cada 30 días para mantener el entorno de tiempo de ejecución de los servidores de juegos. Para obtener instrucciones, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Tipos de instancias

El tipo de instancia de una flota administrada determina el tipo de hardware que se implementa en todas las instancias de flota y, por lo general, los tipos de instancia están disponibles en varios tamaños. Todas las flotas Amazon GameLift Servers administradas utilizan instancias de Amazon EC2 y admiten una amplia variedad de tipos de instancias que ofrecen diferentes combinaciones de potencia informática, memoria, almacenamiento y funciones de red. La disponibilidad de los tipos de instancia varía en función de las ubicaciones que elija.

La Amazon GameLift Servers consola proporciona herramientas útiles que le ayudarán a encontrar los tipos de instancias adecuados para la compilación del juego y las ubicaciones de implementación. En el caso de las flotas de contenedores administradas, la consola también ofrece instrucciones sobre los requisitos de memoria y potencia de la CPU del juego.

Al elegir entre los tipos de instancias disponibles para su juego, tenga en cuenta los siguientes aspectos:

- La arquitectura de cómputo de tu servidor de juegos: x64 o Arm (AWS Graviton).

Note

Las instancias de Graviton ARM requieren una compilación de servidor para una AMI de Linux. Se requiere el SDK del servidor 5.1.1 o posterior para C++ y C#. Se requiere el SDK del servidor 5.1.1 o posterior para continuar. Estas instancias no out-of-the-box admiten la instalación de Mono en Amazon Linux 2023 (AL2023) o Amazon Linux 2 (AL2).

- Los requisitos informáticos, de memoria y de almacenamiento de la compilación del servidor de juegos.
- El tamaño del tipo de instancia. Además de cumplir con los requisitos de los ejecutables del software del servidor de juegos, los tipos de instancia más grandes pueden ejecutar varios and/or contenedores de procesos del servidor de juegos en cada instancia. Los factores a tener en cuenta incluyen el costo (es más económico ejecutar unas pocas instancias grandes o muchas instancias pequeñas). Ten en cuenta también cómo puede verse afectada la capacidad de las sesiones de juego si se añaden o quitan instancias durante los eventos de escalado de la flota o cuando se cierran instancias que no funcionan correctamente. Si cada instancia ejecuta varios procesos del servidor de juegos al mismo tiempo, añadir o eliminar una instancia puede afectar considerablemente a la capacidad de alojamiento del juego.

Para obtener más información sobre los tipos de instancias, consulte [Tipos de instancias Amazon EC2](#).

Instancias bajo demanda frente a instancias de spot

Las instancias bajo demanda y las instancias de spot de Amazon EC2 ofrecen el mismo hardware y rendimiento, pero difieren en cuanto a disponibilidad y costo.

instancias bajo demanda

Puede adquirir una instancia bajo demanda cuando la necesite y mantenerla todo el tiempo que desee. Las instancias bajo demanda tienen un costo fijo, lo que significa que usted paga por la cantidad de tiempo que las utilice. No es necesario asumir compromisos a largo plazo.

Spot Instances

Las instancias puntuales pueden ofrecer una alternativa rentable a las instancias bajo demanda al utilizar la AWS capacidad informática no utilizada. Los precios de las instancias puntuales fluctúan en función de la oferta y la demanda de cada tipo de instancia en cada ubicación. AWS puede recuperar instancias puntuales con una notificación de dos minutos cada vez que necesite recuperar la capacidad, y las sesiones de juego que se estén ejecutando activamente en una instancia recuperada se interrumpen.

Amazon GameLift Servers ofrece varias herramientas que ayudan a reducir la probabilidad de que Spot interrumpa tus sesiones de juego. Un algoritmo de viabilidad de spot rastrea los datos históricos del tipo de instancia para anticipar cuándo el riesgo de interrupción alcanza un punto crítico y evita colocar nuevas sesiones de juego en instancias de spot de ese tipo. Si se produce una interrupción,

los servidores de juegos pueden usar la notificación para finalizar una sesión de juego para los jugadores sin problemas.

El alojamiento de juegos con las flotas de Spot debe utilizar una cola para ubicar las sesiones de juego. Una cola permite priorizar las ubicaciones de las sesiones de juego en función de la viabilidad, el costo y otros factores de la flota de Spot. Consulta estos temas para obtener más información sobre cómo aprovechar Spot como alojamiento de servidores de juegos:

- [Reducción de los costos de alojamiento de juegos con flotas de spot](#)
- [Creación de una cola para instancias de spot](#)

Cuotas de servicio

Puede ver las cuotas de servicio predeterminadas de Amazon GameLift Servers y su estado actual para su Cuenta de AWS con las siguientes herramientas:

- Para obtener información general sobre las cuotas de servicio de Amazon GameLift Servers, consulte [Puntos de conexión y cuotas de Amazon GameLift Servers](#) en la Referencia general de AWS.
- Para ver una lista de los tipos de instancia disponibles por ubicación de la cuenta, abra la página [Cuotas de servicio](#) de la consola de Amazon GameLift Servers. Esta página también muestra el uso actual de su cuenta para cada tipo de instancia en cada ubicación.
- Para obtener una lista de las cuotas actuales de tu cuenta (por ejemplo, tipos de instancias) por región, ejecuta el comando AWS Command Line Interface (AWS CLI). [describe-ec2-instance-limits](#) Este comando devuelve el número de instancias activas de los que dispone en su región predeterminada (o en otra región que especifique).

Mientras se prepara para iniciar el juego, complete el cuestionario de lanzamiento en la [consola de Amazon GameLift Servers](#). El equipo de Amazon GameLift Servers utiliza el cuestionario de lanzamiento para determinar las cuotas y los límites correctos para el juego.

Personalización de una flota de contenedores de Amazon GameLift Servers

En los temas de esta sección, se describen algunas de las características opcionales de los contenedores administrados por Amazon GameLift Servers. Puede optar por utilizar todas o algunas de las características siguientes.

Temas

- [Establecimiento de los límites de recursos](#)
- [Configuración del acceso NVMe a la unidad](#)
- [Designación de contenedores esenciales](#)
- [Configuración de conexiones de red](#)
- [Configure las comprobaciones de estado de los contenedores](#)
- [Establecimiento de dependencias de contenedores](#)
- [Configure una flota de contenedores](#)

Establecimiento de los límites de recursos

Para cada grupo de contenedores, puede determinar cuánta memoria y potencia de cálculo necesita el grupo de contenedores para ejecutar su software. Amazon GameLift Servers se basa en esta información para administrar los recursos en todo el grupo de contenedores. También utiliza esta información para calcular cuántos grupos de contenedores de servidores de juegos puede contener una imagen de flota. También puede definir límites para contenedores individuales.

Puede establecer un límite máximo de memoria y potencia de cálculo para un grupo de contenedores. De forma predeterminada, todos los contenedores del grupo comparten estos recursos. Puede personalizar aún más la administración de recursos estableciendo límites para los contenedores individuales.

Establezca límites opcionales para contenedores individuales

Establecer límites de recursos específicos para cada contenedor te permite ejercer un mayor control sobre cómo los contenedores individuales pueden usar los recursos del grupo. Si no estableces límites específicos para los contenedores, todos los contenedores del grupo comparten los recursos del grupo. El uso compartido ofrece una mayor flexibilidad para usar los recursos donde se necesiten. También aumenta la posibilidad de que los procesos compitan entre sí y provoquen la falla del contenedor.

Defina cualquiera de las siguientes propiedades `ContainerDefinition` para cualquier contenedor.

- `MemoryHardLimitMebibytes`: establezca un límite máximo de memoria para el contenedor. Si el contenedor supera este límite, se reinicia.
- Límite de `Vcpu`: reserve una cantidad mínima de recursos de vCPU para uso exclusivo del contenedor. El contenedor siempre tiene disponible la cantidad reservada. Puede superar este

mínimo en cualquier momento, si hay recursos adicionales disponibles. (1024 unidades de CPU equivalen a 1 vCPU).

Establezca los límites totales de recursos para un grupo de contenedores

Si estableces límites para contenedores individuales, es posible que tengas que modificar la cantidad de recursos de memoria y vCPU que necesita el grupo de contenedores. El objetivo es asignar recursos suficientes para optimizar el rendimiento del servidor de juegos. Amazon GameLift Servers utiliza estos límites para calcular cómo empaquetar los grupos de contenedores de servidores de juegos en una instancia de flota. También los usarás al elegir un tipo de instancia para una flota de contenedores.

Calcule la memoria total y la vCPU necesarias para un grupo de contenedores. Considere lo siguiente:

- ¿Cuáles son todos los procesos que se ejecutan en todos los contenedores del grupo de contenedores? Sume los recursos necesarios para estos procesos. Tome nota de los límites específicos de los contenedores.
- ¿Cuántos procesos simultáneos del servidor de juegos planeas ejecutar en cada grupo de contenedores? Esto lo determinas en la imagen del contenedor del servidor de juegos.

En función de su estimación de los requisitos de los grupos de contenedores, defina las siguientes propiedades `ContainerGroupDefinition`:

- `TotalMemoryLimitMebibytes`: establezca un límite máximo de memoria para el grupo de contenedores. Todos los contenedores del grupo comparten la memoria asignada. Si establece límites de contenedores individuales, el límite de memoria total debe ser igual o superior al límite máximo de memoria específico del contenedor.
- `TotalVcpuLimit`: establezca un límite máximo de vCPU para el grupo de contenedores. Todos los contenedores del grupo comparten los recursos de CPU asignados. Si establece límites de contenedores individuales, el límite total de CPU debe ser igual o superior a la suma de todos los límites de CPU específicos del contenedor. Como práctica recomendada, considere configurar este valor para que duplique la suma de los límites de CPU del contenedor.

Escenario de ejemplo

Supongamos que estamos definiendo un grupo de contenedores de servidores de juegos con los tres contenedores siguientes:

- El contenedor A es nuestro contenedor para servidores de juegos. Estimamos los requisitos de recursos para un servidor de juegos en 512 MiB y 1024 CPU. Planeamos que el contenedor

ejecute un proceso de servidor. Como este contenedor ejecuta nuestro software más crítico, no establecemos ningún límite de memoria ni límite de reserva de vCPU.

- El contenedor B es un contenedor de soporte con requisitos de recursos estimados en 1024 MiB y 1536 CPU. Hemos establecido un límite de memoria de 2048 MiB y un límite de reserva de CPU de 1024 CPU.
- El contenedor C es otro contenedor de soporte. Hemos establecido un límite de memoria dura de 512 MiB y un límite de reserva de CPU de 512 CPU.

Con esta información, establecemos los siguientes límites totales para el grupo de contenedores:

- Límite total de memoria: 7680 MiB. Este valor supera el límite de memoria más alto (1024 MiB).
- Límite total de CPU: 13312 CPU. Este valor supera la suma del límite de CPU (1024+512 CPU).

Configuración del acceso NVMe a la unidad

En las instancias de tipo d, la NVMe unidad se monta automáticamente en el /data directorio durante el inicio del host. Para permitir que los contenedores accedan al almacenamiento SSD, defina la siguiente propiedad: `ContainerGroupDefinition MountPoints`

- `InstancePath`— /data Configúrelo en para hacer referencia a la NVMe unidad montada automáticamente en la instancia host.
- `AccessLevel`— Elija el nivel de acceso adecuado a las necesidades de su contenedor (por ejemplo, `READ_ONLY` o `READ_WRITE`).
- `ContainerPath`— (Opcional) Especifique la ruta en la que se montará la ruta de la instancia dentro del contenedor. Si no se especifica, el valor predeterminado es la ruta de la instancia.

Para obtener más información sobre los puntos de montaje, consulte [ContainerMountPoint](#) la referencia de la API de Amazon GameLift Servers.

Designación de contenedores esenciales

Para un grupo de contenedores por instancia, designa cada contenedor como esencial o no esencial. Los grupos de contenedores por instancia deben tener al menos un contenedor de soporte esencial. El contenedor esencial realiza el trabajo fundamental del grupo de contenedores. Se espera que el contenedor esencial esté siempre funcionando. Si se produce un error, se reinicia todo el grupo de contenedores.

Establezca la propiedad `ContainerDefinition Essential` en verdadero o falso para cada contenedor.

Configuración de conexiones de red

Puede personalizar el acceso a la red para permitir que el tráfico externo se conecte a cualquier contenedor de una flota de contenedores. Por ejemplo, debe establecer conexiones de red con el contenedor que ejecuta los procesos del servidor de juegos para que los clientes del juego puedan unirse y jugar al juego. Los clientes de juegos se conectan a los servidores de juegos mediante puertos y direcciones IP.

En una flota de contenedores, la conexión entre un cliente y un servidor no es directa. Internamente, un proceso de un contenedor escucha en un puerto de contenedores. Externamente, el tráfico entrante se conecta a una instancia de flota mediante un puerto de conexión. Amazon GameLift Servers mantiene las asignaciones entre los puertos contenedores internos y los puertos de conexión externos, de modo que el tráfico entrante se dirija al proceso correcto de la instancia.

Amazon GameLift Servers proporciona una capa adicional de control para sus conexiones de red. Cada flota de contenedores tiene una configuración de permisos de entrada, que le permite controlar el acceso a cada puerto de conexión externo. Por ejemplo, puede eliminar los permisos de todos los puertos de conexión para impedir el acceso a los contenedores de la flota.

Puede actualizar los permisos de entrada, los puertos de conexión y los puertos de contenedores de una flota.

Warning

Si proporciona un valor personalizado `InstanceConnectionPortRange` o `InstanceInboundPermissions`, ya no Amazon GameLift Servers administrará ninguno de los dos valores para su flota. Debe configurar ambos campos para evitar un comportamiento indefinido.

Configuración de puertos de contenedores

Configure los rangos de puertos de contenedores como parte de la definición de cada contenedor. Este es un parámetro obligatorio para la definición de un grupo de contenedores. Debe configurar suficientes puertos para dar cabida a todos los procesos que se ejecutan simultáneamente y que necesitan acceso externo. Algunos contenedores no necesitarán ningún puerto.

El contenedor del servidor de juegos, en el que se ejecutan los servidores de juegos, necesita un puerto para cada proceso del servidor de juegos que se ejecute simultáneamente. El proceso del servidor de juegos escucha en el puerto asignado e informa a Amazon GameLift Servers.

Configuración de puertos de conexión

Configure su flota de contenedores con un conjunto de puertos de conexión. Los puertos de conexión proporcionan acceso externo a las instancias de la flota en las que se utilizan sus contenedores. Amazon GameLift Servers asigna los puertos de conexión y los asigna a los puertos de contenedores según sea necesario.

De forma predeterminada, Amazon GameLift Servers calcula el número de puertos necesarios para todos los grupos de contenedores y establece un rango de puertos para acomodarlos. Le recomendamos encarecidamente que utilice valores calculados de Amazon GameLift Servers, que se actualizan al implementar actualizaciones en la definición de un grupo de contenedores. Si necesita personalizar los rangos de puertos de conexión, utilice las siguientes instrucciones.

Al crear una flota de contenedores, defina un rango de puertos de conexión (consulte [ContainerFleet: InstanceConnectionPortRange](#)). Asegúrate de que el rango tenga suficientes puertos para asignarlos a todos los puertos de contenedores definidos para todos los contenedores de ambos grupos de contenedores de la flota. Para calcular los puertos de conexión mínimos necesarios, utilice la siguiente fórmula:

```
[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]
```

Como práctica recomendada, duplique el número mínimo de puertos de conexión.

Note

El número de puertos de conexión puede limitar potencialmente el número de grupos de contenedores de servidores de juegos por instancia. Si una flota solo tiene puertos de conexión suficientes para un grupo de contenedores de servidores de juegos por instancia, Amazon GameLift Servers implementará solo un grupo de contenedores de servidores de juegos, incluso si las instancias tienen suficiente potencia de cálculo para varios grupos de contenedores de servidores de juegos.

Configuración de permisos de entrada

Los permisos de entrada controlan el acceso externo a una flota de contenedores especificando qué puertos de conexión se deben abrir para el tráfico entrante. Puede usar esta configuración para activar y desactivar el acceso a la red de una flota según sea necesario.

De forma predeterminada, Amazon GameLift Servers calcula el número de puertos necesarios para todos los grupos de contenedores y establece un rango de puertos para acomodarlos. Le recomendamos encarecidamente que utilice valores calculados de Amazon GameLift Servers, que se actualizan al implementar actualizaciones en la definición de un grupo de contenedores. Si necesita personalizar los rangos de puertos de conexión, utilice las siguientes instrucciones.

Al crear una flota de contenedores, defina un conjunto de permisos de entrada (consulte [ContainerFleet: InstanceInboundPermissions](#)). Los puertos de permisos de entrada deben coincidir con los rangos de puertos de conexión de la flota.

Note

Dado que los puertos de contenedores se seleccionan aleatoriamente entre InstanceConnectionPortRange sí, para garantizar que se puedan realizar las conexiones de sesión, todos los puertos de entrada InstanceConnectionPortRange deben estar cubiertos por los puertos de InstanceInboundPermissions

Escenario de ejemplo

Este ejemplo ilustra cómo definir las tres propiedades de conexión de red.

- El grupo de contenedores de servidores de juegos de nuestra flota tiene 1 contenedor, que ejecuta 1 proceso de servidor de juegos.

En la definición del grupo de contenedores de servidores de juegos, establecemos el parámetro PortConfiguration para este contenedor de la siguiente manera:

```
"PortConfiguration": {  
  "ContainerPortRanges": [ { "FromPort": 10, "ToPort": 20, "Protocol": "TCP" } ]  
}
```

- Nuestra flota también tiene un grupo de contenedores por instancia con 1 contenedor. Tiene 1 proceso que necesita acceso a la red. En la definición del contenedor por instancia,

establecemos el parámetro `PortConfiguration` para este contenedor de la siguiente manera:

```
"PortConfiguration": {  
  "ContainerPortRanges": [ { "FromPort": 25, "ToPort": 25, "Protocol": "TCP"} ] }
```

- Nuestra flota está configurada con 20 grupos de contenedores de servidores de juegos por instancia de flota. Con esta información, podemos usar la fórmula para calcular el número de puertos de conexión que necesitamos:
 - Mínimo: 21 puertos [1 puerto contenedor de servidor de juegos * 20 grupos de contenedores de servidores de juegos por instancia + 1 puerto contenedor por instancia]
 - Práctica recomendada: 42 puertos [puertos mínimo * 2]

Al crear la flota de contenedores, configuramos el parámetro `InstanceConnectionPortRange` de la siguiente manera:

```
"InstanceConnectionPortRange": { "FromPort": 1010, "ToPort": 1071 }
```

- Queremos permitir el acceso a todos los puertos de conexión disponibles. Al crear la flota de contenedores, configuramos el parámetro `InstanceInboundPermissions` de la siguiente manera:

```
"InstanceInboundPermissions": [  
  { "FromPort": 1010, "ToPort": 1071, "IpRange": "10.24.34.0/23", "Protocol":  
    "TCP"} ]
```

Configure las comprobaciones de estado de los contenedores

Un contenedor se reinicia automáticamente si sufre una falla en la terminal y deja de funcionar. Si un contenedor se considera esencial, solicita a todo el grupo de contenedores que se reinicie.

Todos los contenedores de servidores de juegos se consideran esenciales automáticamente. Los contenedores Support pueden considerarse esenciales, pero deben tener un mecanismo para informar sobre el estado de salud. También puede establecer comprobaciones de estado para los contenedores de apoyo no esenciales.

Puede definir criterios personalizados adicionales para medir el estado de los contenedores y utilizar una comprobación de estado para comprobar esos criterios. Para configurar una comprobación del estado de un contenedor, puede definirla en una imagen de contenedor de Docker o en la definición

de contenedor. Si configuras una comprobación de estado en la definición del contenedor, esta anulará cualquier configuración de la imagen del contenedor.

Defina las siguientes propiedades `SupportContainerDefinition` para una comprobación del estado del contenedor:

- **Command:** proporcione un comando que compruebe algún aspecto del estado del contenedor. Usted decide qué criterios utilizar para medir la salud. El comando debe dar como resultado un valor de salida de 1 (en mal estado) o 0 (en buen estado).
- **StartPeriod:** especifique un retraso inicial antes de que los fallos en las comprobaciones de estado empiecen a contabilizarse. Este retraso da tiempo al contenedor para iniciar sus procesos.
- **Interval:** decida con qué frecuencia se va a ejecutar el comando de comprobación de estado. ¿Con qué rapidez desea detectar y resolver un fallo en un contenedor?
- **Timeout:** decida cuánto tiempo esperar para que se ejecute correctamente o no antes de volver a intentar ejecutar el comando de comprobación de estado. ¿Cuánto tiempo debe tardar en completarse el comando de comprobación de estado?
- **Retries:** ¿Cuántas veces se debe volver a intentar el comando de comprobación de estado antes de que se registre un error?

Establecimiento de dependencias de contenedores

Dentro de cada grupo de contenedores, puede establecer las dependencias entre los contenedores en función del estado del contenedor. Una dependencia afecta al momento en que el contenedor dependiente puede iniciarse o cerrarse en función del estado de otro contenedor.

Un caso de uso clave de las dependencias es crear secuencias de inicio y cierre para el grupo de contenedores.

Por ejemplo, es posible que desee que el contenedor A comience primero y se complete correctamente antes de que se inicien los contenedores B y C. Para lograrlo, cree primero una dependencia para el contenedor B en el contenedor A, con la condición de que el contenedor A se complete correctamente. A continuación, cree una dependencia para el contenedor C en el contenedor A con la misma condición. Las secuencias de inicio se producen en orden inverso al de cierre.

Configure una flota de contenedores

Al crear una flota de contenedores, tenga en cuenta los siguientes puntos de decisión. La mayoría de estos puntos dependen de la arquitectura y configuración del contenedor.

Decida dónde desea implementar su flota

En general, querrá implementar las flotas geográficamente cerca de los jugadores para minimizar la latencia. Puede desplegar su flota de contenedores en cualquiera Región de AWS que lo Amazon GameLift Servers soporte. Si quieres implementar el mismo servidor de juegos en ubicaciones geográficas adicionales, puedes añadir ubicaciones remotas a la flota, incluidas Regiones de AWS las Zonas Locales. En el caso de las flotas con múltiples ubicaciones, puede ajustar la capacidad de forma independiente en cada ubicación de la flota. Para obtener más información sobre las ubicaciones de flota admitidas, consulte [Ubicaciones de servicio de Amazon GameLift Servers](#).

Puede usar [Señalizadores de pings de UDP](#) para recopilar datos de latencia de red en varias ubicaciones geográficas para anticipar la latencia entre los dispositivos de los jugadores y las posibles ubicaciones de la flota. Estos puntos de conexión especiales aceptan mensajes UDP en lugar de los pings de ICMP tradicionales, lo que proporciona mediciones de latencia precisas para ayudarlo a seleccionar las ubicaciones óptimas de la flota.

Elija un tipo de instancia para la flota.

Amazon GameLift Servers admite una amplia gama de tipos de instancias de Amazon EC2, todas las cuales están disponibles para su uso con una flota de contenedores. La disponibilidad del tipo de instancia y el precio varían según la ubicación. Puede ver una lista de los tipos de instancias compatibles, filtrados por ubicación, en la consola de Amazon GameLift Servers (en Recursos, instancias y cuotas de servicio).

Al elegir un tipo de instancia, ten en cuenta primero la familia de instancias. Las familias de instancias ofrecen varios tipos de combinaciones de CPU, memoria, almacenamiento y capacidades de red. Obtenga más información sobre las [familias de instancias de EC2](#). Dentro de cada familia, puede elegir entre una variedad de tamaños de instancia. Tenga en cuenta los siguientes aspectos al seleccionar un tamaño de instancia:

- ¿Cuál es el tamaño mínimo de instancia que puede soportar tu carga de trabajo? Usa esta información para eliminar cualquier tipo de instancia que sea demasiado pequeña.
- ¿Qué tamaños de tipos de instancias son adecuados para su arquitectura de contenedores? Lo ideal es elegir un tamaño que pueda alojar varias copias del grupo de contenedores de servidores de juegos con un mínimo de espacio desperdiciado.

- ¿Qué granularidad de escalado tiene sentido para tu juego? Escalar la capacidad de la flota implica añadir o eliminar instancias, y cada instancia representa la capacidad de albergar un número específico de sesiones de juego. Tenga en cuenta la cantidad de capacidad que desea añadir o eliminar con cada instancia. Si la demanda de los jugadores varía en miles de un minuto a otro, podría ser conveniente utilizar instancias muy grandes que puedan alojar cientos o miles de sesiones de juego. Por el contrario, es posible que prefiera un control de escalado más detallado con tipos de instancias más pequeñas.
- ¿Hay ahorros de costos disponibles en función del tamaño? Es posible que el costo de algunos tipos de instancias varíe según la ubicación debido a la disponibilidad.

Establezca otros ajustes de flota opcionales

Puede utilizar las siguientes características opcionales al configurar una flota de contenedores:

- Configura tus servidores de juego para acceder a otros AWS recursos. Consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).
- Evita que las sesiones de juego con jugadores activos se terminen de forma prematura durante un evento de reducción vertical.
- Limite el número de sesiones de juego que una persona puede crear en la flota dentro de un lapso de tiempo limitado.

Reducción de los costos de alojamiento de juegos con flotas de spot

A la hora de alojar servidores de juegos multijugador mediante un alojamiento administrado por Amazon GameLift Servers, las instancias de spot pueden ofrecer una alternativa rentable a las instancias bajo demanda. El modelo de precios de spot ofrece el mismo hardware y rendimiento que bajo demanda, pero con posibilidades de un ahorro de costos significativo (de hasta un 70-90 %). Sin embargo, tienen una limitación: cuando AWS necesita recuperar la capacidad, puede recuperar estas instancias con una notificación de interrupción de dos minutos.

Amazon GameLift Servers mitiga el riesgo de interrupciones en el alojamiento del servidor de juegos. Amazon GameLift Servers predice la probabilidad de que se produzcan interrupciones en los tipos de instancias de spot y evita poner en riesgo las sesiones de juego de cualquier instancia. Si se produce una interrupción, lo que será poco frecuente, la notificación le permite finalizar sin problemas una sesión de juego para los jugadores.

Funcionamiento de Amazon GameLift Servers con flotas de spot

Cuando se configuran flotas de spot para el alojamiento de juegos, Amazon GameLift Servers evalúa continuamente los tipos de instancias y las ubicaciones de las flotas de spot para comprobar la viabilidad del alojamiento de juegos.

- El algoritmo de viabilidad de spot analiza los patrones de disponibilidad recientes y las tasas de interrupción históricas de los tipos de instancias de spot por ubicación.
- Basándose en este análisis, Amazon GameLift Servers identifica los tipos de instancias de spot y las ubicaciones en las que las probabilidades de interrupción de las sesiones de juego son inaceptables. Lleva a cabo las siguientes acciones:
 - Marca la combinación del tipo de instancia y la ubicación como temporalmente inviable.
 - Deja de considerar cualquier ubicación de flota de spot inviable a la hora de ubicar las nuevas sesiones de juego. Como resultado de ello, las sesiones de juego solo se ubican en ubicaciones de flota de spot que ofrecen una alta probabilidad de alojar servidores de juegos sin interrupciones.
 - Esto agota las instancias existentes en la flota de spot, incluso si AWS no las reclama, de modo que no tenga que pagar por instancias que no se puedan utilizar para el alojamiento de juegos. Si la protección de las sesiones de juego está activada, las instancias solo se cierran una vez completadas las sesiones de juego activas.
- Amazon GameLift Servers reevalúa continuamente los tipos y ubicaciones de las instancias de las flotas de spot para comprobar la viabilidad del alojamiento de juegos. Cuando un tipo de instancia que antes no era viable vuelve a ser viable en función de los datos históricos actualizados, puede volver a escalar verticalmente la flota de spot; en este caso, Amazon GameLift Servers reanudará la ubicación de las sesiones de juego con ella.

Consideraciones sobre el diseño

A la hora de diseñar una solución que utilice flotas de spot, tenga en cuenta los siguientes aspectos:

- Evalúe la duración de las sesiones de juego: la duración media de las sesiones de juego puede influir en cómo funciona Spot con su juego. Con sesiones de juego más cortas, el menor tiempo de respuesta hace que las sesiones de juego se ejecuten en tipos de instancias viables basándose en los datos históricos más recientes. Las sesiones de juego más largas siguen ejecutándose en tipos de instancias sin evaluar los datos de viabilidad recientes, lo que aumenta el riesgo de interrupción a lo largo del tiempo.

- Evalúe la disponibilidad de los tipos de instancia: no todas las ubicaciones de la flota ofrecen todos los tipos de instancias como spot. Al elegir un tipo de instancia para su flota de spot, utilice la herramienta de creación de flotas de la consola de Amazon GameLift Servers para encontrar tipos de instancias de spot en las ubicaciones que necesite. Con esta herramienta, puede seleccionar ubicaciones de flota y, a continuación, ver la disponibilidad de los tipos de instancias en esas ubicaciones.
- Cree flotas de Spot con múltiples ubicaciones: puede crear flotas de spot con múltiples ubicaciones. Una única flota de Spot con múltiples ubicaciones implementa instancias con el mismo tipo de instancia en varias Zonas Locales Regionales de AWS o en Zonas Locales. El algoritmo de viabilidad de spot evalúa la viabilidad en función del tipo de instancia y la ubicación. Si la ubicación de una flota de spot se considera inviable, esto no afectará a otras ubicaciones de la flota, que podrán seguir utilizándose para alojar sesiones de juego.
- Cree colas con diversidad de flotas de spot: si utiliza flotas de spot para el alojamiento del juego, tendrá que configurar una cola de ubicación de sesiones de juego. Para cada nueva solicitud de sesión de juego, la cola busca los recursos de alojamiento de juegos disponibles y selecciona la mejor opción posible. Con las flotas de spot, lo ideal es disponer de una cola capaz de buscar entre múltiples flotas que varíen tanto en cuanto a ubicaciones como a tipos de instancias, e incluir al menos una flota bajo demanda como capacidad de reserva. Una cola de múltiples flotas correctamente diseñada y que ofrezca diversas opciones de ubicación es muy resistente a interrupciones, ralentizaciones y cortes. Para obtener más información sobre cómo diseñar una cola para spot, consulte [Creación de una cola para instancias de spot](#).
- Gestione las interrupciones eficazmente: configure el servidor de juegos para minimizar el impacto en los jugadores en caso de que se produzca una interrupción de spot. Cuando AWS recupera una instancia de spot, envía la notificación de finalización a Amazon GameLift Servers todos los procesos del servidor afectados mediante la función de devolución de llamada del SDK del servidor. `onProcessTerminate()` El juego debe implementar esta devolución de llamada para finalizar la sesión de juego sin problemas. Para obtener más información, consulte [Respuesta a una notificación de cierre del proceso del servidor](#).

 Note

AWS hace todo lo posible por proporcionar la notificación antes de recuperar una instancia, pero es posible que AWS recupere la instancia puntual antes de que llegue la advertencia. También debe preparar el servidor de juegos para que gestione las interrupciones imprevistas.

- Configure el escalado automático de las flotas de reserva para mantener el servicio durante una interrupción de spot. El escalado automático con seguimiento de objetivos mantiene un búfer de capacidad y escala automáticamente en función de la demanda. Con el escalado automático, las flotas de reserva (de spot o bajo demanda) empezarán a aumentar su capacidad al empezar a recibir más solicitudes de sesiones de juego.

Para reemplazar rápidamente la capacidad perdida cuando una flota de spot deja de ser viable, un mecanismo de escalado personalizado puede utilizar las métricas de colas y flotas disponibles para iniciar un escalado rápido de las flotas de reserva. Detecte cuándo las flotas de spot dejan de ser viables con métricas como `FirstChoiceOutOfCapacity`, `FirstChoiceNotViable` y `PercentAvailableGameSessions`. Para hacer una estimación de las necesidades de capacidad de sustitución, puede analizar los datos recientes de las métricas de `PlacementsStarted`. Tras escalar las flotas de reserva para dar cabida a la demanda inmediata, puede continuar con el escalado automático normal.

- Integración con FlexMatch: si su solución utiliza un emparejador de FlexMatch, no hay requisitos especiales para las flotas de spot. Puede configurar un emparejador para que utilice una cola con las flotas de spot. Amazon GameLift Servers prioriza automáticamente las ubicaciones de las partidas en las flotas de spot y bajo demanda, incluso al ubicar las nuevas sesiones de juego y al reponer los espacios vacíos para jugadores en las sesiones de juego existentes.

Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado

Puede establecer la configuración del tiempo de ejecución de una flota de EC2 administrada para que ejecute varios procesos de servidor de juegos por instancia. De esa manera, se utilizan sus recursos de alojamiento de manera más eficiente.

Modo en que una flota administra varios procesos

Amazon GameLift Servers utiliza la configuración del tiempo de ejecución de una flota para determinar el tipo y el número de procesos que se deben ejecutar en cada instancia. Una configuración de tiempo de ejecución contiene al menos una configuración de proceso de servidor que representa un archivo ejecutable de servidor de juegos. Puede definir configuraciones de procesos de servidor adicionales para ejecutar otros tipos de procesos relacionados con el juego. Cada configuración del proceso del servidor contiene la siguiente información:

- El nombre de archivo y la ruta de acceso de un ejecutable de la compilación del juego.

- (Opcional) Los parámetros que se pasan al proceso del servidor durante el lanzamiento
- El número de procesos que se van a ejecutar al mismo tiempo.

Cuando se activa una instancia de la flota, lanza inmediatamente el conjunto de procesos del servidor definidos en la configuración del tiempo de ejecución. Con varios procesos, Amazon GameLift Servers escalona el lanzamiento de cada proceso. Los procesos del servidor tienen una vida útil limitada. A medida que finalizan, Amazon GameLift Servers lanza nuevos procesos para mantener el número y el tipo de procesos del servidor definidos en la configuración del tiempo de ejecución.

Puede cambiar la configuración del entorno de tiempo de ejecución de una flota en cualquier momento agregando, modificando o eliminando configuraciones de procesos del servidor. Cada instancia comprueba periódicamente si hay actualizaciones en la configuración del tiempo de ejecución de la flota para implementar los cambios. A continuación se muestra cómo Amazon GameLift Servers realiza cambios en la configuración del entorno de tiempo de ejecución:

1. La instancia envía una solicitud a Amazon GameLift Servers para obtener la versión más reciente de la configuración del tiempo de ejecución.
2. La instancia compara sus procesos activos con la configuración del tiempo de ejecución más reciente y, a continuación, realiza el siguiente procedimiento:
 - Si la configuración del tiempo de ejecución actualizada elimina un tipo de proceso del servidor, los procesos activos del servidor de ese tipo continúan ejecutándose hasta que terminan. La instancia no reemplaza esos procesos del servidor.
 - Si la configuración del tiempo de ejecución actualizada reduce el número de procesos simultáneos para un tipo de proceso del servidor: los procesos del servidor sobrantes de ese tipo continúan ejecutándose hasta que terminan. La instancia no reemplaza esos procesos del servidor sobrantes.
 - Si la configuración del tiempo de ejecución actualizada añade un nuevo tipo de proceso del servidor o aumenta los procesos simultáneos para un tipo existente, la instancia inicia los nuevos procesos del servidor hasta el número máximo de procesos de Amazon GameLift Servers. En este caso, la instancia lanza nuevos procesos del servidor cuando los procesos existentes finalizan.

Optimización de una flota para varios procesos

Para utilizar varios procesos en una flota, realice el siguiente procedimiento:

- [Cree una compilación](#) que contenga los archivos ejecutables del servidor de juegos que desea implementar en una flota y, a continuación, cargue la compilación en Amazon GameLift Servers. Todos los servidores de juegos de una compilación deben ejecutarse en la misma plataforma y utilizar el SDK del servidor de Amazon GameLift Servers.
- Cree una configuración de tiempo de ejecución con una o varias configuraciones del proceso del servidor y múltiples procesos simultáneos.
- Integre los clientes del juego con la versión 2016-08-04 o posterior del AWS SDK.

Para optimizar el rendimiento de la flota, le recomendamos que realice las siguientes acciones:

- Gestione los escenarios de cierre de procesos del servidor para asegurarse de que Amazon GameLift Servers pueda reciclar los procesos de manera eficiente. Por ejemplo:
 - Añadir un procedimiento de cierre al código del servidor de juegos que llame a la API del servidor `ProcessEnding()`.
 - Implemente la función de devolución de llamada `OnProcessTerminate()` en el código del servidor de juegos para tramitar correctamente las solicitudes de finalización de Amazon GameLift Servers.
- Asegúrese de que Amazon GameLift Servers cierre y vuelva a lanzar los procesos del servidor cuyo estado sea defectuoso. Informe del estado a Amazon GameLift Servers implementando la función de devolución de llamada `OnHealthCheck()` en el código del servidor de juegos. Amazon GameLift Servers cierra automáticamente los procesos del servidor que hayan devuelto un estado defectuoso en tres informes consecutivos. Si no implementa `OnHealthCheck()`, Amazon GameLift Servers presupone que un proceso de servidor está en buen estado, a menos que el proceso no responda a una comunicación.

Elección del número de procesos por instancia

Al decidir el número de procesos simultáneos que se van a ejecutar en una instancia, tenga en cuenta los siguientes aspectos:

- Amazon GameLift Servers limita cada instancia a un [número máximo de procesos simultáneos](#). La suma de todos los procesos simultáneos de las configuraciones de procesos de los servidores de una flota no puede superar esa cuota.
- Para mantener niveles de rendimiento aceptables, el tipo de instancia de Amazon EC2 puede limitar el número de procesos que pueden ejecutarse de forma simultánea. Pruebe diferentes

configuraciones del juego para encontrar el número correcto de procesos del tipo de instancia preferido.

- Amazon GameLift Servers no ejecuta más procesos simultáneos que el número total configurado. Esto significa que la transición de la configuración del tiempo de ejecución anterior a la nueva configuración podría producirse de forma gradual.

Uso el agente de Amazon GameLift Servers

El agente de Amazon GameLift Servers supervisa la ejecución de los procesos del servidor de juegos en las flotas de Amazon GameLift Servers. El agente se implementa en todos los recursos de computación de una flota, y proporciona administración de procesos automatizada, administración del alojamiento y registro del recurso de computación. Para usar el agente, debe tener la compilación del servidor de juegos integrada con el SDK 5.x (o una versión posterior) del servidor de Amazon GameLift Servers.

El agente de Amazon GameLift Servers está disponible externamente para el uso con flotas de Amazon GameLift Servers que no sean flotas de EC2 administradas. (las flotas de EC2 administradas gestionan automáticamente las tareas del agente). Puede optar por ejecutar las flotas de Amazon GameLift Servers, incluidas las flotas de Anywhere, con o sin el agente. Sin el agente, debe ofrecer una solución alternativa para completar las tareas obligatorias.

Cuando se implementa en un recurso de computación, el agente de Amazon GameLift Servers debe lanzarse antes de iniciar cualquier proceso del servidor de juegos. En el lanzamiento, el agente lleva a cabo las tareas siguientes:

- Registra el recurso de computación con una flota de Amazon GameLift Servers mediante la API [RegisterCompute](#).
- Llama a la API [GetComputeAuthToken](#) para obtener un token de autorización y lo almacena para que lo utilicen los procesos del servidor que se ejecutan en el recurso de computación.
- Establece la variable de entorno de la URL de WebSocket para el recurso de computación, y establece una conexión de WebSocket con el servicio de Amazon GameLift Servers.
- Solicita la última versión de la configuración del tiempo de ejecución de la flota desde el servicio de Amazon GameLift Servers.
- Inicia y detiene los procesos del servidor según las instrucciones de configuración del tiempo de ejecución.

El código fuente y las instrucciones de compilación del agente de Amazon GameLift Servers están disponibles en el GitHub del [agente de Amazon GameLift Servers](#).

Acerca del agente

El agente de Amazon GameLift Servers está diseñado para gestionar las siguientes tareas para las flotas:

Administración de procesos

- Inicia nuevos procesos del servidor, tal y como se define en las instrucciones de tiempo de ejecución. El agente puede usar una configuración de tiempo de ejecución personalizada que se haya implementado con el agente. También puede proporcionar una `RuntimeConfiguration` como parte de la definición de la flota. Este método tiene la ventaja de que puede modificar la configuración de tiempo de ejecución de la flota en cualquier momento. El agente solicita de forma periódica configuraciones de tiempo de ejecución actualizadas al servicio de Amazon GameLift Servers.
- Supervisa las activaciones de los procesos del servidor y finaliza los procesos cuando no se activan a tiempo.
- Envía latidos a Amazon GameLift Servers. Si el agente no envía latidos, es posible que el proceso se marque como obsoleto.
- Informa a Amazon GameLift Servers cuando finaliza un proceso del servidor. Amazon GameLift Servers utiliza esta información para supervisar la disponibilidad del servidor de juegos para la ubicación de las sesiones de juego.
- Emite eventos de flota para los procesos del servidor, como los siguientes.
 - `SERVER_PROCESS_INVALID_PATH`: los parámetros de inicio del proceso del servidor de juegos se configuraron incorrectamente.
 - `SERVER_PROCESS_TERMINATED_UNHEALTHY`: el proceso del servidor de juegos no notificó una comprobación de estado válida en el plazo de 3 minutos desde la activación y, por lo tanto, se finalizó.
 - `SERVER_PROCESS_FORCE_TERMINATED`: el proceso del servidor de juegos no se cerró correctamente en el plazo de 30 segundos desde el envío de `OnProcessTerminate()`.
 - `SERVER_PROCESS_CRASHED`: un proceso del servidor de juegos se bloqueó por alguna razón.

Administración de recursos de computación

- Recibe mensajes del servicio de Amazon GameLift Servers para apagar el recurso de computación.
- Solicita que Amazon GameLift Servers finalice el recurso de computación.

Registro

- Carga registros en un bucket de Amazon S3 en su cuenta de AWS.

Abstracción de una designación de flota de Amazon GameLift Servers con un alias

Los alias de Amazon GameLift Servers se utilizan para abstraer destinos de alojamiento. Los destinos de alojamiento le indican a Amazon GameLift Servers dónde debe buscar recursos disponibles con el fin de alojar una nueva sesión de juego para los jugadores. Los alias son útiles en las siguientes situaciones:

- Si tu juego no utiliza colas de varias flotas para ubicar las sesiones de juego, solicitará nuevas sesiones de juego especificando un identificador de Amazon GameLift Servers flota. Durante la vida útil de un juego, sustituirá la flota varias veces para actualizar la compilación del servidor, para actualizar el sistema operativo y el hardware de alojamiento o para resolver problemas de rendimiento. Use un alias para resumir el ID de flota y así cambiar fácilmente el tráfico de jugadores de una flota existente a la nueva.
- Si quiere hacer algo más que crear una nueva sesión de juego cuando un cliente de juego la solicite. Por ejemplo, es posible que desees dirigir a los jugadores que utilizan un out-of-date cliente a un sitio web de actualizaciones.

Un alias debe especificar una estrategia de enrutamiento. Hay dos tipos. Una estrategia de enrutamiento sencillo dirige el tráfico de los jugadores a un ID de flota específico, que puede actualizar para redirigir el tráfico. Una estrategia de enrutamiento de terminal devuelve un mensaje al cliente en lugar de crear una nueva sesión de juego. Puede cambiar la estrategia de enrutamiento de un alias en cualquier momento.

Si utiliza una cola para ubicar las sesiones de juego, no necesitará un alias para redirigir el tráfico cuando sustituya una flota. Con una cola, solo tiene que añadir la nueva flota y eliminar la antigua. Los jugadores no pueden ver esta acción, ya que las solicitudes de nuevas sesiones de juego se atienden automáticamente con la nueva flota. No afecta a las sesiones de juego existentes. Puede identificar los destinos de las colas mediante un ID de flota o un alias.

Temas

- [Creación de un alias de Amazon GameLift Servers](#)

Creación de un alias de Amazon GameLift Servers

En este tema, se describe cómo crear un alias de Amazon GameLift Servers para usarlo en la ubicación de sesiones de juego.

Creación de un alias

Para crear un alias, utilice la consola de Amazon GameLift Servers o la AWS Command Line Interface (AWS CLI).

Console

En la [consola de Amazon GameLift Servers](#), use el panel de navegación para abrir la página Alias.

1. Elija Crear alias.
2. Escriba un Nombre para el alias. Es recomendable incluir características significativas en el nombre del alias: será útil cuando haya que consultar una lista de alias.
3. Introduzca la Descripción del alias según sea necesario.
4. Elija una Estrategia de enrutamiento para el alias.
 - a. Si elige una estrategia de enrutamiento Sencillo, seleccione un ID de flota de la lista para asociarlo a este alias. La lista incluye todas las flotas en las Región de AWS seleccionadas actualmente. Debe crear un alias en la misma región que la flota.
 - b. Si elige una estrategia de enrutamiento de Terminal, introduzca un valor de cadena que desee que Amazon GameLift Servers devuelva al cliente de juego como respuesta a una solicitud de sesión de juego. Una solicitud con un alias de terminal genera una excepción con el mensaje incrustado.
5. (Opcional) Añada Etiquetas al recurso del alias. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Asigne etiquetas a los recursos de AWS que desee categorizar y hágalo de un modo que resulte útil; por ejemplo, por finalidad, propietario o entorno. Para cada etiqueta que desee añadir, elija Añadir etiqueta nueva.
6. Cuando esté listo para implementar la nueva flota, elija Crear.

AWS CLI

Utilice el comando [create-alias](#) para crear un alias. Amazon GameLift Servers crea el recurso de alias en su Región de AWS predeterminada actual (también puede añadir una etiqueta `--region` para especificar una Región de AWS distinta).

Como mínimo, incluya un nombre de alias y una estrategia de enrutamiento. Para una estrategia de enrutamiento sencillo, especifique el ID de una flota en la misma región que el alias. Para una estrategia de enrutamiento de terminal, proporcione una cadena de mensaje.

Administración de una solución de alojamiento de Amazon GameLift Servers (lanzamiento y pasos posteriores)

En esta sección se proporcionan directrices sobre cómo administrar a largo plazo los recursos de Amazon GameLift Servers. Lanzar correctamente el juego multijugador no es más que el principio. Administrar una solución de alojamiento en producción requiere prestar constante atención al rendimiento, la optimización de costos, las actualizaciones de seguridad y la experiencia de los jugadores. En esta sección se proporcionan directrices sobre los aspectos operativos de ejecutar Amazon GameLift Servers en producción, desde las actividades de lanzamiento iniciales hasta el mantenimiento y la optimización a largo plazo.

Tanto si está preparando un lanzamiento como administrando un juego en directo u optimizando una solución de alojamiento ya establecida, estos temas te ayudarán a mantener un alojamiento de juegos fiable y rentable que se adapte a su base de jugadores.

Administración de los recursos de alojamiento de juegos con Amazon GameLift Servers

Administre los recursos de Amazon GameLift Servers, incluidas las compilaciones, las flotas, las colas de sesiones de juego y los emparejadores. Puede utilizar la consola de Amazon GameLift Servers para analizar listas de recursos o para ver información detallada de los recursos, incluidas las métricas. También puede recuperar la información de los recursos mediante el AWS SDK o la AWS CLI para Amazon GameLift Servers.

Características de la consola de Amazon GameLift Servers

Utilice la consola de Amazon GameLift Servers para ver y administrar los recursos de alojamiento de juegos y la actividad de alojamiento en curso prácticamente en tiempo real. La consola ofrece una interfaz gráfica para la mayoría de las funciones de la API de servicio de Amazon GameLift Servers. Puede utilizar la consola para hacer lo siguiente:

- Utilizar el panel para obtener una instantánea detallada. Puede ver los números y el estado actual de todos los recursos de alojamiento de Amazon GameLift Servers y seguir los enlaces para obtener información detallada de recursos concretos.

- Administrar recursos de alojamiento concretos. Puede crear, visualizar y eliminar todos los recursos de Amazon GameLift Servers y actualizar sus propiedades mutables. También puede ver ciertos tipos de actividad de alojamiento, como los eventos y las métricas de rendimiento.
- Interactuar con la actividad de las sesiones de juego y de jugador. Puede hacer un seguimiento de la actividad de las sesiones de juego y de las sesiones de jugador por flota, y utilizar esta información para solucionar problemas relacionados con las sesiones de juego. Consulte los detalles de una sesión de juego, consulte las sesiones de jugador de cada sesión de juego, o consulte la actividad de los jugadores en diversas sesiones de juego. También puede cerrar sesiones de juego concretas según sea necesario.

Temas

- [Panel de alojamiento de la consola de Amazon GameLift Servers](#)
- [Compilaciones de servidores de juegos](#)
- [Flotas en la consola de Amazon GameLift Servers](#)
- [Alias en la consola de Amazon GameLift Servers](#)
- [Colas de sesiones de juego en la consola de Amazon GameLift Servers](#)

Panel de alojamiento de la consola de Amazon GameLift Servers

Usa el panel de control de la Amazon GameLift Servers consola para obtener una instantánea de alto nivel sobre el estado actual de los recursos de Amazon GameLift Servers alojamiento de tu AWS cuenta. El panel de Amazon GameLift Servers ofrece una vista de los elementos siguientes:

- El número de compilaciones con el estado Listo, Iniciado y Error. Elija Ver compilaciones para obtener más información sobre las compilaciones de su región actual.
- El número de flotas en todos los estados. Elija Ver compilaciones para obtener información sobre las flotas de su región actual.
- Sus recursos actuales.
- Anuncios de nuevas características y servicios.

Para abrir el panel de Amazon GameLift Servers

- En la [consola de Amazon GameLift Servers](#), en el panel de navegación, elija Panel.

En el panel, podrá realizar los siguientes procedimientos:

- Para preparar el juego para el lanzamiento, seleccione Prepararse para el lanzamiento y complete el cuestionario de lanzamiento correspondiente.
- Para solicitar aumentos de cuota de servicio como preparación para los lanzamientos o como respuesta a los lanzamientos, elija Ver service quotas.
- Para ver las publicaciones del blog y la información detallada sobre las nuevas características, elija el enlace que aparece en Características destacadas.



GameLift > Dashboard

Dashboard

Build status overview

Viewing data for all builds in N. Virginia region.

 Ready
1

 Initialized
0

 Failed
0

[View builds](#)

Fleet status overview

Viewing data for all fleets in N. Virginia region.

 Active
0

 New
0

 Deleting
0

 Error
0

 In progress
0

 Terminated
0

[View fleets](#)

Resources (1)

[View service quotas](#)

Resource type	Count
Builds	1
Scripts	0
Fleets	0
Aliases	0
Queues	0
Matchmaking rule sets	0
Matchmaking configurations	0

Prepare for your game launch

[Learn more](#)

Fill out a launch questionnaire

Fill out our game launch questionnaire and email it to the GameLift launch team to ensure a smooth launch. The GameLift launch team will verify your GameLift setup and service limits, preparing you for launch.

[Prepare to launch](#)

Features spotlight

Updates on features available in N. Virginia region

March 22, 2022

Updates to Amazon GameLift FlexMatch for greater flexibility

October 28, 2021

New Asia Pacific (Osaka) region and Graviton2 support for Amazon GameLift

Compilaciones de servidores de juegos

El recurso de compilación constituye el software del servidor de juegos. Debe cargar su paquete de compilación Amazon GameLift Servers para desplegarlo en las EC2 flotas gestionadas.

Consulta información sobre las compilaciones de servidores de juegos en la Amazon GameLift Servers consola o en las que se utiliza el AWS SDK. Amazon GameLift Servers

Console

En la página de compilaciones de la [Amazon GameLift Servers consola](#), puedes ver información sobre todas las compilaciones de servidores de juegos que has subido y gestionarlas Amazon GameLift Servers para desplegarlas en EC2 las flotas gestionadas. En el panel de navegación, selecciona Hosting, Managed EC2, Builds.

La página Compilaciones muestra la siguiente información para cada compilación: Puede ajustar el contenido de la tabla según sea necesario mediante la herramienta Preferencias (consulte el icono



en la esquina superior derecha de la tabla). Las preferencias personalizadas se guardan en su cuenta de usuario de AWS y se aplican automáticamente cada vez que visita esta página.

Note

La página de compilaciones muestra solo las compilaciones de tu AWS región actual.

- Nombre: nombre asociado a la compilación cargada.
- Estado: estado de la compilación. Muestra uno de los tres mensajes de estado:
 - Iniciado: la carga no se ha iniciado o aún está en curso.
 - Preparado: la compilación está lista para la creación de la flota.
 - Error: la compilación ha agotado el tiempo de espera antes de que Amazon GameLift Servers recibiera los archivos binarios.
- Hora de creación: fecha y hora en las que cargó la compilación en Amazon GameLift Servers.
- ID de compilación: ID único asignado al cargar la compilación.
- Versión: etiqueta de versión asociada a la compilación cargada.

- Sistema operativo: SO en el que se ejecuta la compilación. El SO de la compilación determina el sistema operativo que instala Amazon GameLift Servers en las instancias de una flota.
- Tamaño: tamaño, en megabytes (MB), del archivo de compilación cargado en Amazon GameLift Servers.
- Flotas: cantidad de flotas implementadas con la compilación.

En esta página puede hacer lo siguiente:

- Ver los detalles de las compilaciones. Elija el nombre de una compilación para abrir la página de detalles de la compilación.
- Crear una flota nueva a partir de una compilación. Seleccione una compilación y, a continuación, elija Crear flota.
- Filtrar y ordenar la lista de compilación. Utilice los controles en la parte superior de la tabla.
- Eliminar una compilación. Seleccione una compilación y, a continuación, elija Eliminar.

Detalles de la compilación

En la página Compilaciones, elija el nombre de una compilación para abrir la página de detalles sobre ella. La sección Información general de la página de detalles muestra la misma información resumida de la compilación que aparece en la página Compilaciones. La sección Flotas muestra una lista de las flotas que se están ejecutando con la compilación, incluida la misma información resumida que aparece en la [página Flotas](#).

AWS SDK

Utilice los siguientes comandos AWS CLI para recuperar información sobre este recurso:

- [ListBuild](#)
- [DescribeBuild](#)

Actualización de una compilación de servidor de juegos

Al implementar la compilación del servidor de juegos para el alojamiento de EC2 administrado por Amazon GameLift Servers, debe cargar el software de servidor de juegos y crear un recurso de compilación de Amazon GameLift Servers. Después de crear una Amazon GameLift Servers compilación, puedes actualizar los metadatos de la compilación, pero no puedes actualizar los

archivos de compilación en sí. Para implementar actualizaciones en la compilación de tu servidor de juegos, debes cargar un nuevo conjunto de archivos y crear una nueva Amazon GameLift Servers compilación mediante el AWS CLI [upload-build](#) comando. Si lo prefiere, puede usar el comando [create-build](#) para cargar una nueva compilación desde un bucket de Amazon S3 que controle. A continuación, implemente la nueva compilación creando una nueva flota para ella.

Puede actualizar los metadatos de una compilación, lo que incluye el nombre y la descripción. Para estas tareas, usa la Amazon GameLift Servers consola o el [update-build](#) AWS CLI comando.

Automatización de las actualizaciones en una compilación de juegos

Siga estos consejos para automatizar y agilizar el proceso de actualización de las compilaciones de servidores de juegos para las flotas administradas de Amazon GameLift Servers:

- Utilice colas de sesiones de juego e intercambie las flotas según sea necesario. Cuando envíe solicitudes de sesiones de juego a Amazon GameLift Servers, especifique una cola de sesiones de juego en lugar de una flota concreta. Con las colas, puede añadir nuevas flotas con una nueva compilación y eliminar las flotas antiguas según sea necesario. Para obtener más información, consulte [Configuración de la ubicación de las sesiones de juego](#).
- Utilice alias para transferir jugadores a una compilación del juego nueva. Cuando envíe solicitudes de sesiones de juego a Amazon GameLift Servers, especifique un alias de flota en lugar de un ID de flota. Para obtener más información, consulte [Creación de un alias de Amazon GameLift Servers](#).
- Prepárese para el desarrollo iterativo. Durante el desarrollo del juego, explore las opciones para configurar un entorno de pruebas alojado que permita desarrollar de forma rápida e iterativa. Consulte [Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere](#).

Pruebe estos recursos del [kit de herramientas de Amazon GameLift Servers](#) disponible en GitHub:

Fast Build Update Tool (solo para el desarrollo)

Esta herramienta le ayuda a modificar las compilaciones de servidores de juegos que ya están implementadas en los recursos de computación de una flota de EC2 administrada, lo que le permite ahorrar tiempo durante la iteración rápida del desarrollo. La herramienta tiene varias opciones: puede reemplazar una compilación de juego completa o cambiar archivos específicos, y puede administrar cómo reiniciar los procesos del servidor de juegos tras las actualizaciones; también puede usarla para actualizar todos los recursos de computación en una flota o en recursos de computación de destino individuales.

Visite el repositorio del kit de herramientas de Amazon GameLift Servers en GitHub para obtener la [herramienta de actualización rápida de compilaciones](#) e información sobre su uso.

Ejemplo de script de implementación de producción

Este script muestra cómo se puede automatizar el proceso de actualización de las compilaciones de servidores de juegos implementadas en flotas de EC2 administradas en producción. Para usar este script, su solución de Amazon GameLift Servers alojamiento debe usar alias para abstraer la flota IDs. El script de ejemplo lleva a cabo los siguientes pasos básicos: cargar una compilación actualizada, crear una nueva compilación e implementarla en una nueva flota, redirigir el tráfico de jugadores desde una flota existente a la nueva flota y eliminar la flota anterior. Personalice el script de ejemplo para adaptarlo a sus propios requisitos de implementación.

Visite el repositorio del kit de herramientas de Amazon GameLift Servers en GitHub para obtener el [ejemplo de script de implementación para producción](#) e información sobre su uso.

Flotas en la consola de Amazon GameLift Servers

El recurso de flota representa un conjunto de máquinas de computación que se implementan con los servidores de juegos y los ejecutan. En función de la solución de alojamiento, es posible que haya gestionado EC2 flotas, gestionado flotas de contenedores, flotas de Anywhere o una combinación de ambas.

Consulta la información sobre las flotas en la Amazon GameLift Servers consola o en las que usas el SDK. AWS Amazon GameLift Servers

Console

Puedes ver información sobre todas las flotas creadas para alojar tus juegos en tu Amazon GameLift Servers AWS cuenta. En el panel de navegación de la izquierda de la consola, busca la opción de alojamiento que desees (Anywhere, Managed EC2, Managed containers) y selecciona Flotas.

Para cada opción de alojamiento, la página Flotas muestra las flotas que se encuentran en la región actualmente seleccionada. AWS En la página Flotas, puede crear una flota nueva o ver información detallada adicional sobre una flota. La [página de detalles](#) de una flota contiene información de uso, métricas, datos de sesión de juego y datos de sesión de jugador. También puede editar el registro de una flota o eliminarla.

De forma predeterminada, la página Flotas muestra la siguiente información resumida. Puede ajustar el contenido de la tabla según sea necesario mediante la herramienta Preferencias (consulte el icono



en la esquina superior derecha de la tabla). Las preferencias personalizadas se guardan en su cuenta de usuario de AWS y se aplican automáticamente cada vez que visita esta página.

- ID: identificador único asignado a la flota. Este ID es único en el Región de AWS lugar donde se creó la flota.
- Nombre: nombre fácil de recordar de la flota.
- Estado: estado actual de la flota, que puede ser uno de los siguientes: Nuevo, Descargando, Creando y Activo.
- Hora de creación: la fecha y la hora en que se creó la flota.

 Note

Una flota muestra un icono de aviso para las flotas que se crearon hace más de 90 días. Como práctica recomendada, te recomendamos sustituir las flotas cada 30 días para mantener un entorno seguro y de up-to-date ejecución para tus servidores de juegos alojados. Para obtener instrucciones, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

- Tipo de flota: disponibilidad de las instancias utilizadas para alojar los juegos, y que podría afectar a los costos de alojamiento. Una flota administrada puede utilizar instancias bajo demanda (siempre disponibles) o de spot (disponibilidad variable).
- Instancias activas: la cantidad de EC2 instancias que se utilizan en la flota.
- Instancias deseadas: la cantidad de EC2 instancias que se deben mantener activas.
- Sesiones de juego: número de sesiones de juego activas que se ejecutan en la flota. Los datos tienen una demora de cinco minutos.
- Sesiones de jugadores: número de sesiones de jugadores activas en la flota. Los datos tienen una demora de cinco minutos.

AWS SDK

Utilice los siguientes comandos AWS CLI para recuperar información sobre este recurso:

- EC2 Flotas gestionadas
 - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)
 - [DescribeFleetCapacity](#)
 - [DescribeFleetUtilization](#)
- Flotas de contenedores administradas
 - [ListContainerFleets](#)
 - [DescribeContainerFleet](#)
 - [ListContainerGroupDefinitions](#)
 - [DescribeContainerGroupDefinition](#)
- - [ListFleets](#)
 - [ListCompute](#)
 - [DescribeFleetAttributes](#)
 - [DescribeCompute](#)

Actualización de la configuración de una flota de Amazon GameLift Servers

Utilice la Amazon GameLift Servers consola o la AWS CLI para actualizar la configuración de la flota, cambiar las ubicaciones remotas o eliminar una flota. En el caso de las flotas administradas, no se puede cambiar la compilación del servidor de juegos ni el tipo de instancia. En lugar de eso, es necesario reemplazar la flota.

Fast Build Update Tool (solo para desarrollo)

Con EC2 las flotas gestionadas, para implementar una actualización de compilación de un servidor de juegos, debes cargar cada nueva versión Amazon GameLift Servers y crear una nueva flota para ella.

Con Fast Build Update Tool, puede omitir estos pasos durante el desarrollo, lo que le permite ahorrar tiempo y agilizar la iteración del desarrollo. Con esta herramienta, puede actualizar rápidamente los archivos de compilación de juegos en todos los recursos de computación de una flota existente. La herramienta tiene varias opciones: puede reemplazar una compilación de juego completa o cambiar 6 archivos específicos, y puede administrar cómo reiniciar

los procesos del servidor de juegos tras las actualizaciones; también puede usarla para actualizar los recursos de computación de una flota.

Para conseguir Fast Build Update Tool y obtener más información sobre su uso, visite el repositorio del kit de herramientas de Amazon GameLift Servers para [Fast Build Update Tool](#) en GitHub.

Puede actualizar los atributos mutables de la flota, la configuración de los puertos y las configuraciones de tiempo de ejecución mediante la Amazon GameLift Servers consola o la AWS CLI. Para cambiar los límites de escalado, consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).

Console

1. En el panel de navegación de la [consola de Amazon GameLift Servers](#), elija Flotas.
2. Elija la flota que desee actualizar. Una flota debe tener el estado ACTIVE para poder editarla.
3. En la página de detalles de la flota, en cualquiera de las siguientes secciones, seleccione Editar.
 - Configuración de la flota
 - Modifique los atributos de la flota; por ejemplo, Name (Nombre) y Description (Descripción).
 - Añade o elimina grupos de métricas, que Amazon CloudWatch utiliza para realizar un seguimiento de Amazon GameLift Servers las métricas agregadas de varias flotas.
 - Actualice la configuración de los límites de creación de recursos.
 - Active o desactive la protección de sesiones de juego.
 - Configuración del tiempo de ejecución: puede cambiar cualquiera de los siguientes ajustes de sus configuraciones de tiempo de ejecución y añadir o eliminar configuraciones de tiempo de ejecución.
 - Cambie la ruta de lanzamiento del servidor de juegos.
 - Añada, elimine o cambie los parámetros de lanzamiento opcionales.
 - Cambie el número de procesos simultáneos que ejecuten los servidores de juegos.
 - Activación de la sesión de juego: cambie la forma en que desea que se ejecuten y alojen los procesos del servidor mediante la actualización de los campos Número máximo

de activaciones de sesiones de juego simultáneas y Tiempo de espera de la nueva activación.

- EC2 Configuración de puertos: actualiza las direcciones IP y los rangos de puertos que permiten el acceso entrante a la flota.

4. Para guardar los cambios, elija Confirmar.

AWS CLI

Utilice los siguientes comandos AWS CLI para actualizar una flota:

- [update-fleet-attributes](#)
- [update-fleet-port-settings](#)
- [update-runtime-configuration](#)

Actualización de ubicaciones de la flota

Puede añadir o eliminar las ubicaciones remotas de una flota mediante la consola de Amazon GameLift Servers o la CLI de AWS . No se puede cambiar la región de origen de una flota.

Console

1. En el panel de navegación de la [consola de Amazon GameLift Servers](#), elija Flotas.
2. Elija la flota que desee actualizar. Una flota debe tener el estado ACTIVE para poder editarla.
3. En la página Detalles de la flota, elija la pestaña Ubicaciones para ver las ubicaciones de la flota.
4. Para añadir nuevas ubicaciones remotas, elija Añadir y seleccione las ubicaciones en las que desee implementar las instancias. Esta lista no incluye las instancias en las que el tipo de instancia de la flota no esté disponible.
5. Con las nuevas ubicaciones seleccionadas, elija Añadir. Amazon GameLift Servers añade las nuevas ubicaciones a la lista con el estado establecido en NEW. Amazon GameLift Servers comienza entonces a realizar el aprovisionamiento de una instancia en cada ubicación añadida y a prepararla para alojar sesiones de juego.
6. Para eliminar las ubicaciones remotas existentes de la flota, utilice las casillas de verificación para seleccionar una o más ubicaciones de la lista.

7. Con una o más flotas seleccionadas, elija Eliminar. Las ubicaciones eliminadas permanecen en la lista con el estado establecido en DELETING. Amazon GameLift Servers comienza entonces el proceso de finalizar la actividad en la ubicación eliminada. Si hay instancias activas que alojan sesiones de juego, Amazon GameLift Servers utiliza el proceso de finalización del servidor de juegos para finalizar correctamente las sesiones de juego y cerrar los servidores de juegos y las instancias.

AWS CLI

Utilice los siguientes comandos de la AWS CLI para actualizar las ubicaciones de la flota:

- [create-fleet-locations](#)
- [delete-fleet-locations](#)

Eliminación de una flota

Puede eliminar una flota cuando ya no la necesite. Con la eliminación de una flota de forma permanente se borran todos los datos asociados con sesiones de juego y sesiones de jugador, así como los datos de las métricas recopilados. Como opción, puede retener la flota, deshabilitar el escalado automático y escalar manualmente la flota a 0 instancias.

Note

Si la flota tiene una conexión de emparejamiento de VPC, primero solicite la autorización llamando. [CreateVpcPeeringAuthorization](#) Amazon GameLift Servers elimina la conexión de emparejamiento de VPC durante la eliminación de la flota.

Puede utilizar la Amazon GameLift Servers consola o la herramienta AWS CLI para eliminar una flota.

Console

1. En el panel de navegación de la [consola de Amazon GameLift Servers](#), elija Flotas.
2. Elija la flota que desee eliminar. Solo podrá eliminar las flotas que se encuentren en el estado ACTIVE o ERROR.
3. Elija Eliminar.

4. En el cuadro de diálogo Eliminar flota, introduzca **delete** para confirmar la eliminación.
5. Elija Eliminar.

AWS CLI

Utilice el siguiente comando AWS CLI para eliminar una flota:

- [delete-fleet](#)

Detalles de la flota en la consola de Amazon GameLift Servers

Elija el nombre de la flota para acceder a la página de detalles Flota desde el panel o desde la página Flotas.

En la página Detalles de la flota puede llevar a cabo las siguientes acciones:

- Actualizar los atributos, los ajustes de los puertos y la configuración del tiempo de ejecución de una flota.
- Añadir o eliminar ubicaciones de flota.
- Cambiar la configuración de la capacidad de la flota.
- Establecer o cambiar el escalado automático del seguimiento de objetivos.
- Eliminar una flota.

Details

Configuración de la flota

- ID de la flota: identificador único asignado a la flota. Este identificador es único en el Región de AWS lugar donde se creó la flota.
- Nombre de la flota: nombre fácil de recordar de la flota.
- ARN: identificador único asignado a esta flota. El ARN de una flota la identifica como un recurso de Amazon GameLift Servers y especifica la región y la cuenta de AWS .
- Descripción: breve descripción identificable de la flota.
- Estado: estado actual de la flota, que puede ser Nuevo, Descargando, Creando y Activo.
- Hora de creación: la fecha y la hora en que se creó la flota.

 Note

Una flota muestra un icono de aviso para las flotas que se crearon hace más de 90 días. Como práctica recomendada, te recomendamos sustituir las flotas cada 30 días para mantener un entorno seguro y de up-to-date ejecución para tus servidores de juegos alojados. Para obtener instrucciones, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

- Tipo de computación: el tipo de computación utilizado para alojar sus juegos. Una flota puede ser una flota gestionada EC2, de contenedores gestionados o de cualquier lugar.
- Sistema operativo: imagen de máquina de Amazon (AMI) que se implementa en todas las instancias de la flota. La versión de la AMI es la última versión disponible en el momento en que se crea la flota.
- Tipo de certificado: el estado de la capacidad de la flota para utilizar un certificado TLS para autenticar un servidor de juegos y cifrar todas las comunicaciones. client/server
- Tipo de flota: disponibilidad de las instancias utilizadas para alojar los juegos, y que podría afectar a los costos de alojamiento. Una flota puede utilizar instancias bajo demanda (siempre disponibles) o de spot (disponibilidad variable).
- EC2 tipo de instancia: [tipo de EC2 instancia](#) de Amazon seleccionado para la flota cuando se creó.
- Rol de instancia ARN: rol de AWS IAM que administra el acceso a sus otros AWS recursos, si se proporcionó uno durante la creación de la flota.
- Proveedor de credenciales de rol de instancia: rol de AWS IAM que administra el acceso a sus otros AWS recursos, si se proporcionó uno durante la creación de la flota.
- Nombre del grupo de métricas: grupo utilizado para agregar las métricas de varias flotas.
- Política de protección de escalado de juegos: indica si la protección de las sesiones de juego de la flota está habilitada, lo que evita que las sesiones de juego activas finalicen prematuramente durante un evento de reducción vertical.
- Número máximo de sesiones de juego por jugador: número máximo de sesiones que un jugador puede crear durante el periodo de política.
- Periodo de política: periodo de tiempo utilizado para limitar el número de sesiones que ha creado un jugador.

Detalles de la compilación

La sección Detalles de la compilación muestra la compilación alojada en la flota. Seleccione el nombre de compilación para ver la página de detalles de la compilación completa.

Configuración del tiempo de ejecución

La sección Configuración del tiempo de ejecución muestra los procesos del servidor que se van a lanzar en cada instancia. Incluye la ruta del ejecutable del servidor de juegos y los parámetros de lanzamiento opcionales.

Activación de la sesión de juego

La sección Activación de la sesión de juego muestra el número de procesos del servidor que se inician al mismo tiempo y cuánto tiempo hay que esperar a que el proceso se active antes de finalizarlo.

EC2 configuración de puerto

La sección Puertos muestra los permisos de conexión de la flota, incluida la dirección IP y los rangos de configuración del puerto.

Métricas

La pestaña Métricas muestra una representación gráfica de las métricas de la flota a lo largo del tiempo. Para obtener más información acerca del uso de las métricas de Amazon GameLift Servers, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

Events (Eventos)

La pestaña Events proporciona un registro de todos los eventos que se han producido en la flota, incluido el código del evento, el mensaje y la marca temporal. Consulte las descripciones de [eventos](#) en la Referencia de la API de Amazon GameLift Servers.

Escalado

La pestaña Escalado contiene información sobre la capacidad de la flota, incluido el estado actual y los cambios de capacidad a lo largo del tiempo. También proporciona herramientas para actualizar los límites de capacidad y administrar el escalado automático.

Escalado de capacidad

Consulte la configuración de la capacidad de la flota actual para cada ubicación de la flota. Para obtener más información sobre el cambio de límites y de capacidad, consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).

- **AWS Ubicación:** nombre de la ubicación en la que se implementan las instancias de flota.
- **Estado:** estado del alojamiento de la ubicación de la flota. El estado de la ubicación debe ser **ACTIVE** para poder alojar juegos.
- **Tamaño mínimo:** número mínimo de instancias que se deben implementar en la ubicación.
- **Instancias deseadas:** cantidad de destino de instancias activas para mantener la ubicación. Cuando las instancias activas y deseadas no son las mismas, se inicia un evento de escalado para iniciar o cerrar las instancias según sea necesario hasta que las instancias activas equivalgan a las instancias deseadas.
- **Tamaño máximo:** el mayor número de instancias que se puede implementar en la ubicación.
- **Disponible:** el límite de servicio de las instancias menos la cantidad de instancias en uso. Este valor indica el número máximo de instancias que puede añadir a la ubicación.

Políticas de escalado automático

En esta sección se incluye información sobre las políticas de escalado automático que se aplican a la flota. Puede configurar o actualizar una política basada en objetivos. Aquí se muestran las políticas basadas en reglas de la flota, que deben definirse mediante el AWS SDK o la CLI. Para obtener más información sobre el escalado, consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).

Historial de escalado

Consulte los gráficos de los cambios de capacidad a lo largo del tiempo.

Ubicaciones

La pestaña Ubicaciones muestra todas las ubicaciones en las que se implementan las instancias de flota. Las ubicaciones incluyen la región de origen de la flota y cualquier ubicación remota que se haya añadido. Puede añadir o eliminar ubicaciones directamente en esta pestaña.

- **Ubicación:** nombre de la ubicación en la que se implementan las instancias de flota.
- **Estado:** estado del alojamiento de la ubicación de la flota. Con la opción Estado de la ubicación, se realiza un seguimiento del proceso de activación de las primeras instancias de la ubicación. Además, la opción debe tener el estado **ACTIVE** para poder alojar juegos.
- **Instancias activas:** cantidad de instancias con procesos de servidor que se ejecutan en la ubicación de la flota.

- **Servidores activos:** número de procesos del servidor de juegos que pueden alojar sesiones de juego en la ubicación de la flota.
- **Sesiones de juego:** número de sesiones de juego activas en instancias en la ubicación de la flota.
- **Sesiones de jugador:** número de sesiones de jugador, que representan a jugadores individuales, que participan en las sesiones de juego activas en la ubicación de la flota.

Sesiones de juego

La pestaña **Game sessions** (Sesiones de juego) enumera las sesiones de juego pasadas y presentes alojadas en la flota, incluida información detallada. Elija un ID de sesión de juego para acceder a la información adicional de la sesión de juego, incluidas las sesiones de jugador. Para obtener más información sobre las sesiones de jugador, consulte [Sesiones de juego y de jugador en la consola de Amazon GameLift Servers](#).

Sesiones de juego y de jugador en la consola de Amazon GameLift Servers

Puede usar la consola de Amazon GameLift Servers para trabajar con las sesiones de juego y las sesiones de jugador. Para obtener más información acerca de las sesiones de juego y de jugador, consulte [Amazon GameLift Servers y la experiencia del jugador](#). La consola de Amazon GameLift Servers proporciona información y herramientas para ayudarle a investigar los problemas relacionados con las sesiones de juego.

Qué puede hacer:

- Explorar la actividad de las sesiones de juego y sesiones de jugador alojadas en una flota específica.
- Consultar la actividad de la sesión de juego de un jugador específico en varias flotas.
- Cerrar una sesión de juego específica.

Visualización de los detalles de las sesiones de juego

Los datos de las sesiones de juego y de jugador están organizados por la flota que aloja la sesión de juego.

Para acceder a la información de sesiones de juego y sesiones de jugador:

1. En la [consola de Amazon GameLift Servers](#), abra el panel de navegación izquierdo. Seleccione un tipo de solución de alojamiento y abra la página **Flotas**. Por ejemplo:

- Alojamiento, Anywhere, Flotas
 - Alojamiento, EC2 administrado, Flotas
 - Alojamiento, Contenedores administrados, Flotas
2. En cada página de flotas se muestra la lista de las flotas que ha seleccionado actualmente. Región de AWS Elija la flota cuyos datos de sesiones de juego desea ver.
 3. En la página de detalles de la flota, abra la pestaña Sesiones de juego. Esta pestaña genera una lista de todas las sesiones de juego alojadas en la flota, junto con información resumida. Puede ajustar el contenido de la tabla según sea necesario mediante la herramienta Preferencias (consulte el icono  en la esquina superior derecha de la tabla). Las preferencias personalizadas se guardan en su cuenta de usuario de AWS y se aplican automáticamente cada vez que visita esta página.
 4. Elija una sesión de juego de la lista para ver información adicional.
 5. Si la sesión de juego incluye datos de sesión de jugador, seleccione Ver sesiones de jugador para abrir la herramienta de búsqueda de sesiones de jugador con el ID de la sesión de juego rellenado automáticamente.

Los detalles de sesiones de juego incluyen la siguiente información.

- Estado: estado de la sesión de juego.
 - Activando: la instancia inicia una sesión de juego.
 - Activo: se está ejecutando una sesión de juego y está disponible para recibir jugadores, en función de la [política de creación de jugadores](#) de la sesión.
 - Terminado: ha finalizado la sesión de juego.
- ARN: nombre de recurso de Amazon (ARN) del grupo de usuario.
- Nombre: nombre generado para la sesión de juego.
- Ubicación: ubicación en la que Amazon GameLift Servers alojó la sesión de juego.
- Hora de creación: fecha y hora en que Amazon GameLift Servers creó la sesión de transmisión.
- Hora de finalización: fecha y hora en que finalizó la sesión de juego.
- Nombre de DNS: nombre del host de la sesión de juego.
- Dirección IP: dirección IP específica de la sesión de juego.
- Puerto: número de puerto que se utiliza para conectar con la sesión de juego.

- ID de creador: identificador único del jugador que inició la sesión de juego.
- Política de creación de sesiones del jugador: indica si la sesión de juego acepta nuevos jugadores.
- Política de protección para el escalado de juegos: tipo de protección de sesión de juego que se establecerá en todas las instancias nuevas que Amazon GameLift Servers inicie en la flota.

Datos de juegos

Datos de propiedades del juego, con el formato de cadena, para el envío a la sesión de juego en el inicio.

Propiedades del juego

Datos de propiedades del juego, formateados en key/value pares, para enviarlos a la sesión de juego al inicio.

Datos de emparejamiento

Si la sesión de juego se ha creado con FlexMatch, los datos de emparejamiento describen información sobre la configuración y el conjunto de reglas del emparejamiento. Esto incluye los atributos de los jugadores de cada emparejamiento, así como las asignaciones de equipo. Los datos se encuentran en formato JSON. Para obtener más información sobre el emparejamiento de FlexMatch, consulte [Creación de un emparejador](#).

Búsqueda de los datos de la sesión del jugador

Si tu solución de alojamiento de juegos usa sesiones de jugadores y proporciona un jugador único IDs, puedes explorar la actividad específica de cada jugador durante las sesiones de juego pasadas o presentes en varias flotas. Abra la herramienta Búsqueda de sesiones de jugador utilizando uno de los siguientes métodos:

- En la consola de Amazon GameLift Servers, abra el panel de navegación izquierdo, seleccione Búsqueda de sesiones de jugador y seleccione el tipo de filtro que desea utilizar.
- Cuando vea los detalles de las sesiones de juego de una flota, seleccione Ver sesiones de jugador. La herramienta de búsqueda se abre con la sesión de juego con el filtro de ID de sesión de juego preseleccionado y el valor de la sesión de juego rellenado.

Al utilizar la herramienta de búsqueda, puede proporcionar la siguiente información:

- Un ID de sesión de jugador para obtener información sobre una sesión de jugador específica.

- Un ID de sesión de juego para obtener información sobre todas las sesiones de los jugadores para la sesión de juego solicitada. Los resultados representan a todos los jugadores que han reservado un espacio o se han conectado a la sesión de juego. Si lo desea, puede filtrar los resultados por el estado de la sesión del jugador.
- Un ID de jugador para obtener información sobre todas las sesiones de jugador para el jugador solicitado. Los resultados representan todas las sesiones de juego en las que participó el jugador.

 Note

La herramienta de búsqueda busca toda la actividad de las sesiones de los jugadores en las sesiones seleccionadas actualmente. Región de AWS Si tiene varias flotas en la región, los resultados incluyen la actividad de las sesiones del jugador en todas las flotas. En el caso de las flotas con múltiples ubicaciones, los resultados también incluyen la actividad de las sesiones del jugador en todas las ubicaciones remotas de la flota.

Para cada sesión de juego se recopilan los siguientes datos de sesiones de jugador:

- ID de sesión del jugador: el identificador asignado a la sesión del jugador.
- ID de jugador: identificador exclusivo para el jugador. Elija este ID para obtener información adicional del jugador.
- ID de la sesión de juego: el identificador asignado a la sesión de juego.
- ID de la flota: el identificador asignado a la flota que ha alojado la sesión de juego.
- Estado: estado de la sesión de jugador. Los posibles estados son los siguientes:
 - Reservada: la sesión de jugador está reservada, pero el jugador no se ha conectado todavía.
 - Activa: la sesión de jugador está conectada en el servidor de juegos.
 - Completada: la sesión de jugador ha finalizado y el jugador ya no está conectado.
 - Tiempo de espera agotado: el jugador no pudo conectarse.
- Hora de creación: hora en que el jugador se conectó a la sesión de juego.
- Hora de finalización: hora en que el jugador se desconectó de la sesión de juego.
- Datos de conexión: la dirección IP, el puerto y el nombre de DNS empleados por el jugador para conectarse a la sesión de juego.

- **Datos del jugador:** información sobre el jugador proporcionada durante la creación de la sesión del jugador.

Cierre de una sesión de juego

Utilice la consola de Amazon GameLift Servers para cerrar una sesión de juego específica. Esta característica le proporciona un método sencillo y rápido para localizar una sesión de juego y enviar una señal para finalizarla. Otro método de finalización requiere encontrar la instancia de flota en la que se está ejecutando la sesión de juego, acceder a la instancia de forma remota y cerrar manualmente la sesión de juego.

Puede cerrar una sesión de juego por cualquier motivo. La razón más común es resolver una sesión de juego que no se cierra de forma natural. Como resultado, el recurso de alojamiento de la sesión de juego no se puede liberar para alojar una nueva sesión de juego y la capacidad de alojamiento de la flota se reduce.

Note

Esta característica depende de ciertos ajustes de configuración de la solución de alojamiento. Esta presenta las siguientes restricciones:

- La sesión de juego debe estar alojada en una flota que ejecute una compilación de servidor de juegos con un SDK del servidor para la versión 5 (o superior) de Amazon GameLift Servers. Si los servidores de juegos están implementados con una versión anterior, tendrá que usar el acceso remoto para eliminar la sesión de juego.
- Si la sesión de juego está alojada en una flota de Anywhere, la flota debe utilizar el agente de Amazon GameLift Servers para administrar los procesos del servidor del juego.

Para finalizar una sesión de juego

1. En la [consola de Amazon GameLift Servers](#), abra el panel de navegación izquierdo. Seleccione un tipo de solución de alojamiento y abra la página Flotas. Por ejemplo:
 - Alojamiento, Anywhere, Flotas
 - Alojamiento, EC2 administrado, Flotas
 - Alojamiento, Contenedores administrados, Flotas

2. En cada página de flotas se muestra la lista de las flotas que has seleccionado actualmente. Región de AWS Elija la flota que aloja la sesión de juego que desea finalizar.
3. En la página de detalles de la flota, abra la pestaña Sesiones de juego. En la lista de sesiones de juego, seleccione la que desea finalizar y pulse el botón Finalizar.
4. En la ventana ¿Finalizar sesión de juego?, compruebe que está cerrando la sesión de juego correcta y elija un método de finalización.
 - Cierre normal de la sesión de juego: esta opción envía una señal al proceso del servidor que aloja la sesión de juego para que se cierre. Si la versión del servidor de juegos se integró correctamente para Amazon GameLift Servers, el proceso del servidor inicia la secuencia de cierre de la sesión de juego, notifica a Amazon GameLift Servers que está finalizando y se detiene. Según el diseño del juego, la secuencia de cierre puede incluir pasos para completar correctamente la sesión de juego, como guardar los datos y avisar a los jugadores activos. Este método puede requerir un pequeño retraso para completar la secuencia de cierre de la sesión de juego.
 - Cierre inmediato de la sesión de juego: esta opción envía una señal a un administrador de procesos para que cierre el proceso del servidor que aloja la sesión de juego. Esta opción evita el cierre normal de la sesión de juego. Puede finalizar la sesión de juego incluso cuando el proceso del servidor no puede responder.
5. Confirme la finalización de la sesión de juego. Puede hacer un seguimiento del progreso de cierre en la página de sesiones de juego de la consola. El estado de la sesión de juego cambiará a “Finalizando” y luego a “Finalizada”.

Temas relacionados

- También puedes cerrar las sesiones de juego mediante el AWS SDK y el AWS CLI Para obtener más detalles y ejemplos, consulta el tema de referencia de la Amazon GameLift Servers API [TerminateGameSession](#).
- Para obtener más información sobre la integración de servidores de juegos y sobre cómo responde un proceso del servidor a las señales del servicio Amazon GameLift Servers, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Actualización de una flota de contenedores administrada por Amazon GameLift Servers

Puede actualizar la mayoría de las propiedades de una flota de contenedores administrada, incluidas las definiciones de grupos de contenedores. En función de la configuración que vaya a actualizar, una actualización de la flota podría iniciar una nueva implementación de la flota. En una implementación de flota, todas las instancias de la flota se eliminan y se sustituyen por instancias con la nueva configuración. Entre los ajustes que requieren implementación se incluyen:

- Definiciones de grupos de contenedores, incluidas las actualizaciones de las imágenes de contenedor
- Rangos de puertos de conexión y permisos de entrada
- Configuración de registro

Puede monitorizar el estado de las implementaciones de flota en la [consola de Amazon GameLift Servers](#) o en la AWS Command Line Interface (AWS CLI) para crear una flota de contenedores.

Note

No es posible actualizar el entorno de tiempo de ejecución de una flota. En el momento de crear la flota, la imagen de máquina de Amazon (AMI) de la flota se establece en la última versión disponible de la AMI de Linux. Todas las imágenes de contenedor que se implementen en esta flota deben ser compatibles con esta versión. Para cambiar la AMI de la flota o actualizar a una versión más reciente, debe crear una flota nueva. Como práctica recomendada, es aconsejable reemplazar las flotas cada 30 días para mantener un entorno de tiempo de ejecución seguro y actualizado para los servidores de juegos alojados. Para obtener más orientación, consulte [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#).

Console

En la [consola de Amazon GameLift Servers](#), seleccione la Región de AWS en la que desea crear la flota. Las definiciones de grupos de contenedores deben residir en la misma región en la que se desea crear la flota.

Abra la barra de navegación izquierda de la consola y elija Contenedores administrados: flotas. En la página de flotas de contenedores administradas, seleccione una flota de la lista y elija Editar.

1. Actualice la configuración de la flota de contenedores según corresponda. Cuando haya terminado, elija Create (Crear).
2. Si las actualizaciones requieren una implementación de flota, se le pedirá que especifique las opciones de implementación de la siguiente manera:
 - Protección de la sesión de juego. Puede optar por proteger las instancias de flota que tengan sesiones de juego activas (implementación segura). Con esta configuración, las instancias de flota no se sustituyen hasta que finalizan las sesiones de juego. También puede optar por sustituir las instancias de flota independientemente de la actividad de las sesiones de juego (implementación no segura). Las implementaciones no seguras son útiles durante las fases de desarrollo y prueba, para reducir el tiempo de implementación.
 - Porcentaje de buen estado mínimo. Puede administrar la rapidez con la que se sustituyen las instancias de flota. Utilice esta configuración para mantener una cantidad mínima de tareas en buen estado durante la implementación. Un valor bajo prioriza la velocidad de implementación, mientras que un valor alto garantiza que la disponibilidad del servidor de juegos se mantenga en niveles altos durante toda la implementación.
 - Estrategia en caso de error de implementación. Decida qué acciones se deben adoptar en el caso de que se produzca un error en la implementación. Un error en la implementación significa que algunos de los contenedores actualizados no han superado las comprobaciones de estado y se consideran defectuosos. Puede configurar las implementaciones para que reviertan automáticamente todas las instancias de flota al estado de implementación anterior. Como alternativa, puede optar por mantener algunas de las instancias de flota defectuosas para utilizarlas en la depuración.

Si la solicitud se realiza correctamente, la consola muestra la pestaña Implementaciones de la flota de contenedores administrada. Utilice esta pestaña para monitorizar el estado de cada implementación. Si inicia una nueva implementación de la flota, esta acción cancela automáticamente cualquier implementación de la flota que haya en curso.

AWS CLI

Para crear una flota de contenedores con la AWS CLI, abra una ventana de línea de comandos y utilice el comando `update-container-fleet`. Para obtener más información sobre este comando, consulte [update-container-fleet](#) en la Referencia de comandos de la AWS CLI.

En el siguiente ejemplo se ilustra la actualización de una flota de contenedores existente con las siguientes características:

- Actualiza la definición del grupo de contenedores del servidor de juegos para que utilice la versión 2.
- Especifica opciones de implementación segura.

```
{
  "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"
}
```

Actualización de una definición de grupo de contenedores para una flota de contenedores de Amazon GameLift Servers

Puede actualizar la mayoría de las propiedades de una definición de grupo de contenedores, incluidas las definiciones de contenedores individuales. Las definiciones de grupos de contenedores tienen un número de versión. Al actualizar una definición de grupo de contenedores, Amazon GameLift Servers guarda la actualización e incrementa el número de versión de la definición. Al configurar una flota de contenedores, puede especificar qué versión de una definición de grupo de contenedores implementar.

Tras actualizar la definición de un grupo de contenedores, puede implementar la nueva versión en una flota de contenedores nueva o existente.

Actualización de una definición de grupo de contenedores del servidor de juegos

En este tema se describe cómo actualizar la definición del grupo de contenedores de servidores de juegos mediante la Amazon GameLift Servers consola o las herramientas AWS CLI. Para obtener información más detallada sobre las características opcionales, consulte [Personalización de una flota de contenedores de Amazon GameLift Servers](#).

Actualización de una definición de grupo de contenedores:

Console

En la [Amazon GameLift Servers consola](#), selecciona el Región de AWS lugar en el que deseas crear el grupo de contenedores.

Abra la barra de navegación izquierda de la consola y elija Contenedores administrados: definiciones de grupos. En la página Definición de grupos de contenedores, elija una definición y la versión del grupo de contenedores que desea actualizar.

Una vez que haya guardado las actualizaciones, puede usar la nueva versión para crear una nueva flota de contenedores o puede implementar las actualizaciones en una flota de contenedores existente.

Paso 1: Definición de los detalles de la definición del grupo de contenedores

- Puede actualizar la configuración de los límites de memoria total y vCPU.

Paso 2: Adición de definiciones de contenedor

Puede hacer las siguientes actualizaciones en una definición de contenedor:

- Actualizar las definiciones de contenedor existentes.
 - Añadir nuevas definiciones de contenedor de apoyo.
 - Eliminar definiciones de contenedor de apoyo.
1. Puede actualizar la URI de la imagen de ECR. Asegúrese de actualizar el ajuste de versión del SDK del servidor para que coincida con la nueva imagen.
 2. Puede actualizar el rango de puertos de contenedor interno según corresponda. Los cambios que realice en estos ajustes pueden afectar a la configuración del puerto de conexión de una

flota de contenedores cuando estos cambios se implementen en una flota. Para obtener más información, consulte [Configuración de conexiones de red](#).

Paso 3: Configuración de las dependencias

- Puede cambiar las dependencias según sea necesario. Para obtener más información, consulte [Establecimiento de dependencias de contenedores](#).

Paso 3: Revisar y crear

- Revise las actualizaciones de la definición de grupo de contenedores. Utilice Editar para hacer cambios adicionales en cualquier sección. Cuando haya terminado, elija Crear para generar una nueva versión de la definición del grupo de contenedores.

Si la solicitud se realiza correctamente, la consola muestra la página de detalles del nuevo recurso de definición de grupo de contenedores. Inicialmente el estado es COPYING, cuando Amazon GameLift Servers comienza a tomar instantáneas de todas las imágenes de contenedor del grupo. Cuando se completa esta fase, el estado de la definición del grupo de contenedores cambia a READY. Una definición de grupo de contenedores debe tener el estado READY para poder crear una flota de contenedores con ella.

AWS CLI

Cuando utilice la AWS CLI para crear o actualizar una definición de grupo de contenedores, mantenga las configuraciones de definición de contenedores en un JSON archivo independiente. Puede hacer referencia al archivo en el comando de la CLI. Consulte [Creación de un archivo JSON de definición de contenedor](#) para ver esquemas de ejemplo.

Al actualizar una definición, solo necesita especificar los valores que desea actualizar. Amazon GameLift Servers retiene los valores que no incluya en la solicitud de actualización. Si está cambiando la definición de un contenedor. Sin embargo, al cambiar la definición de un contenedor, proporcione un conjunto completo.

Actualización de una definición de grupo de contenedores

Para actualizar una nueva definición de grupo de contenedores, utilice el comando `update-container-group-definition` de la CLI. Para obtener más información sobre este comando, consulte [update-container-group-definition](#) la Referencia de comandos de AWS CLI.

Example: grupo de contenedores de servidor de juegos

Puede especificar una versión de definición de grupo de contenedores al recuperar, actualizar o eliminar una definición de grupo de contenedores, así como al crear o actualizar una flota de contenedores. Cada definición de grupo de contenedores tiene una propiedad correspondiente a la versión. Además, el valor ARN de la definición especifica el número de versión.

Este ejemplo ilustra una solicitud de cambio en la definición de un grupo de contenedores de servidor de juegos. Presupone que se ha creado un archivo JSON con las definiciones de contenedor de este grupo. En este ejemplo, se utiliza el valor ARN para el nombre de la definición y se especifica que la actualización se realiza a la versión 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
```

Clonación de una definición de grupo de contenedores

Puede usar la consola de Amazon GameLift Servers para clonar una definición de grupo de contenedores existente.

Creación de un grupo de contenedores

1. En la [consola de Amazon GameLift Servers](#), elija Grupos de contenedores en el panel de navegación izquierdo.
2. En la página de lista Grupos de contenedores, seleccione el grupo de contenedores existente que desea clonar. Tras seleccionar un grupo de contenedores, el botón Clonar estará activo.
3. Elija Clonar. Esta acción abre el asistente de creación de grupos de contenedores con los ajustes predefinidos.
4. Introduzca un nuevo nombre para el grupo de contenedores clonado. Los grupos de contenedores de una misma región deben tener nombres únicos.

5. Recorra las páginas de definición de contenedores y grupos de contenedores, revise la información y elija Crear para crear el nuevo grupo de contenedores.

Eliminación de una definición de grupo de contenedores de una flota de contenedores de Amazon GameLift Servers

Dispone de varias opciones para eliminar una definición de grupo de contenedores. Cuando se elimina una definición de grupo de contenedores, también se eliminan todas las definiciones de contenedores del grupo de contenedores.

Las definiciones de grupos de contenedores pueden tener varias versiones. Las versiones de los grupos de contenedores tienen el mismo nombre, pero un número de versión diferente. En la definición del grupo de contenedores se ARNs especifican tanto el nombre como la versión.

Puede especificar una versión de definición de grupo de contenedores al recuperar, actualizar o eliminar una definición de grupo de contenedores, así como al crear o actualizar una flota de contenedores. Cada definición de grupo de contenedores tiene una propiedad correspondiente a la versión. Además, el valor ARN de la definición especifica el número de versión.

Existen varias formas posibles de eliminar definiciones de grupos de contenedores:

- Puede eliminar todas las versiones de una definición específica.
- Puede eliminar una versión concreta de una definición específica.
- Puede conservar algunas de las versiones más recientes y eliminar las versiones más antiguas de una definición específica. Por ejemplo, puede eliminar todas las versiones anteriores a la versión 5.

Una versión de una definición de grupo de contenedores solo se puede eliminar si no se está utilizando en una flota de contenedores.

Eliminación de una definición de grupo de contenedores

Console

En la [Amazon GameLift Serversconsola](#), seleccione el Región de AWS lugar en el que desee crear el grupo de contenedores.

Abra la barra de navegación izquierda de la consola y elija Contenedores administrados: definiciones de grupos. En la página Definiciones de grupos de contenedores, seleccione la definición que desea modificar y elija Eliminar.

Se le solicitará que seleccione el tipo de eliminación que desea realizar y que especifique otros ajustes en función del tipo de eliminación.

AWS CLI

- Para eliminar una definición de grupo de contenedores, utilice el comando `delete-container-group-definition` de la CLI y proporcione valores para el tipo de eliminación que desea realizar. Para obtener más información sobre este comando, consulte la Referencia [delete-container-group-definition](#) de comandos de la AWS CLI.

En este ejemplo se ilustra una solicitud para eliminar todas las versiones de una definición de grupo de contenedores de servidor de juegos anteriores a la versión 5.

Example

```
aws gamelift delete-container-group-definition \
  --name MyAdventureGameContainerGroup \
  --version-count-to-retain 5 \
```

Alias en la consola de Amazon GameLift Servers

El recurso de alias es una indicación al destino de una flota o terminal. Se utiliza para dirigir el tráfico de jugadores a destinos de alojamiento específicos. Utilice un alias para abstraer un destino, de modo que pueda redirigir el tráfico más fácilmente cuando actualice sus flotas de alojamiento.

Vea información sobre los alias en la Amazon GameLift Servers consola o utilice el AWS SDK para Amazon GameLift Servers.

Console

La página Alias muestra información sobre los alias de Amazon GameLift Servers que dirigen el tráfico a destinos de alojamiento específicos. Para ver los alias, seleccione Alojamiento y Alias en el panel de navegación.

En la página de alias, puede realizar las siguientes acciones:

- Crear un alias nuevo. Elija Crear alias.
- Filtrar y ordenar la tabla de alias. Utilice los controles en la parte superior de la tabla. Puede ajustar el contenido de la tabla según sea necesario mediante la herramienta Preferencias (consulte el icono



en la esquina superior derecha de la tabla). Las preferencias personalizadas se guardan en su cuenta de usuario de AWS y se aplican automáticamente cada vez que visita esta página.

- Ver los detalles de los alias. Elija un nombre del alias para abrir la página de detalles del alias.
- Eliminar un alias. Elija un alias y, a continuación, elija Eliminar.

Detalles de alias

La página Detalles de alias muestra información resumida sobre el alias.

En esta página puede hacer lo siguiente:

- Editar un alias. Elija Edit (Edición de).
- Ver las flotas asociadas al alias.
- Eliminar un alias. Elija Eliminar.

La información detallada del alias incluye:

- ID: número exclusivo que se utiliza para identificar al alias.
- Descripción: descripción del alias.
- ARN: nombre de recurso de Amazon (ARN) de un alias.
- Creación: es la fecha y la hora en que se creó el alias.
- Última actualización: fecha y la hora en la que se actualizó por última vez el alias.
- Tipo de direccionamiento: opción de direccionamiento para el alias, que puede ser una de las siguientes:
 - Simple: redirige el tráfico de jugadores a un ID de la flota especificado. Puede actualizar el ID de flota de un alias en cualquier momento.
 - Terminal: transmite un mensaje al cliente. Por ejemplo, puedes dirigir a los jugadores que utilizan un out-of-date cliente a una ubicación en la que puedan obtener una mejora.
- Etiquetas: pares de clave y valor que se utilizan para identificar el alias.

AWS SDK

Utilice los siguientes comandos AWS CLI para recuperar información sobre este recurso:

- [ListAliases](#)
- [DescribeAlias](#)

Edición de un alias

Puede editar un alias utilizando la consola de Amazon GameLift Servers o el comando [update-alias](#) de la CLI de AWS.

En este tema se describe cómo editar un alias de Amazon GameLift Servers para usarlo en la ubicación de las sesiones de juego. Puede hacer los siguientes cambios:

Edición de un alias

Para editar un alias, utilice la consola de Amazon GameLift Servers o la AWS Command Line Interface (AWS CLI).

Console

En la [consola de Amazon GameLift Servers](#), use el panel de navegación para abrir la página Alias.

1. Seleccione el alias que desee editar y elija Editar. Si no ve el alias que quiere editar, compruebe cuál es la Región de AWS seleccionada actualmente.
2. En la página Editar alias, puede hacer los siguientes cambios:
 - Cambiar el nombre del alias.
 - Cambiar la descripción del alias.
 - Cambie la estrategia de enrutamiento: de sencillo a terminal o de terminal a sencillo.
 - Cambiar el ID de flota al que está asociado el alias (para los alias con una estrategia de enrutamiento sencillo).
 - Cambiar el texto del mensaje (para los alias con una estrategia de enrutamiento de terminal).
3. Seleccione Guardar cambios. Al actualizar el ID de flota para un alias con una estrategia de enrutamiento sencillo, la transición puede tardar hasta dos minutos en completarse. Durante este tiempo, es posible que se coloquen nuevas sesiones de juego en la flota antigua.

AWS CLI

Utilice el comando [update-alias](#) para hacer cambios en un recurso de alias. Puede actualizar un recurso de alias en su Región de AWS predeterminada actual, o puede añadir una etiqueta `--region` para especificar una Región de AWS distinta.

Puede modificar las siguientes propiedades:

- Nombre del alias.
- Descripción del alias.
- Tipo de estrategia de enrutamiento. Asegúrese de proporcionar un ID de flota o una cadena de mensaje para la nueva estrategia de enrutamiento.
- ID de flota para una estrategia de enrutamiento sencillo existente. El ID de flota debe estar en la misma región que el alias.
- Cadena de mensaje para una estrategia de enrutamiento de terminal existente.

Colas de sesiones de juego en la consola de Amazon GameLift Servers

El recurso de cola es su configuración del componente de ubicación de las sesiones de juego. Determina la forma en la que Amazon GameLift Servers busca y elige el mejor recurso de computación posible para alojar una nueva sesión de juego en función de las opciones de configuración que elija.

Consulta información sobre las colas de sesiones de juego en la Amazon GameLift Servers consola o sobre el uso del AWS SDK para Amazon GameLift Servers.

Console

Puede ver información de todas las colas, que se utiliza para procesar las solicitudes de nuevas sesiones de juego. La página de colas muestra las colas de sesiones de juego seleccionadas actualmente. Región de AWS En la página *Queues*, puede crear una cola nueva, eliminar las colas existentes o abrir una página de detalles de una cola seleccionada. Una página de detalles de la cola contiene la configuración y los datos de métricas de la cola. Para obtener más información sobre las colas, consulte [Configuración de la ubicación de las sesiones de juego](#).

La página *Colas* muestra la siguiente información resumida para cada cola. Puede ajustar el contenido de la tabla según sea necesario mediante la herramienta *Preferencias* (consulte el icono



en la esquina superior derecha de la tabla). Las preferencias personalizadas se guardan en su cuenta de usuario de AWS y se aplican automáticamente cada vez que visita esta página.

- **Nombre de la cola:** el nombre asignado a la cola. Las solicitudes de sesiones de juego nuevas utilizan este nombre para identificar una cola.
- **Tiempo de espera de la cola:** duración máxima, en segundos, que una solicitud de ubicación de sesión de juego se mantiene en la cola antes de expirar.
- **Destinos en cola:** número de flotas enumeradas en la configuración de colas. Amazon GameLift Servers ubica las sesiones de juego nuevas en cualquier flota en la cola.

Visualización de los detalles de la cola

Puede acceder a información detallada sobre cualquier cola, incluida la configuración y las métricas de la cola. Para abrir una página de detalles de la cola, vaya a la página [Colas](#) y elija el nombre de la cola.

La página de detalles de la cola muestra una tabla de resumen y pestañas que contienen información adicional. En esta página puede hacer lo siguiente:

- **Actualizar la configuración de la cola,** la lista de destinos y las políticas de latencia de los jugadores. Elija **Edit (Edición de)**.
- **Eliminar una cola.** Después de eliminar una cola, todas las solicitudes de sesiones de juego nuevas que hacen referencia al nombre de la cola darán error. Elija **Eliminar**.

Note

Para restaurar una cola eliminada, cree una nueva con el nombre de la cola eliminada.

Details

La sección **Información general** muestra el Nombre de recurso de Amazon (ARN) y el tiempo de espera de la cola. Puede utilizar el ARN para hacer referencia a la cola en otras acciones o áreas de Amazon GameLift Servers. El tiempo de espera es la duración máxima, en segundos, que una solicitud de ubicación de sesión de juego se mantiene en la cola antes de expirar.

Notificación de eventos

La sección Notificación de eventos muestra el tema de SNS en el que Amazon GameLift Servers publica las notificaciones de eventos y los datos de eventos que se añaden a todos los eventos creados por esa cola.

Tags

La tabla de Etiquetas muestra las claves y los valores utilizados para etiquetar el recurso. Para obtener más información sobre el etiquetado, consulte Recursos de [etiquetado. AWS](#)

Métricas

La pestaña Metrics muestra una representación gráfica de las métricas de la cola a lo largo del tiempo.

Las métricas de la cola incluyen un amplio abanico de información que describe la actividad de ubicación de toda la cola, incluidas las ubicaciones correctas organizadas por región. Puede utilizar los datos de la región para saber dónde alojar los juegos. Las métricas de ubicación regional pueden ayudar a detectar problemas con el diseño general de las colas.

Las métricas de colas también están disponibles en Amazon CloudWatch. Para ver descripciones de métricas disponibles, consulte [Métricas de Amazon GameLift Servers para colas](#).

Destinos

La pestaña Destinations muestra todas las flotas o alias correspondientes a la cola.

Cuando Amazon GameLift Servers busca los destinos para los recursos disponibles para alojar una nueva sesión de juego, la búsqueda la realiza según el orden predeterminado que se indique aquí. Siempre que haya capacidad en el primer destino mostrado, Amazon GameLift Servers ubicará en él las sesiones de juego nuevas. Para hacer que determinadas solicitudes de ubicación de sesiones de juego anulen el orden predeterminado, proporcione los datos de latencia de los jugadores. Estos datos indican a Amazon GameLift Servers que busque un destino disponible que tenga la latencia media de jugadores más baja. Para obtener más información sobre el diseño de colas, consulte [Personalización de colas de sesiones de juego](#).

Ubicación de la sesión

Ajustes de configuración que personalizan el proceso de ubicación del juego.

Políticas de latencia de jugadores

La sección Políticas de latencia de jugadores muestra todas las políticas que utiliza la cola. En las tablas aparecen las políticas en el orden en que se aplican.

Ubicaciones

La sección Ubicaciones muestra las ubicaciones en las que esta cola puede incluir una sesión de juego.

Priority (Prioridad)

La sección Prioridad muestra el orden en que la cola evalúa los detalles de una sesión de juego.

Orden de ubicación

La sección Orden de ubicación muestra el orden predeterminado que utiliza la cola al colocar las sesiones de juego. La cola usa ese orden si no ha definido otros tipos de prioridad.

AWS SDK

Utilice los siguientes comandos AWS CLI para recuperar información sobre este recurso:

- [DescribeGameSessionQueues](#)

Preparativos para el lanzamiento con alojamiento en Amazon GameLift Servers

Use las siguientes listas de comprobación para validar todas las fases de implementación del juego. Los elementos marcados como [Crítico] son fundamentales para el lanzamiento a la producción.

Descargue y complete el cuestionario de lanzamiento de Amazon GameLift Servers, que está disponible en la [consola de Amazon GameLift Servers](#). Queremos que todos los desarrolladores de juegos que utilicen Amazon GameLift Servers tengan una experiencia de lanzamiento fluida; la información solicitada nos resultará útil para ayudarle a prepararse para las pruebas de carga, el lanzamiento temporal y el lanzamiento público. Planifique el envío del cuestionario completado al menos tres (3) meses antes de realizar la primera prueba de carga.

Temas

- [Preparación del juego](#)
- [Preparativos para la prueba](#)

- [Preparación para el lanzamiento](#)
- [Planificación de actualizaciones posteriores al lanzamiento](#)

Preparación del juego

- [Imprescindible] Compruebe que ha completado todos los [pasos del plan de desarrollo](#) en la solución de alojamiento y que dispone de todos los componentes necesarios, lo que incluye un servidor de juegos integrado, un servicio de backend para los clientes del juego, flotas de alojamiento y un método de ubicación de sesiones de juego (por ejemplo, una cola).
- [Crítico] [Crea funciones AWS Identity and Access Management \(IAM\)](#) que permitan a tu servidor de juegos acceder a otros AWS recursos mientras se ejecuta.
- [Imprescindible] Diseñe e implemente la conmutación por error en otros recursos de alojamiento según sea necesario.
- [Planifique la implementación de las flotas en las ubicaciones de destino](#) teniendo en cuenta las colas y la estructura de las flotas del juego.
- [Automatice su implementación](#) utilizando la infraestructura como código (IaC) con CloudFormation y el AWS Cloud Development Kit (AWS CDK)
- [Recopile registros y análisis](#) con Amazon CloudWatch y Amazon Simple Storage Service (Amazon S3).

Preparativos para la prueba

- [Crítico] [Solicite aumentos en las cuotas de Amazon GameLift Servers servicio y otras Servicio de AWS cuotas para](#) que su entorno activo pueda adaptarse a las necesidades de producción.
- [Crítico] Verifique que los puertos abiertos de las flotas activas coincidan con el rango de puertos que podrían utilizar sus servidores.
- [Crítico] Cierre el puerto RDP 3389 y el puerto SSH 22.
- Desarrolle un plan para la DevOps gestión de tu juego. Si utiliza Amazon CloudWatch Logs o las métricas CloudWatch personalizadas de Amazon, defina alarmas para problemas graves o críticos en la flota de servidores. Simule los errores y pruebe los manuales de ejecución.
- Compruebe que los recursos de computación empleados admitan la cantidad de procesos de servidor que desea ejecutar simultáneamente en cada recurso de computación.
- [Ajuste su política de escalado](#) para que sea más conservadora al principio y proporcione más capacidad inactiva de la que cree que necesita. Puede optimizar los costos más adelante.

Considere el uso de una política de escalado basada en objetivos con una capacidad inactiva del 20 por ciento.

- En el caso de FlexMatch, utilice [reglas de latencia](#) para emparejar a los jugadores que estén geográficamente cerca entre sí. Compruebe cómo se comporta bajo carga con datos de latencia sintéticos del cliente de pruebas de carga.
- Ponga a prueba su infraestructura de autenticación de jugadores y sesiones de juego para comprobar si se escala de forma eficaz a la demanda.
- Verifique que un servidor que ha estado funcionando durante varios días siga aceptando conexiones.
- Aumente el nivel de su Soporte plan a Business o Enterprise para que AWS pueda responderle en caso de problemas o interrupciones.

Preparación para el lanzamiento

- [Crítico] [Configure la política de protección de la flota](#) para proteger por completo todas las flotas activas, de modo que la reducción vertical no interrumpa las sesiones de juego activas.
- [Crítico] [Establezca un tamaño máximo de flota](#) lo suficientemente alto como para adaptarse a los picos de demanda previstos, como mínimo. Le recomendamos que duplique el tamaño máximo para una demanda no prevista.
- Anime a todo el equipo de desarrollo a participar en el evento de lanzamiento y a supervisar el lanzamiento del juego en una sala de lanzamiento.
- Supervise la latencia y la experiencia de los jugadores.

Planificación de actualizaciones posteriores al lanzamiento

- [Ajuste la política de escalado](#) para minimizar la capacidad inactiva en función del uso del jugador.
- [Modifique las reglas de FlexMatch](#) o [añada ubicaciones de alojamiento](#) en función de los datos de latencia de los jugadores y de los requisitos revisados.
- Optimice la configuración del tiempo de ejecución para ejecutar tantas sesiones de juego como sea posible en cada recurso de computación. Maximizar la eficiencia del rendimiento de este modo puede afectar directamente a los costos de la flota, ya que quizá le permita ejecutar más procesos de servidor con los mismos recursos de computación.
- [Utilice los datos de análisis](#) para impulsar el desarrollo continuo, mejorar la experiencia de los jugadores y la longevidad del juego, y optimizar la monetización.

Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation

Puede utilizarlos CloudFormation para gestionar sus Amazon GameLift Servers recursos. En CloudFormation, crea una plantilla que modela cada recurso y, a continuación, utiliza la plantilla para crear sus recursos. Para actualizar los recursos, debe realizar los cambios en la plantilla y utilizarlos CloudFormation para implementar las actualizaciones. Puede organizar sus recursos en grupos lógicos, llamados "pilas" y "conjuntos de pilas".

El uso CloudFormation para mantener los recursos de Amazon GameLift Servers alojamiento ofrece una forma más eficiente de administrar conjuntos de AWS recursos. Puede utilizar el control de versiones para realizar un seguimiento de los cambios en las plantillas a lo largo del tiempo y coordinar las actualizaciones realizadas por varios miembros del equipo. También puede reutilizar las plantillas. Por ejemplo, al implementar un juego en varias regiones, puede usar la misma plantilla para crear recursos idénticos en cada región. También puede utilizar estas plantillas para implementar los mismos conjuntos de recursos en otra partición.

Para obtener más información al respecto CloudFormation, consulte la [Guía AWS CloudFormation del usuario](#). Para ver información de la plantilla de recursos de Amazon GameLift Servers, consulte la [Referencia de tipos de recursos de Amazon GameLift Servers](#).

Prácticas recomendadas

Para obtener instrucciones detalladas sobre su uso CloudFormation, consulte las [prácticas CloudFormation recomendadas](#) en la Guía del AWS CloudFormation usuario. Además, estas prácticas recomendadas tienen especial relevancia cuando se utiliza Amazon GameLift Servers.

- Administre de forma coherente los recursos que utiliza mediante CloudFormation. Si cambias tus recursos por fuera, CloudFormation tus recursos no estarán sincronizados con tus plantillas de recursos.
- Usa CloudFormation pilas y conjuntos de pilas para gestionar de forma eficiente varios recursos.
 - Utilice pilas para administrar grupos de recursos conectados. Por ejemplo, una pila que contiene una compilación, una flota que hace referencia a la compilación y un alias que hace referencia a la flota. Si actualizas la plantilla para reemplazar una compilación, CloudFormation reemplaza las flotas conectadas a la compilación. CloudFormation a continuación, actualiza los alias existentes para que apunten a las nuevas flotas. Para obtener más información, consulte [Trabajo con pilas](#) en la Guía del usuario de AWS CloudFormation .

- Usa conjuntos de CloudFormation pilas si vas a implementar pilas idénticas en varias regiones o cuentas. AWS Para obtener más información, consulte [Trabajo con conjuntos de pilas](#) en la Guía del usuario de AWS CloudFormation .
- Si utiliza instancias de spot, incluya una flota bajo demanda como respaldo. Es recomendable que configure las plantillas con dos flotas en cada región, una con instancias de spot y otra con instancias bajo demanda.
- Agrupe los recursos específicos de la ubicación y los recursos globales en pilas distintas cuando administre recursos en varias ubicaciones.
- Coloque sus recursos globales cerca de los servicios que los utilizan. Los recursos como las colas y las configuraciones de emparejamiento suelen recibir un gran volumen de solicitudes de orígenes específicos. Al colocar sus recursos cerca del origen de esas solicitudes, minimiza el tiempo de viaje de la solicitud y puede mejorar el rendimiento general.
- Coloque su configuración de emparejamiento en la misma región que la cola de sesiones del juego que utiliza.
- Cree un alias distinto para cada flota de la pila.

Uso de pilas CloudFormation

Recomendamos utilizar las siguientes estructuras al configurar CloudFormation pilas de recursos. Amazon GameLift Servers La estructura óptima de la pila varía en función de si va a implementar el juego en una sola región o en varias regiones.

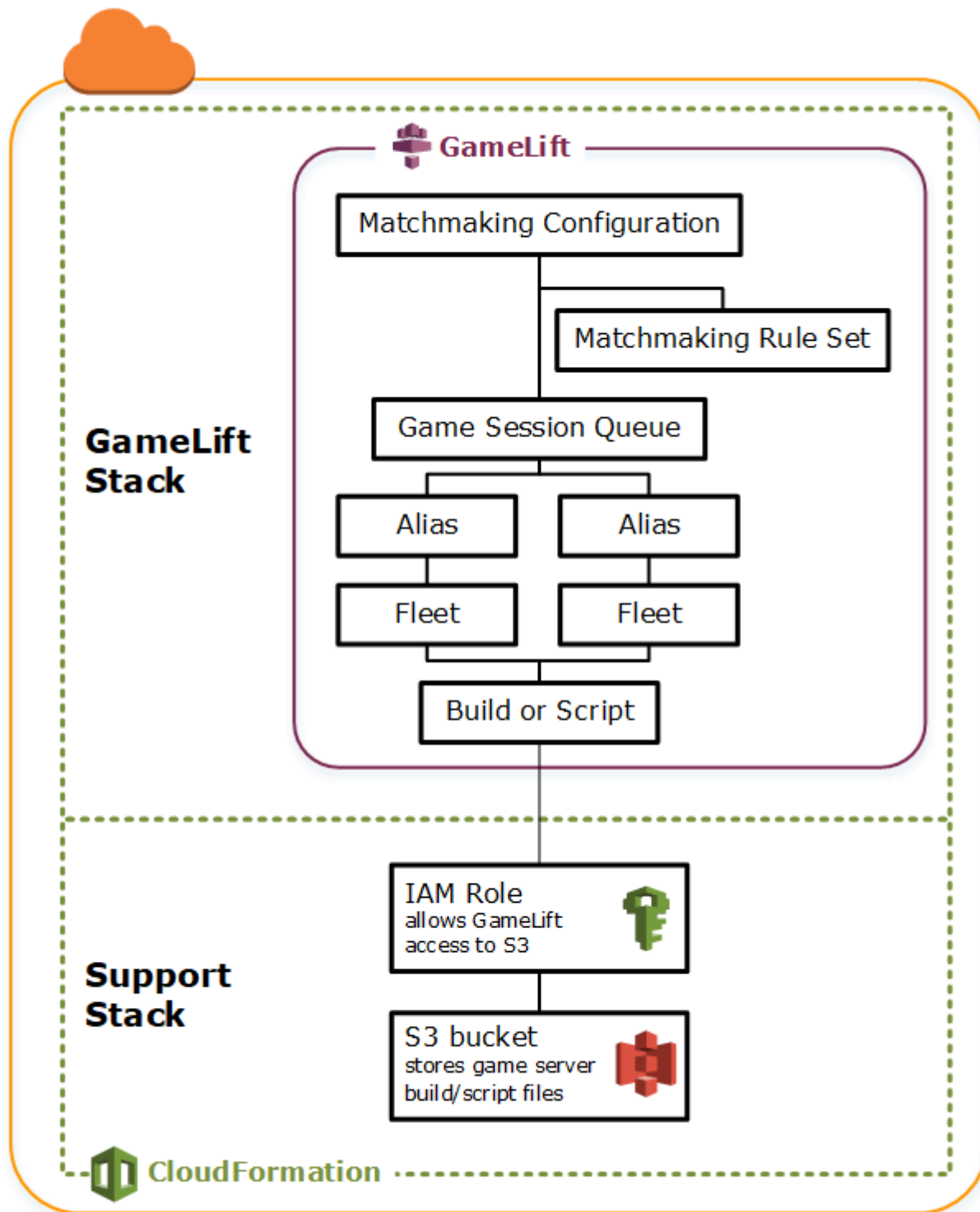
Pilas para una sola ubicación

Para administrar los recursos de Amazon GameLift Servers en una sola ubicación, recomendamos una estructura de dos pilas:

- Pila de respaldo: esta pila contiene recursos de los que dependen los recursos de Amazon GameLift Servers. Como mínimo, esta pila debe incluir el bucket de S3 donde almacena el servidor de juegos personalizado o los archivos de script de Realtime. La pila también debe incluir una función de IAM que dé Amazon GameLift Servers permiso para recuperar los archivos del bucket de S3 al crear un recurso de Amazon GameLift Servers compilación o script. Esta pila también puede contener otros AWS recursos que se utilizan con el juego, como tablas de DynamoDB, clústeres de Amazon Redshift y funciones de Lambda.
- Amazon GameLift Serverspila: esta pila contiene todos los Amazon GameLift Servers recursos, incluida la compilación o el script, un conjunto de flotas, los alias y la cola de sesiones de juego.

CloudFormation crea un recurso de compilación o script con los archivos almacenados en la ubicación del depósito de S3 y despliega la compilación o el script en uno o más recursos de la flota. Cada flota debe tener su alias correspondiente. La cola de sesiones del juego hace referencia a algunos o todos los alias de la flota. Si lo utilizas FlexMatch para el emparejamiento, esta pila también contiene una configuración de emparejamiento y un conjunto de reglas.

El siguiente diagrama ilustra una estructura de dos pilas para implementar recursos en una sola región. AWS



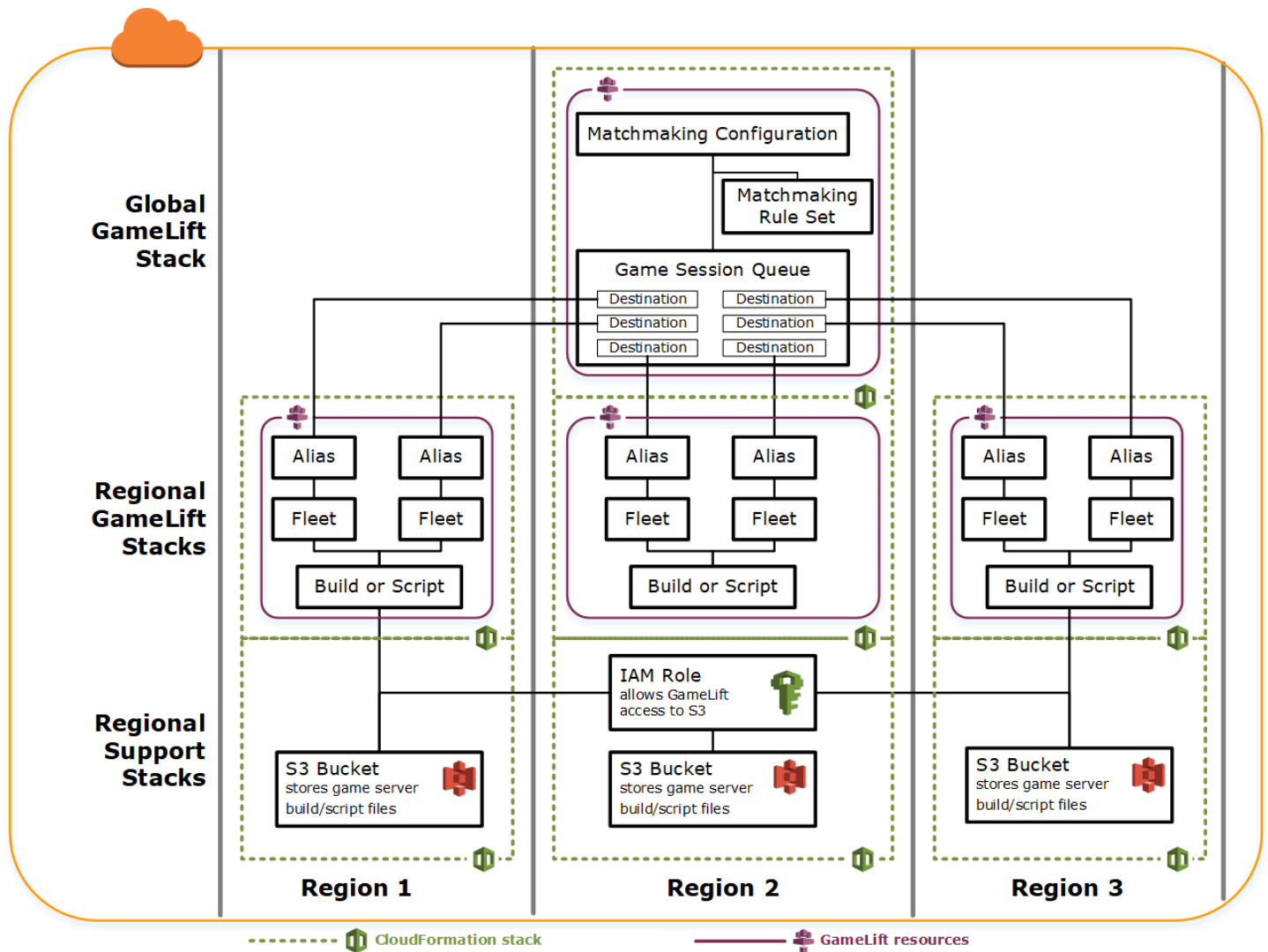
Pilas para varias regiones

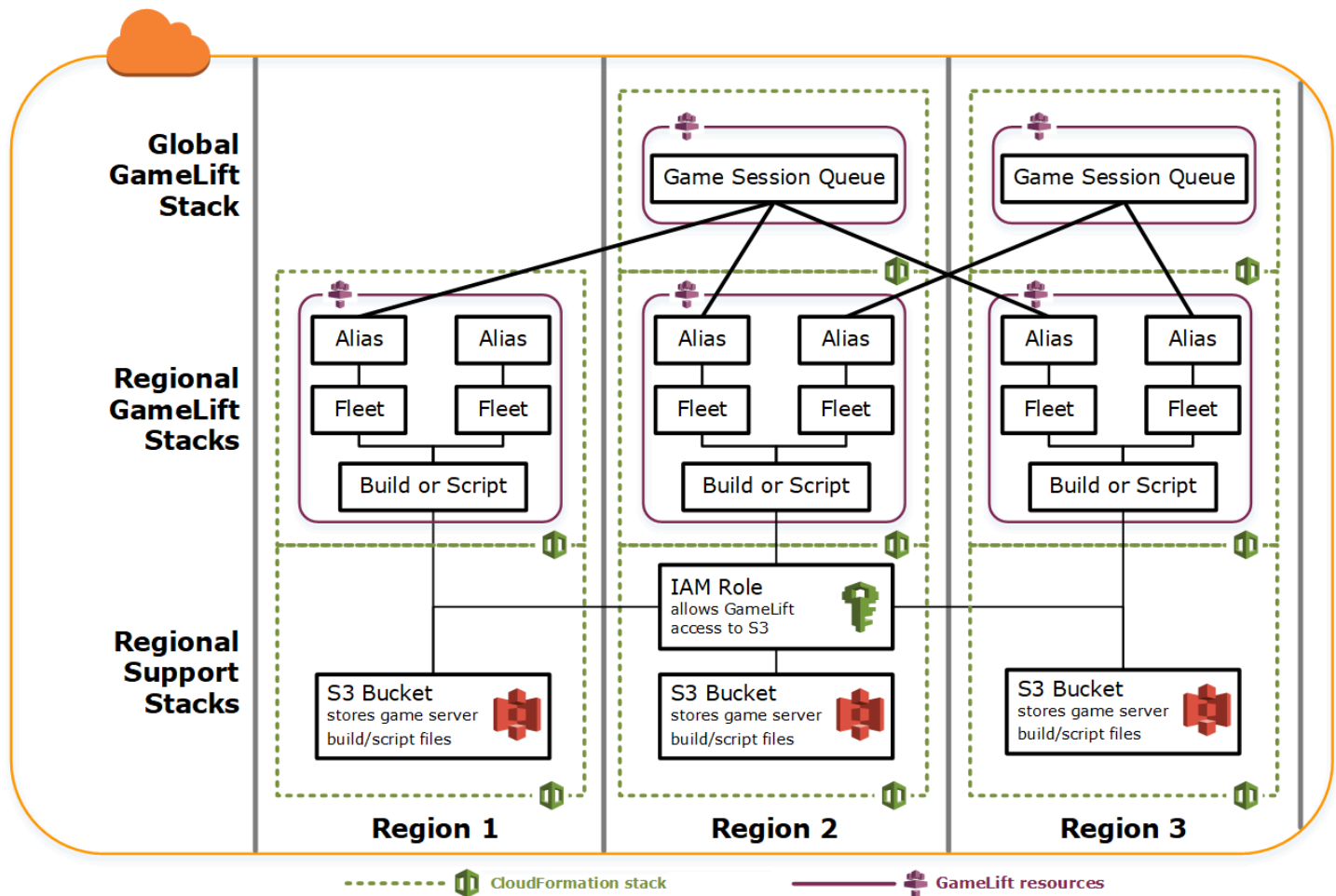
Al implementar el juego en más de una región, tenga en cuenta cómo los recursos pueden interactuar entre regiones. Algunos recursos, como las flotas de Amazon GameLift Servers, solo pueden hacer referencia a otros recursos de la misma región. Otros recursos, como una Amazon

GameLift Servers cola, son independientes de la región. Para administrar recursos de Amazon GameLift Servers en varias regiones, recomendamos la siguiente estructura.

- **Pilas de respaldo regional:** estas pilas contienen recursos de los que dependen los recursos de Amazon GameLift Servers. Esta pila debe incluir el bucket de S3 en el que almacena el servidor de juegos personalizado o los archivos de script de Realtime. También puede contener otros AWS recursos para el juego, como tablas de DynamoDB, clústeres de Amazon Redshift y funciones Lambda. Muchos de estos recursos son específicos de la región, por lo que debe crearlos en cada región. Amazon GameLift Servers también necesita un rol de IAM que permita el acceso a estos recursos auxiliares. Puesto que un rol de IAM es independiente de la región, solo necesita un recurso de rol, colocado en cualquier región y al que se haga referencia en todas las demás pilas de respaldo.
- **Pilas regionales Amazon GameLift Servers:** esta pila contiene los Amazon GameLift Servers recursos que deben existir en cada región en la que se vaya a implementar el juego, incluida la compilación o el script, un conjunto de flotas y los alias. CloudFormation crea un recurso de compilación o script con archivos en una ubicación de bucket de S3 y despliega la compilación o el script en uno o más recursos de la flota. Cada flota debe tener su alias correspondiente. La cola de sesiones del juego hace referencia a algunos o todos los alias de la flota. Puede mantener una plantilla para describir este tipo de pila y utilizarla para crear conjuntos de recursos idénticos en cada región.
- **Pila global de Amazon GameLift Servers:** esta pila contiene la cola de sesiones de juego y los recursos de emparejamiento. Estos recursos se pueden colocar en cualquier región, aunque se suelen colocar en la misma región. La cola puede hacer referencia a flotas o alias que se encuentran en cualquier región. Para colocar colas adicionales en diferentes regiones, cree pilas globales adicionales.

Los siguientes diagramas ilustran una estructura de varias pilas para implementar recursos en varias AWS regiones. El primer diagrama muestra una estructura para una sola cola de sesiones del juego. El segundo diagrama muestra una estructura con varias colas.





Actualización de las compilaciones

Las compilaciones de Amazon GameLift Servers son inmutables, al igual que la relación entre una compilación y una flota. Por consiguiente, cuando actualice sus recursos de alojamiento para usar un nuevo conjunto de archivos de compilación del juego, proceda de la siguiente forma:

- Cree una nueva compilación usando el nuevo conjunto de archivos (reemplazo).
- Cree un nuevo conjunto de flotas para implementar la nueva compilación del juego (reemplazo).
- Redirija los alias para que apunten a las nuevas flotas (actualización sin interrupción).

Para obtener más información, consulte [Comportamientos de actualización de los recursos de la pila](#) en la Guía del usuario de AWS CloudFormation .

Implementación de actualizaciones de compilación automáticamente

Al actualizar una pila que contiene recursos de construcción, flota y alias relacionados, el CloudFormation comportamiento predeterminado es realizar estos pasos de forma automática y secuencial. Esta actualización se activa al cargar primero los nuevos archivos de compilación en una nueva ubicación de S3. A continuación, modificas la plantilla de CloudFormation compilación para que apunte a la nueva ubicación de S3. Cuando actualiza la pila con la nueva ubicación S3, se desencadena la siguiente secuencia de CloudFormation :

1. Se recuperan los nuevos archivos de S3, se validan los archivos y se crea una nueva compilación de Amazon GameLift Servers.
2. Se actualiza la referencia de compilación en la plantilla de flota, lo que desencadena la creación de una nueva flota.
3. Una vez activadas las nuevas flotas, se actualiza la referencia de flota en el alias, lo que hace que el alias se actualice para apuntar a las nuevas flotas.
4. Se elimina la flota antigua.
5. Se elimina la compilación antigua.

Si la cola de sesiones del juego utiliza alias de flota, el tráfico de jugadores se cambia automáticamente a las nuevas flotas en cuanto se actualizan los alias. Las flotas antiguas se van quedando poco a poco sin jugadores a medida que terminan las sesiones del juego. El escalado automático controla la tarea de añadir y eliminar instancias de cada conjunto de flotas a medida que fluctúa el tráfico de jugadores. También puede especificar un recuento inicial de las instancias deseadas para que el cambio se produzca rápidamente y habilitar el escalado automático más adelante.

También puedes hacer que CloudFormation conserve los recursos en lugar de eliminarlos. Para obtener más información, consulta [RetainResources](#) en la AWS CloudFormation Referencia de la API de .

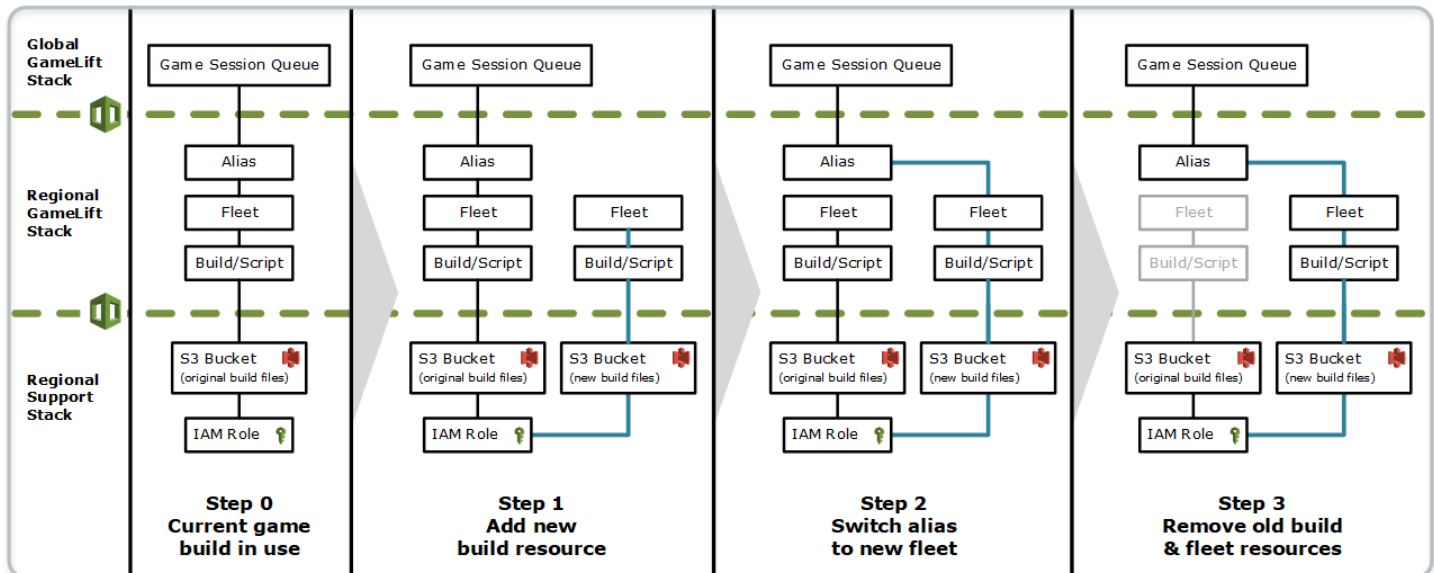
Implementación de actualizaciones de compilación manualmente

Si desea tener más control sobre cuándo se activan las nuevas flotas para los jugadores, dispone de algunas opciones. Puede elegir administrar los alias manualmente mediante la consola de Amazon GameLift Servers o la CLI. Otra opción es añadir un segundo conjunto de definiciones de compilación y flota en la plantilla, en lugar de actualizar la plantilla de compilación para reemplazar la compilación y las flotas. Al actualizar la plantilla, CloudFormation crea un segundo recurso de

construcción y las flotas correspondientes. Dado que los recursos existentes no se reemplazan, estos no se eliminan y los alias siguen apuntando a las flotas originales.

La principal ventaja de este enfoque es que le ofrece flexibilidad. Puede crear recursos distintos para la nueva versión de su compilación, probar los nuevos recursos y controlar cuándo se activan las nuevas flotas para los jugadores. Un posible inconveniente es que requiere el doble de recursos en cada región durante un breve período de tiempo.

En el siguiente diagrama se ilustra este proceso.



Cómo funcionan las restauraciones

Al ejecutar una actualización de recursos, si algún paso no se completa correctamente, CloudFormation inicia automáticamente una restauración. Este proceso invierte cada paso en orden, eliminando los recursos recién creados.

Si necesita activar manualmente una restauración, cambie la clave de ubicación de S3 de la plantilla de compilación a la ubicación original y actualice la pila. Se crea una nueva compilación y flota de Amazon GameLift Servers, y el alias cambia a la nueva flota una vez que la flota esté activa. Si administra alias por separado, debe cambiarlos para que apunten a las nuevas flotas.

Para obtener más información sobre cómo controlar una restauración que da un error o se bloquea, consulte [Continuación de la restauración de una actualización](#) en la Guía del usuario de AWS CloudFormation.

DDoS Protección S con Amazon GameLift Servers

Amazon GameLift Servers Player Gateway es una red basada en retransmisiones que enruta el tráfico UDP entre los clientes del juego y los servidores de juegos alojados en ellos. Amazon GameLift Servers Player Gateway proporciona una protección DDoS proactiva al validar el tráfico antes de que llegue a los servidores del juego, limitar la velocidad del tráfico de los jugadores, ocultar al público las direcciones IP de los servidores de juegos y proporcionar puntos de conexión actualizados cuando los puntos de enlace de retransmisión dejan de funcionar. Esta función ayuda a proteger tus servidores de juegos de los ataques volumétricos con una latencia adicional insignificante.

Player Gateway requiere que los servidores de juegos se ejecuten en flotas EC2 Amazon GameLift Servers gestionadas o flotas de contenedores basadas en Linux. Para poder utilizar esta función, debes activar Player Gateway durante la creación de la flota y actualizar el cliente y el backend del juego.

Ventajas principales

Player Gateway ofrece las siguientes ventajas operativas y de seguridad:

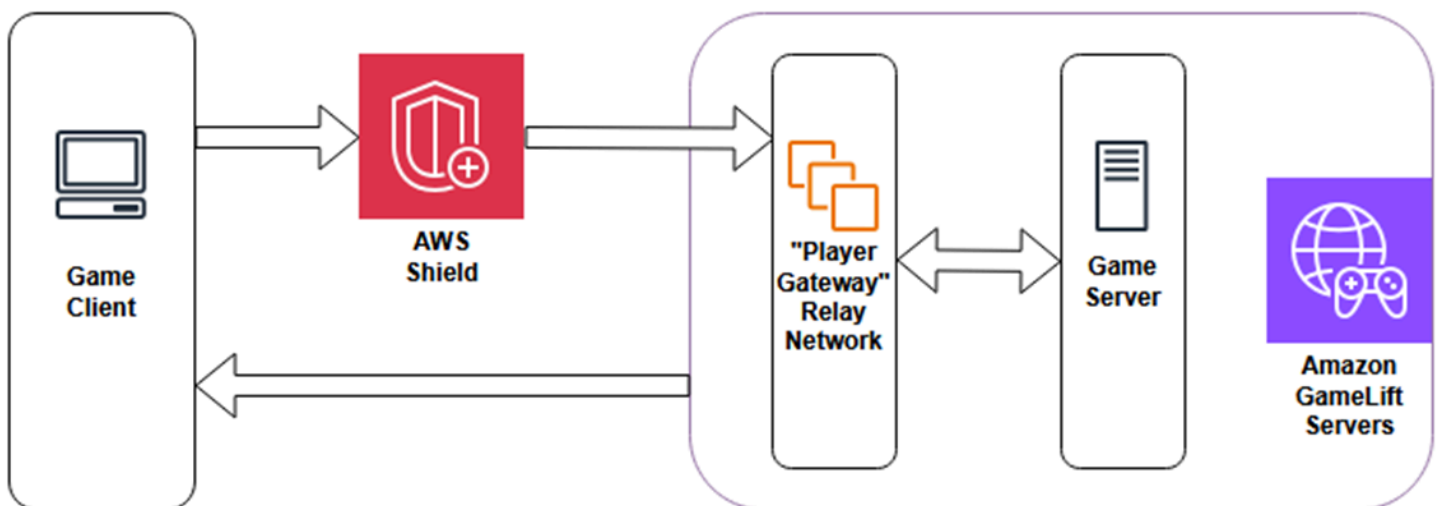
- Oculta las direcciones IP de los servidores de juegos: los clientes de juegos se conectan a través de puntos finales de retransmisión en lugar de directamente a los servidores de juegos, lo que oculta las direcciones de tus servidores de juegos al público.
- Valida el tráfico: todo el tráfico que pasa por la pasarela de jugadores requiere un token de pasarela de jugador, lo que permite que solo el tráfico de jugadores autenticados llegue a tus servidores de juegos.
- Sustitución dinámica de los terminales: cuando los terminales de retransmisión no funcionan correctamente, los sustituye Amazon GameLift Servers rápidamente y proporciona los puntos finales en buen estado actualizados en la siguiente llamada para actualizar los detalles de conexión de los jugadores.
- Distribuye el tráfico de jugadores: los puntos finales de retransmisión varían de un jugador a otro, lo que reduce el impacto que puede tener un único punto final de retransmisión en mal estado sobre los demás jugadores de la misma sesión de juego.

Cómo funciona Player Gateway

Player Gateway utiliza una red de retransmisión para enrutar el tráfico UDP entre los clientes y los servidores del juego. Esto protege contra los ataques tipo DDoS validando el tráfico antes de que llegue a los servidores del juego, limitando la velocidad del tráfico de los jugadores, ocultando al público las direcciones IP de los servidores del juego y proporcionando puntos de conexión actualizados cuando los puntos de enlace de retransmisión no funcionen correctamente.

Flujo de tráfico

Cuando un jugador se conecta a una sesión de juego, el backend del juego recupera los puntos finales de retransmisión y un token de puerta de enlace del jugador de la `GetPlayerConnectionDetails` API y los envía al cliente del juego. El cliente del juego antepone el token de puerta de enlace del jugador a los paquetes UDP y los envía a un punto final de retransmisión. La red de retransmisión valida el token y enruta el tráfico legítimo al servidor del juego. Antes de la entrega, la red de retransmisión elimina el token de puerta de enlace del jugador para que los servidores del juego reciban la carga sin procesar del cliente del juego y no deberían requerir cambios de código. La comunicación entre el servidor del juego y el cliente del juego se devuelve a través de la red de retransmisión sin modificaciones.

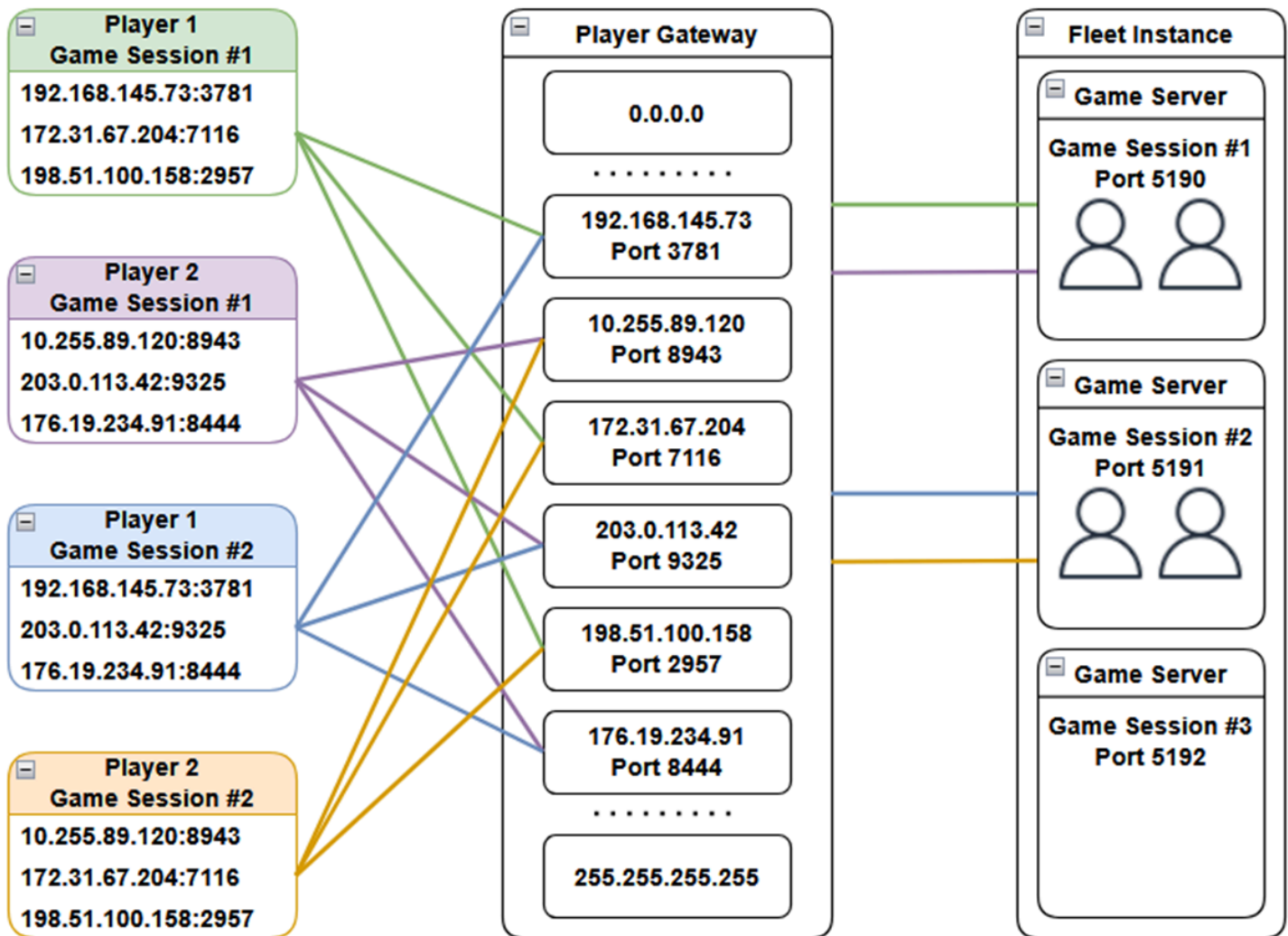


Conceptos clave

Puntos finales de retransmisión

Los puntos finales de retransmisión son combinaciones de direcciones IP y puertos que los clientes de juegos utilizan para enviar tráfico a través de la puerta de enlace del jugador. Cada jugador recibe

varios puntos finales que varían de un jugador a otro para distribuir el tráfico y reducir el impacto de los ataques sobre otros jugadores durante la misma sesión de juego.



Tokens de entrada para jugadores

Los tokens de pasarela de jugador son conjuntos de bytes cifrados que autorizan a un jugador a enviar tráfico a una sesión de juego a través de la pasarela del jugador. La `GetPlayerConnectionDetails` API devuelve los tokens como cadenas codificadas en base64. Los clientes del juego deben anteponer el token de puerta de enlace del jugador a cada paquete UDP. La red de retransmisión valida los tokens y rechaza los paquetes con tokens no válidos o faltantes.

Important

Las fichas de entrada de los jugadores no deben estar cifradas y deben aparecer al principio de cada paquete UDP enviado por el cliente del juego. Si tu juego cifra las cargas útiles, antepone el token de puerta de enlace del jugador que no haya sido modificado después de cifrar los datos del juego.

Los tokens Player Gateway siguen siendo válidos durante al menos 3 minutos. Recomendamos actualizar las fichas cada 60 segundos para garantizar que los jugadores reciban los puntos finales actualizados cuando los puntos finales de los relevos dejen de funcionar.

GetPlayerConnectionDetails API

El backend del juego llama a la `GetPlayerConnectionDetails` API para recuperar los detalles de conexión de los jugadores en una sesión de juego. La API devuelve los puntos de conexión de retransmisión y las fichas de acceso de los jugadores, o recurre a la dirección IP y al puerto del servidor del juego para establecer una conexión directa. El cliente del juego debe estar diseñado para gestionar ambos tipos de conexión. Para recibir puntos finales actualizados cuando los puntos finales de retransmisión no funcionen correctamente, llama a esta API periódicamente durante la sesión de juego (se recomienda cada 60 segundos).

Para obtener más información, consulte [GetPlayerConnectionDetails API](#).

Supervisar la pasarela del jugador

Player Gateway publica métricas en Amazon para ayudarte CloudWatch a monitorizar los patrones de tráfico de la red, identificar posibles ataques DDoS y realizar un seguimiento del rendimiento de las retransmisiones. Las métricas incluyen la entrada y salida de paquetes y bytes, el tráfico limitado y las sesiones de los jugadores. Para ver la lista completa de las métricas de las pasarelas de jugadores, consulte. [DDoMétricas de protección S \(puerta de entrada al jugador\)](#)

IPv4 y IPv6 compatibilidad

Los clientes del juego se comunican mediante IPv4. Player Gateway utiliza IPv6 para comunicarse con los servidores del juego. Amazon GameLift Servers gestiona automáticamente la traducción entre IPv4 y en IPv6 función de la configuración de tu flota.

Para obtener más información sobre cómo configurar Player Gateway en su flota, consulte [CreateFleet](#) o [CreateContainerFleet](#). Para obtener más información IPv4 y IPv6 asistencia, consulte [IPv4 y IPv6 compatibilidad](#).

Habilitar la puerta de enlace de jugadores en las flotas

Activa la pasarela de jugadores al crear una flota para enrutar el tráfico del juego a través de puntos finales de retransmisión que validan y limitan la velocidad del tráfico UDP entrante a tus servidores de juegos.

Requisitos previos

Antes de activar Player Gateway, comprueba los siguientes requisitos:

- Ubicaciones compatibles: para ver una lista de AWS ubicaciones compatibles con Player Gateway, consulta [Ubicaciones de servicio de Amazon GameLift Servers](#).
- Tipo de flota: Player Gateway es compatible con flotas de EC2 gestionadas basadas en Linux y con flotas de contenedores gestionadas.
- SDK de servidor: Player Gateway requiere Amazon GameLift Servers el SDK de servidor 5.0 o una versión posterior.

Modos Player Gateway

Configura la compatibilidad de las pasarelas de jugadores para tu flota mediante la siguiente `PlayerGatewayMode` propiedad:

DESACTIVADO (predeterminado)

Fleet no es compatible con Player Gateway.

ENABLED

Fleet es compatible con Player Gateway en los lugares en los que sea compatible. Las ubicaciones de la flota que no admitan esta función se pueden añadir a la flota, pero no utilizarán la puerta de enlace del jugador, sino que utilizarán directamente la dirección IP del servidor del juego.

REQUIRED

Fleet es compatible con Player Gateway. Las ubicaciones de la flota que no admitan esta función no se pueden añadir a la flota.

IPv4 y IPv6 compatibilidad

Los clientes del juego deben comunicarse con la pasarela del jugador a través de IPv4. La red de retransmisión de la puerta de enlace del jugador se comunica con los servidores del juego mediante IPv6. Las flotas de contenedores gestionadas traducen automáticamente el tráfico de IPv6 a IPv4. En el caso de las flotas de EC2 gestionadas, configure la forma en que su flota gestiona el IPv6 tráfico mediante la propiedad de: `GameServerIpProtocolSupportedPlayerGatewayConfiguration`

IPv4 (predeterminado)

Tu servidor de juegos solo acepta IPv4 tráfico entrante. Se instalará y ejecutará un software ligero de traducción de IP en sus instancias para recibir y transformar el IPv6 tráfico entrante en él IPv4.

DUAL_STACK

Tu servidor de juegos es compatible con el tráfico entrante. IPv6 El software de traducción de IP no se instalará ni ejecutará en sus instancias. Los servidores de juegos compatibles de IPv6 forma nativa pueden tener un rendimiento ligeramente superior.

Integre la puerta de enlace del jugador en un juego

Tras crear una flota compatible con Player Gateway, debes actualizar el cliente y el backend del juego para integrarlos con Player Gateway. El backend del juego recupera los puntos finales de retransmisión y las fichas de pasarela de los jugadores y, a continuación, se los proporciona a los clientes del juego. Los clientes del juego envían todo el tráfico UDP a estos puntos finales de retransmisión.

La integración sigue esta ruta de llamadas:

1. El backend del juego llama a la `GetPlayerConnectionDetails` API para recuperar los puntos finales de retransmisión y las fichas de acceso a los jugadores de cada jugador de una sesión de juego.

2. El backend del juego envía los puntos finales de retransmisión y las fichas de entrada de los jugadores al cliente del juego.
3. El cliente del juego antepone el token de puerta de enlace del jugador a todos los paquetes UDP salientes y los envía a los puntos finales de retransmisión.
4. La red de retransmisión valida el token de puerta de enlace del jugador y enruta el tráfico a tu servidor de juegos.
5. El servidor del juego devuelve el tráfico al cliente del juego a través de las mismas rutas de retransmisión.

Integración de back-end

El backend del juego debe llamar a la `GetPlayerConnectionDetails` API para recuperar los puntos finales de retransmisión y las fichas de acceso a los jugadores. A continuación, el backend proporciona esta información a los clientes del juego.

`GetPlayerConnectionDetails` API

La `GetPlayerConnectionDetails` API devuelve los detalles de conexión de los jugadores en una sesión de juego:

Retransmita puntos finales y fichas

Cuando la puerta de enlace del jugador está habilitada y es compatible con la ubicación, devuelve:

- Puntos finales de retransmisión: varios puntos finales de retransmisión (dirección IP y puerto) que varían de un jugador a otro
- Token de puerta de enlace del jugador: token que el cliente debe anteponer a todos los paquetes UDP (válido durante al menos 3 minutos)
- Vencimiento: fecha y hora de caducidad del token Player Gateway

Conexión directa

Si la puerta de enlace del jugador no está habilitada o no es compatible con la ubicación, devuelve la dirección IP y el puerto del servidor del juego

El cliente del juego debe estar diseñado para gestionar ambos tipos de conexión.

Ejemplo de llamada a la API:

```
// C++ example using AWS SDK
Aws::GameLift::GameLiftClient client;
Aws::GameLift::Model::GetPlayerConnectionDetailsRequest request;
request.SetGameSessionId(gameSessionId);
request.SetPlayerIds(playerIds); // Vector of player IDs

auto outcome = client.GetPlayerConnectionDetails(request);
if (outcome.IsSuccess()) {
    auto result = outcome.GetResult();
    auto connectionDetails = result.GetPlayerConnectionDetails();

    // Process each player's connection details
    for (const auto& detail : connectionDetails) {
        std::string playerId = detail.GetPlayerId();

        // Get relay endpoints (IP address and port)
        auto endpoints = detail.GetEndpoints();
        for (const auto& endpoint : endpoints) {
            std::string ipAddress = endpoint.GetIpAddress();
            int port = endpoint.GetPort();
        }

        // Get player gateway token
        auto token = detail.GetPlayerGatewayToken();

        // Get expiration time
        auto expiration = detail.GetExpiration();

        // Send endpoints and token to game client
    }
}
```

Consulte [GetPlayerConnectionDetails](#) en la referencia Amazon GameLift Servers de la API.

Actualice los puntos finales de retransmisión

Llama `GetPlayerConnectionDetails` periódicamente para proporcionar a los jugadores puntos finales actualizados cuando los puntos finales de los relevos no funcionen correctamente. Recomendamos que todos los jugadores de una sesión de juego llamen a la API cada 60 segundos. Para optimizar el uso de la API, agrupa a varios jugadores IDs en una sola llamada a la API.

Important

Las llamadas de actualización periódicas son el mecanismo principal para ofrecer puntos finales actualizados a los jugadores. Si bien las fichas de acceso a los jugadores siguen siendo válidas durante al menos 3 minutos, si se actualizan cada 60 segundos, los jugadores reciben puntos de conexión actualizados con regularidad.

Integración con el cliente

El cliente del juego debe anteponer los tokens de pasarela del jugador a todos los paquetes UDP salientes y enviarlos a los puntos finales de retransmisión proporcionados. El Amazon GameLift Servers cliente SDK for C++ y Unreal Engine proporciona utilidades para simplificar esta integración.

Requisitos del cliente:

Para dirigir el tráfico a través de Player Gateway, el cliente del juego debe:

- Anteponga los tokens de la puerta de enlace del jugador: anteponga el token de la puerta de enlace del jugador a todos los paquetes UDP salientes. Los tokens de pasarela de jugador no deben estar cifrados y deben aparecer al principio de cada paquete. Los paquetes que no contengan un token de entrada de jugador válido se eliminarán.
- Enviar a los puntos finales de retransmisión: envíe todos los paquetes UDP a los puntos finales de retransmisión proporcionados.
- Mantén la actividad de conexión: asegúrate de que el cliente del juego envíe un paquete al servidor del juego o que el servidor del juego envíe un paquete al jugador, al menos una vez cada 30 segundos. Esta actividad mantiene la conexión a través de la red de retransmisión.
- Gestiona las actualizaciones de los puntos de conexión: acepta los puntos de enlace de retransmisión y las fichas de acceso de los jugadores actualizados desde tu servidor (se recomienda cada 60 segundos) y cambia a nuevos puntos de enlace sin interrumpir la conexión.

SDK de cliente

El SDK de Amazon GameLift Servers cliente para C++ y Unreal Engine proporciona utilidades para simplificar la integración de las pasarelas de reproducción:

- Administración de fichas: coloca las fichas de las pasarelas del jugador en todos los paquetes UDP salientes.

- Selección de puntos finales: enruta el tráfico a los puntos finales de retransmisión mediante un algoritmo configurable.
- Actualización de los terminales: programa llamadas periódicas para recuperar de tu servidor los puntos finales de retransmisión actualizados y las fichas de acceso a los jugadores.

Algoritmos de selección de

El SDK del cliente incluye dos algoritmos integrados para seleccionar el punto final de retransmisión que se va a utilizar:

Algoritmo alternativo

Utiliza un punto final hasta que deja de funcionar correctamente y, a continuación, cambia a otro punto final. Ideal para menús, vestíbulos y juegos por turnos en los que se aceptan interrupciones breves. Durante la conmutación por error del terminal, es posible que se pierdan paquetes durante el tiempo de espera configurado (predeterminado: 2 segundos).

Algoritmo de rotación predictivo

Rota continuamente por todos los puntos finales disponibles y predice los fallos antes de que se produzcan. Ideal para juegos en tiempo real, como juegos de disparos en primera persona y juegos de carreras, donde la entrega constante de paquetes es fundamental. Requiere que el servidor del juego envíe mensajes con una frecuencia constante.

También puedes implementar algoritmos personalizados ampliando la clase de algoritmos base del SDK.

Recursos del SDK del cliente

Para obtener instrucciones de integración completas y códigos de ejemplo, consulte los siguientes recursos:

C++

Hay disponible un SDK de cliente para C++ en [Amazon GameLift ServersClient SDK for C++](#).

Unreal Engine

Hay un complemento para Unreal Engine disponible en el [SDK del Amazon GameLift Servers cliente para Unreal](#) Engine. El acceso requiere ser miembro de la organización de Epic Games a partir del. GitHub Consulta [Unreal Engine en GitHub](#) para obtener más información.

Realización de una prueba de integración

Antes de implementarlo en una flota, prueba la integración de la pasarela de jugadores de forma local con la herramienta de pruebas de la pasarela de jugadores. Esta herramienta simula la infraestructura de proxy UDP de la puerta de enlace del jugador y te ayuda a comprobar si el cliente del juego coloca correctamente los tokens de la puerta de enlace del jugador, dirige el tráfico a través de varios puntos finales y gestiona la degradación de la red.

Para obtener instrucciones de configuración y uso, consulta la aplicación de pruebas de [Player Gateway](#) en GitHub.

Prácticas recomendadas

Sigue estas prácticas recomendadas a la hora de integrar Player Gateway:

- Actualiza los puntos finales cada 60 segundos: llama `GetPlayerConnectionDetails` cada 60 segundos para asegurarte de que los jugadores reciban los puntos finales actualizados con regularidad.
- Llamadas a la API **`GetPlayerConnectionDetails`** por lotes: cuando llames a varios jugadores en la misma sesión de juego, agrupa las llamadas para reducir la sobrecarga de la API.
- Mantén la actividad de conexión: asegúrate de que el tráfico fluya entre el cliente del juego y el servidor del juego al menos una vez cada 30 segundos. Puede ser server-to-client tráfico client-to-server o bien.
- Gestiona las actualizaciones de los terminales de forma adecuada: cuando el cliente del juego reciba los puntos de conexión actualizados del servidor, realiza la transición a los nuevos puntos de conexión sin interrumpir la conexión del jugador.
- Elige el algoritmo correcto: selecciona el algoritmo de selección de puntos finales que mejor se adapte a los requisitos de tu juego. Usa el algoritmo alternativo para las partidas por turnos y el algoritmo de rotación predictiva para las partidas en tiempo real.

Pruebas y solución de problemas de Amazon GameLift Servers

Esta sección proporciona orientación para diagnosticar y resolver problemas con tu solución de GameLift hosting de Amazon. Si experimenta problemas durante las operaciones de desarrollo, implementación o producción, estos recursos le ayudarán a identificar las causas raíz e implementar soluciones eficaces.

- **Entornos de desarrollo y prueba:** configure flujos de trabajo de desarrollo iterativos que recreen las condiciones de producción, lo que le permitirá detectar y resolver los problemas de integración en una etapa temprana del ciclo de desarrollo.
- **Implementación y administración de flotas:** diagnostique los fallos en la creación de flotas, los problemas de activación de procesos del servidor, los problemas de escalado y los errores de implementación más frecuentes que pueden impedir que los servidores de juegos funcionen correctamente.
- **Diagnóstico y monitoreo del tiempo de ejecución:** acceda a las instancias de flota activas para solucionar problemas, analizar el rendimiento y validar las configuraciones en tiempo real con el fin de resolver los problemas que afectan a la experiencia de los jugadores.

Usa estos recursos junto con las capacidades GameLift de monitoreo y registro de Amazon para mantener un rendimiento y una disponibilidad óptimos para tus juegos multijugador.

Preparación para el desarrollo iterativo con Amazon GameLift Servers Anywhere

Amazon GameLift Servers proporciona herramientas y soluciones para ayudarlo a configurar un entorno de pruebas alojado para el uso durante el desarrollo de juegos. Con estas herramientas, puede crear un entorno que refleje la experiencia real de los jugadores con el alojamiento administrado con Amazon GameLift Servers y que permita un proceso de desarrollo rápido e iterativo.

Con un entorno de pruebas independiente, se eliminan los gastos generales de una flota administrada por Amazon GameLift Servers durante las pruebas. Ya no es necesario cargar cada nueva iteración de compilación del servidor de juegos, crear una flota para esta y esperar más de

15 minutos para que se active. En lugar de todo eso, puede crear una nueva compilación, actualizar rápidamente la flota de pruebas con la nueva compilación, iniciarla y empezar las pruebas.

Con una flota de Amazon GameLift Servers Anywhere, puede configurar un entorno de prueba con un dispositivo local, como una estación de trabajo de desarrollo. También puede configurar un entorno de prueba mediante un recurso de alojamiento basado en la nube.

Configure un entorno de prueba de Anywhere para desarrollar y probar distintos escenarios, como los siguientes:

- Pruebe la integración del servidor de juegos con el SDK del servidor de Amazon GameLift Servers. Puedes realizar pruebas incluso sin un cliente de juego que funcione mediante llamadas AWS CLI para iniciar nuevas sesiones de juego y realizar un seguimiento de los eventos de las sesiones de juego.
- Pruebe las interacciones entre el cliente del juego, el servicio de backend y el servicio de Amazon GameLift Servers a medida que desarrolla componentes para el juego. Afine la experiencia del jugador al unirse a una partida.
- Experimente con el diseño de emparejador de FlexMatch. Pruebe las variaciones del conjunto de reglas y otras implementaciones de características de emparejamiento. Configure y pruebe el relleno de emparejamiento.
- Pruebe otras características de alojamiento de Amazon GameLift Servers, como los ajustes de configuración del tiempo de ejecución (con el agente de Amazon GameLift Servers) para administrar el ciclo de vida de los servidores de juegos.
- Cree, pruebe y repita rápidamente para validar todos los aspectos de la experiencia del jugador, lo que incluye las interacciones multijugador, en un entorno alojado y en directo.

Más adelante, cuando prepare el juego para el lanzamiento, será recomendable que añada flotas administradas por Amazon GameLift Servers para refinar las configuraciones de alojamiento y probar escenarios adicionales, como los siguientes:

- Experimente y haga pruebas con los diseños de colas de las sesiones de juego, lo que incluye el uso de flotas con múltiples ubicaciones, flotas de spot y bajo demanda y muchos tipos de instancias.
- Pruebe las opciones de ubicación de las sesiones de juego con flotas administradas, como el uso de políticas de latencia opcionales y los ajustes de priorización de flotas.
- Configure el escalado de la capacidad para adaptarlo a la demanda de jugadores, usando para ello opciones de escalado manual o automático.

- Configure AWS CloudFormation con flotas Amazon GameLift Servers gestionadas para gestionar sus recursos de alojamiento a largo plazo.

Fast Build Update Tool (solo para desarrollo)

Con EC2 las flotas gestionadas, para implementar una actualización de compilación de un servidor de juegos, debes cargar cada nueva versión Amazon GameLift Servers y crear una nueva flota para ella.

Con Fast Build Update Tool, puede omitir estos pasos durante el desarrollo, lo que le permite ahorrar tiempo y agilizar la iteración del desarrollo. Con esta herramienta, puede actualizar rápidamente los archivos de compilación de juegos en todos los recursos de computación de una flota existente. La herramienta tiene varias opciones: puede reemplazar una compilación de juego completa o cambiar 6 archivos específicos, y puede administrar cómo reiniciar los procesos del servidor de juegos tras las actualizaciones; también puede usarla para actualizar los recursos de computación de una flota.

Para conseguir Fast Build Update Tool y obtener más información sobre su uso, visite el repositorio del kit de herramientas de Amazon GameLift Servers para [Fast Build Update Tool](#) en GitHub.

Temas

- [Creación de un entorno de pruebas basado en la nube](#)
- [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#)
- [Prueba de la integración con Amazon GameLift Servers local](#)

Creación de un entorno de pruebas basado en la nube

Note

En este tema se cubren únicamente las pruebas iterativas de juegos integrados en la versión 5.x del SDK del servidor de Amazon GameLift Servers. Si el juego usa la versión 4.x o anterior del SDK del servidor, consulte [Prueba de la integración con Amazon GameLift Servers local](#).

Utilice una flota de Amazon GameLift Servers Anywhere para compilar y probar de forma iterativa los componentes de un juego en un entorno de alojamiento basado en la nube. Cree una flota de Anywhere con recursos de alojamiento y una conexión al servicio Amazon GameLift Servers, ejecute los servidores de juegos en ellos y pruebe las funciones del juego según sea necesario.

Implementación de una flota de Anywhere con el agente de Amazon GameLift Servers

Si la compilación del servidor de juegos está integrada con el SDK 5.x (o una versión posterior) de Amazon GameLift Servers, puede implementarla en una flota de basada en la nube con el agente de Amazon GameLift Servers. El agente es un proceso en segundo plano que administra los ciclos de vida de los servidores de juegos, así como otras tareas en todos los recursos de computación de una flota. Estas tareas incluyen el registro de la computación en una flota de Anywhere, la adquisición de un token de autenticación y los procesos del servidor del starting/stopping juego en función de un conjunto de instrucciones. El agente se controla mediante una configuración de tiempo de ejecución de la flota, que puede actualizar en cualquier momento durante la vida útil de la flota (el agente se implementa automáticamente en las flotas de EC2 administradas). Para obtener más información y descargar el agente, consulte el [Amazon GameLift Servers GitHub repositorio](#).

Configuración de pruebas iterativas con Amazon EC2

Utilice el flujo de trabajo guiado de esta [solución del kit de herramientas de Amazon GameLift Servers](#) a fin de configurar un entorno de alojamiento basado en la nube que simule la experiencia de alojamiento administrado por Amazon GameLift Servers.

El GitHub repositorio proporciona un conjunto de scripts que automatizan la mayoría de los procesos para configurar un entorno de prueba con Amazon GameLift Servers Anywhere y el Amazon GameLift Servers agente. También encontrará orientación para actualizar el entorno cuando tenga que probar una nueva compilación del servidor de juegos. Puede ejecutar un único script que implemente un entorno de pruebas con una compilación de servidor de juegos de muestra, o puede seguir los pasos para configurarlo con su propia compilación de servidor de juegos.

En este flujo de trabajo, trabajará completamente en el Consola de administración de AWS, utilizando AWS CloudShell para ejecutar scripts y completar tareas de línea de comandos.

Note

Para las tareas de este tutorial, necesita un usuario de AWS cuenta con permisos para los siguientes servicios: Amazon GameLift Servers AWS CloudShell,, Amazon S3 AWS Systems

Manager, Amazon EC2 y. AWS Identity and Access Management Los usuarios con acceso de nivel de administrador a la AWS cuenta ya tienen los permisos necesarios.

El flujo de trabajo abarca las siguientes tareas:

- Empaquetado de una la compilación de servidor de juegos para Amazon GameLift Servers.El flujo de trabajo incluye un script para crear un ejemplo de servidor de juegos en C++, que ya se ha integrado en el SDK 5.x del servidor de Amazon GameLift Servers y está listo para el alojamiento. Si lo prefiere, también puede trabajar con su propio proyecto de juego si ha completado la integración.
- Configurar un bucket de Amazon Simple Storage Service para almacenar las compilaciones y dependencias de los servidores de juegos. Al crear nuevas versiones de compilaciones de juegos, podrá almacenarlas en S3 y utilizar los scripts para actualizar la flota de Anywhere a fin de probarlas.
- Obtención y compilación del agente de Amazon GameLift Servers. El agente administra los procesos del servidor de juegos en un recurso de alojamiento basado en la configuración. Utiliza la misma lógica que el alojamiento de EC2 administrado por Amazon GameLift Servers, y se comporta de forma idéntica a este.
- Configuración de una flota de Anywhere para sus recursos de alojamiento. Con una flota de Anywhere, puede utilizar el servicio Amazon GameLift Servers para alojar recursos que Amazon GameLift Servers no administre. En este paso, también establecerá la configuración del tiempo de ejecución, que le indica al agente de Amazon GameLift Servers cuándo y cómo debe iniciar los procesos del servidor de juegos.
- Configurar una instancia de Amazon EC2. Este es el entorno de pruebas para llevar a cabo las pruebas iterativas. Es mucho más rápido utilizar una instancia de EC2 estándar que una instancia de Amazon GameLift Servers totalmente administrada (que está optimizada para el uso en producción). Con una instancia de EC2 estándar, puede actualizar el servidor de juegos de forma rápida y continua, según sea necesario.
- Implementación de la compilación del servidor de juegos y el agente de Amazon GameLift Servers en la instancia de Amazon EC2.El flujo de trabajo proporciona un script que obtiene la última versión de la compilación del juego y todas las dependencias, y la instala en la instancia de EC2. En este flujo de trabajo, las dependencias incluyen el Amazon GameLift Servers agente y el agente. CloudWatch
- Inicio del agente de Amazon GameLift Servers. Tras la instalación, el agente se inicia automáticamente y empieza a ejecutar las instrucciones. Entre ellos se incluyen:

- Registrar la instancia de EC2 como recurso de computación en la flota de Amazon GameLift Servers Anywhere.
- Establezca una WebSocket conexión con el Amazon GameLift Servers servicio y obtenga la configuración de tiempo de ejecución más reciente.
- Iniciar los procesos del servidor de juegos con base en las instrucciones de la configuración del tiempo de ejecución. En este flujo de trabajo, el agente recibe la indicación de iniciar un único proceso del ejecutable del servidor de juegos.
- Probar los escenarios de juego. Cuando el entorno de pruebas esté configurado y se haya instalado la última versión del servidor de juegos, podrá empezar las pruebas. El flujo de trabajo incluye varios pasos para realizar las pruebas, como el inicio de una sesión de juego. Accede a los registros del servidor del CloudWatch juego para hacer un seguimiento del progreso a medida que se inicia la sesión de juego y se prepara para aceptar jugadores.

A medida que vaya desarrollando los componentes del juego, como un cliente de juego y un servicio de backend en el cliente, podrá incluirlos en los escenarios de prueba. Use un cliente de juego para solicitar una sesión de juego, recuperar los datos de conexión del servicio Amazon GameLift Servers y, luego, conectarse directamente a la sesión de juego.

- Implementar una nueva compilación del servidor de juegos y repetir las pruebas. Durante el desarrollo del juego, podrá generar nuevas compilaciones del servidor de juegos y, a continuación, implementarlas rápidamente en el entorno de pruebas de EC2 para probarlas. Cárguelas en el bucket de Amazon S3 y utilice los scripts del flujo de trabajo para actualizar el entorno de pruebas.

Transición de un juego a las flotas administradas por Amazon GameLift Servers

Cuando haya completado las pruebas de desarrollo y esté listo para preparar el lanzamiento, es un buen momento para cambiar a las flotas administradas por Amazon GameLift Servers. Utilice las flotas administradas para afinar y probar los recursos de alojamiento de juegos. Implemente la solución de ubicación de las sesiones de juego (colas y emparejadores), seleccione las mejores ubicaciones y el mejor hardware de alojamiento (lo que incluye las flotas de spot) posible y elija una estrategia para escalar la capacidad. También puedes empezar a utilizarlos AWS CloudFormation para gestionar de forma más eficiente los ciclos de vida de todos tus recursos de alojamiento de juegos, incluidas las flotas, las colas y los matchmakers.

La transición de una flota de pruebas de Anywhere basada en la nube a una flota administrada por Amazon GameLift Servers es muy sencilla. No es necesario cambiar el código del juego y puede utilizar las mismas colas y los mismos emparejadores. Haga lo siguiente:

- Creación de un recurso de compilación de Amazon GameLift Servers. Con una flota de prueba de Anywhere, tendrá que implementar manualmente la compilación y las dependencias del servidor de juegos en cada uno de los recursos de computación de la flota. Con una flota administrada, solo tiene que cargar el paquete de compilación del juego a Amazon GameLift Servers: este lo implementará automáticamente en todos los recursos de computación de la flota. Consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#) para obtener más información sobre cómo empaquetar los archivos de compilación del juego y cómo crear un recurso de compilación con archivos en un bucket de Amazon S3.
- Cree una flota administrada. Cree una flota mediante la consola o la CLI de AWS , y especifique que se trata de una flota de EC2 administrada. Este tipo de flotas requiere ajustes de configuración adicionales; por ejemplo, debe especificar cuáles son los tipos de instancia y recursos de compilación. Puede usar la misma configuración de tiempo de ejecución para administrar el ciclo de vida de los servidores de juegos en todos los recursos de computación de la flota. Consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#) para obtener más información sobre la creación de una flota administrada.
- Redireccione los alias de la flota (opcional). Si ha configurado alias para usarlos en las flotas de Anywhere, puede reutilizar los mismos alias para las flotas administradas. Consulte [Creación de un alias de Amazon GameLift Servers](#) para obtener más información sobre cómo crear o actualizar un alias.

Configuración de pruebas locales con Amazon GameLift Servers Anywhere

Note

En este se cubren únicamente las pruebas de juegos integrados en la versión 5.x del SDK del servidor de Amazon GameLift Servers. Si el juego usa la versión 4.x o anterior del SDK del servidor, consulte [Prueba de la integración con Amazon GameLift Servers local](#).

Utilice una flota de Amazon GameLift Servers Anywhere y su propio hardware para compilar y probar de forma iterativa los componentes de un juego en un entorno de alojamiento simulado. Configure una flota de Anywhere y registre un dispositivo local para establecer una conexión con el servicio Amazon GameLift Servers. Instale la compilación del servidor de juegos en el dispositivo, inicie un proceso de servidor de juegos y pruebe la funcionalidad del juego según sea necesario. Puede actualizar la versión de la compilación del servidor de juegos tantas veces como sea necesario para probar las nuevas iteraciones de la compilación.

Con una flota de Anywhere, puede realizar pruebas mediante la AWS CLI o con scripts de prueba. Si ha integrado un cliente de juegos en Amazon GameLift Servers, puede ejecutar el cliente en el mismo dispositivo local o en un dispositivo diferente.

Las pruebas locales con una flota de Anywhere son especialmente útiles para probar la integración del servidor de juegos con Amazon GameLift Servers. Dispone de visibilidad total sobre toda la actividad de alojamiento en el equipo local, así como sobre los datos de registro y eventos.

Note

¿Utiliza el complemento de Amazon GameLift Servers para Unreal Engine o Unity? Estas herramientas incluyen flujos de trabajo guiados para configurar las pruebas locales con una flota de Anywhere. Siga la documentación para [Complemento para Unity: configuración de pruebas locales con Amazon GameLift Servers Anywhere](#) o [Complemento para Unreal: alojamiento local del juego con Amazon GameLift Servers Anywhere](#).

Temas

- [Configuración de una flota de Anywhere local](#)
- [Actualización e instalación del servidor de juegos](#)
- [Pruebas en la actividad de la sesión de juego](#)
- [Iteración en el servidor de juegos](#)
- [Transición de un juego a las flotas administradas por Amazon GameLift Servers](#)

Configuración de una flota de Anywhere local

Siga estos pasos para crear una flota de Anywhere en su estación de trabajo local. Para obtener instrucciones detalladas mediante la AWS CLI o el Consola de administración de AWS for Amazon GameLift Servers, consulte [Creación de una flota de Amazon GameLift Servers Anywhere](#).

Creación de la flota de Anywhere

1. Cree una ubicación personalizada para la estación de trabajo local (AWS CLI o consola). Una ubicación personalizada no es más que una etiqueta para el recurso de computación que planea incluir en la flota de Anywhere. Los nombres de ubicaciones personalizadas deben empezar por custom-. Por ejemplo: custom-my_laptop. Consulte [Creación de una ubicación personalizada](#).

2. Cree una flota en cualquier lugar (AWS CLI o consola). En este paso, cree el recurso de flota con la ubicación personalizada para la estación de trabajo local. Consulte [Creación de una flota de Anywhere](#).

Anote el valor de ID o ARN de la nueva flota. Lo necesitará para el siguiente paso.

3. Registre su estación de trabajo local como informática de flota (solo AWS CLI). Una flota de Anywhere debe tener, al menos, un recurso de computación para alojar servidores de juegos. Consulte [Añadir un recurso de computación a la flota](#). Para añadir un recurso de computación a la flota, necesita la siguiente información:
 - Un nombre de recurso de computación. Todos los recursos de computación de una flota deben tener un nombre exclusivo.
 - El identificador de la flota de Anywhere. Puede usar el FleetID o el FleetArn.
 - Los datos de conexión del recurso de computación. Especifique un IpAddress o DnsName. Así es como Amazon GameLift Servers y los clientes de juego se conectarán a los servidores del juego.
 - Una ubicación personalizada en la flota de Anywhere.

Anote el valor de retorno de `GameLiftServiceSdkEndpoint`. Necesitará este valor cuando actualice el servidor del juego para que se ejecute en una flota de Anywhere.

Actualización e instalación del servidor de juegos

En esta tarea, se presupone que ya ha integrado una compilación del servidor de juegos en el SDK 5.x del servidor de Amazon GameLift Servers. El proceso de integración implica añadir un código al servidor de juegos para que pueda interactuar con el servicio Amazon GameLift Servers a fin de iniciar y administrar las sesiones de juego.

En el caso de una flota de Anywhere, debe configurar manualmente algunos ajustes del servidor de juegos. En una flota administrada por Amazon GameLift Servers, estos ajustes se configuran automáticamente.

Preparación del servidor de juegos para una flota de Anywhere

1. Obtener un token de autenticación. El servidor de juegos debe incluir un token de autenticación en todas las comunicaciones con el servicio Amazon GameLift Servers. Los tokens de

autenticación de Amazon GameLift Servers son de corta duración y deben actualizarse regularmente.

Una práctica recomendada es crear un script para completar las siguientes tareas:

- Llame a la acción AWS CLI `get-compute-auth-token`.
- Guardar el valor del token devuelto en un lugar en el que los procesos del servidor del juego lo puedan recuperar; por ejemplo, en una variable de entorno del recurso de computación local.

Instale el script con el servidor de juegos en el recurso de computación. Configure el script para que se ejecute antes de que se inicie el primer proceso del servidor de juegos. Mientras los procesos del servidor de juegos estén activos, ejecute el script regularmente para seguir teniendo un token de autenticación válido. Todos los procesos del servidor de juegos del recurso de computación pueden usar el mismo token de autenticación.

2. Actualice el código del servidor de juegos de Amazon GameLift Servers. Al integrar el código del servidor de juegos en el SDK del servidor de Amazon GameLift Servers, ha añadido una llamada a la acción `InitSdk()`. Cuando el servidor de juegos se ejecuta en una flota de Anywhere, esta llamada requiere parámetros de servidor adicionales. Para obtener más información, consulte los artículos [Inicialización del proceso del servidor](#) y [SDK 5.x del servidor de Amazon GameLift Servers](#) correspondientes para su lenguaje de desarrollo. A continuación, tiene una lista con los parámetros del servidor.

- `websocketUrl`: en este parámetro, ponga el valor `GameLiftServiceSdkEndpoint` que se devuelve al registrar un recurso de computación en la flota.
- `hostId`: en este parámetro, ponga el nombre del recurso de computación que especifique al registrar un recurso de computación en la flota de Anywhere.
- `fleetId`: en este parámetro, ponga el ID de la flota de Anywhere.
- `authToken`: en este parámetro, ponga el token devuelto en respuesta a una solicitud de recuperación de token de autenticación para un recurso de computación.
- `processId`: ponga este parámetro de modo que identifique a un proceso del servidor de juegos que se esté ejecutando en el recurso de computación local; cada proceso simultáneo del servidor de juegos debe tener un ID de proceso único.

Los valores de los parámetros del servidor empleados por cada proceso del servidor de juegos deben ser específicos para el recurso de computación de la flota de Anywhere en la que se esté ejecutando el proceso. Para obtener información detallada sobre cómo obtener los valores

adecuados para un recurso de computación, consulte [Añadir un recurso de computación a la flota](#). Como práctica recomendada, defina `websocketUrl`, `hostId`, `fleetId` y `authToken` como variables de entorno en el recurso de computación local. Todos los procesos del servidor que se ejecuten en el recurso de computación utilizarán estos valores.

3. Instale la compilación del servidor de juegos en el recurso de computación local. Incluya todas las dependencias necesarias para ejecutar el servidor de juegos.
4. Inicie uno o más de los procesos de servidor de juegos ejecutados en el recurso de computación local. Cuando el proceso del servidor de juegos llame a la acción `ProcessReady()` del SDK del servidor, el proceso estará listo para alojar una sesión de juego.

Pruebas en la actividad de la sesión de juego

Para probar la integración del servidor de juegos, trabaje con las sesiones de juego. Si no tiene un cliente de juego integrado con la funcionalidad de Amazon GameLift Servers, puede usar la CLI de AWS para iniciar sesiones de juego. Pruebe los siguientes escenarios:

- Cree una sesión de juego. Llame al [create-game-session](#) comando (o a la operación [CreateGameSession](#) de la API). Especifique la ubicación personalizada y el ID de la flota de Anywhere. Esta llamada devuelve un identificador único para la nueva sesión de juego.
- Comprobar el estado de la sesión de juego. [describe-game-sessions](#) Comando de llamada (o acción [DescribeGameSessions](#) de la API). Especifique el ID de la sesión de juego. Esta llamada devuelve información detallada de la sesión de juego, lo que incluye el estado de la sesión de juego. Las sesiones de juego en estado activo están listas para que los jugadores se conecten. Para obtener una lista de todas las sesiones de juego de la flota, llama al [list-game-sessions](#) comando (o a la acción de la [ListGameSessions](#) API).
- Conectarse a la sesión de juego. Si el cliente de juego puede unirse a una sesión de juego, use los datos de conexión incluidos en la información de la sesión de juego.

Iteración en el servidor de juegos

Puede usar el mismo recurso de computación y la misma flota de Anywhere para probar otras versiones de la compilación del servidor de juegos.

1. Limpiar la **GameSession** existente. Si el proceso del servidor de juegos se bloquea o no llama a `ProcessEnding()`, Amazon GameLift Servers limpiará la `GameSession` cuando el servidor de juegos deje de enviar comprobaciones de estado.

2. Generar una nueva compilación del servidor de juegos. Haga cambios en el servidor de juegos y empaquete una compilación revisada.
3. Actualizar la compilación del servidor de juegos en el recurso de computación local. Su flota de Anywhere anterior sigue activa y su portátil sigue registrado como recurso informático en la flota.
4. Obtener un token de autorización actualizado. Llame al comando [get-compute-auth-token](#) CLI y almacene el token en el equipo local.
5. Iniciar uno o más de los procesos de servidor de juegos ejecutados en el recurso de computación local. Cuando el proceso del servidor de juegos llame a `ProcessReady()`, estará listo para el uso en las pruebas.

Transición de un juego a las flotas administradas por Amazon GameLift Servers

Cuando haya completado las pruebas de desarrollo y esté listo para preparar el lanzamiento, es un buen momento para cambiar a las flotas administradas por Amazon GameLift Servers. Utilice las flotas administradas para afinar y probar los recursos de alojamiento de juegos. Implemente la solución de ubicación de las sesiones de juego (colas y emparejadores), seleccione las mejores ubicaciones y el mejor hardware de alojamiento (lo que incluye las flotas de spot) posible y elija una estrategia para escalar la capacidad. También puedes empezar a utilizarlos AWS CloudFormation para gestionar de forma más eficiente los ciclos de vida de todos tus recursos de alojamiento de juegos, incluidas las flotas, las colas y los matchmakers.

Para pasar de una flota de prueba local de Anywhere a una flota administrada por Amazon GameLift Servers, debe hacer una serie de pequeños cambios. Puede volver a utilizar las mismas colas y los mismos emparejadores. Haga lo siguiente:

- Cambie la llamada del código del servidor de juegos a **InitSdk()**. Elimine los parámetros del servidor. En el caso de una flota administrada, Amazon GameLift Servers hace un seguimiento automático de esta información.
- Creación de un recurso de compilación de Amazon GameLift Servers. Con una flota de prueba de Anywhere, tendrá que implementar manualmente la compilación y las dependencias del servidor de juegos en cada uno de los recursos de computación de la flota. Con una flota administrada, usted crea y carga el paquete de compilación del juego en Amazon GameLift Servers, y este lo implementa automáticamente en todos los recursos de computación de la flota. Consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#) para obtener más información sobre cómo empaquetar los archivos de compilación del juego y cómo crear un recurso de compilación con archivos en un bucket de Amazon S3. No incluya scripts que

registren un recurso de computación y obtengan un token de autenticación, ya que, para flotas administradas, Amazon GameLift Servers gestiona automáticamente estas tareas.

- Cree una flota administrada. Cree una flota mediante la consola o la AWS CLI, especificando una flota gestionada por EC2. Este tipo de flotas requiere ajustes de configuración adicionales; por ejemplo, debe especificar cuáles son los tipos de instancia y recursos de compilación. Debe establecer una configuración de tiempo de ejecución para administrar el ciclo de vida de los servidores de juegos en todos los recursos de computación de la flota. Consulte [Cree una EC2 flota Amazon GameLift Servers gestionada](#) para obtener más información sobre la creación de una flota administrada.
- Redireccione los alias de la flota (opcional). Si ha configurado alias para usarlos en las flotas de Anywhere, puede reutilizar los mismos alias para las flotas administradas. Consulte [Creación de un alias de Amazon GameLift Servers](#) para obtener más información sobre cómo crear o actualizar un alias.

Prueba de la integración con Amazon GameLift Servers local

Note

En este se cubren únicamente las pruebas de juegos integrados en las versiones 3.x y 4.x del SDK del servidor de Amazon GameLift Servers. El paquete del SDK del servidor incluye una versión compatible de Amazon GameLift Servers local. Si utiliza la versión 5.x del SDK del servidor, consulte [Pruebas locales con el SDK 5.x del SDK del servidor de GameLift](#) para realizar pruebas locales con una flota de Amazon GameLift Servers Anywhere. Para obtener instrucciones sobre cómo migrar del SDK 4.x a 5.x, consulte [Migración al SDK 5.x del servidor de GameLift](#).

Utilice Amazon GameLift Servers Local para ejecutar una versión limitada del servicio Amazon GameLift Servers administrado en un dispositivo local y probar cómo se integra el juego. Esta herramienta es útil al realizar el desarrollo iterativo en la integración del juego. La alternativa, que consiste en cargar cada nueva compilación en Amazon GameLift Servers y configurar una flota para alojar el juego, puede tardar varios minutos, o más, cada vez.

Con Amazon GameLift Servers Local, puede verificar lo siguiente:

- Que el servidor de juegos se integra correctamente con el SDK del servidor y comunica de forma adecuada con el servicio Amazon GameLift Servers para empezar nuevas sesiones de juego, aceptar jugadores nuevos e informar sobre el estado.
- Que el cliente de juego está integrado correctamente con el SDK de AWS para Amazon GameLift Servers y que puede recuperar información sobre sesiones de juego existentes, iniciar sesiones de juego nuevas, conectar jugadores a juegos y conectarse a la sesión de juego.

Amazon GameLift Servers Local es una herramienta de línea de comandos que inicia una versión autónoma del servicio Amazon GameLift Servers administrado. Amazon GameLift Servers local también proporciona un registro de evento de ejecución de la inicialización del proceso del servidor, comprobaciones de estado y llamadas y respuestas de la API. Amazon GameLift Servers local reconoce un subconjunto de las acciones del SDK de AWS para Amazon GameLift Servers. Puede realizar llamadas desde la AWS CLI o desde el cliente de juego. Todas las acciones de la API que se ejecutan localmente lo hacen de la misma forma que en el servicio web de Amazon GameLift Servers.

Cada proceso del servidor solo debe alojar una sesión de juego. La sesión de juego es el archivo ejecutable que se utiliza para conectarse a Amazon GameLift Servers local. Cuando se complete la sesión de juego, debe llamar a `GameLiftServerSDK::ProcessEnding` y salir del proceso. Al realizar pruebas de forma local con Amazon GameLift Servers local, puede iniciar varios procesos del servidor. Cada proceso se conectará a Amazon GameLift Servers local. A continuación, podrá crear una sesión de juego para cada proceso del servidor. Cuando finalice la sesión de juego, el proceso del servidor de juegos debería cerrarse. A continuación, debe iniciar manualmente otro proceso de servidor.

Amazon GameLift Servers admite las siguientes API:

- `CreateGameSession`
- `CreatePlayerSession`
- `CreatePlayerSessions`
- `DescribeGameSessions`
- `DescribePlayerSessions`

Configuración de Amazon GameLift Servers local

Amazon GameLift Servers Local se suministra en un archivo `.jar` ejecutable que se incluye en el [SDK del servidor](#). Se puede ejecutar en Windows o Linux y utilizarse con cualquier lenguaje compatible con Amazon GameLift Servers. La versión 4.0.2 del SDK se puede descargar desde las [versiones oficiales de GitHub](#). El archivo `GameLiftLocal.jar` se incluye en el artefacto `GameLift-CSharp-ServerSDK-4.0.2.zip`.

Antes de ejecutar Local, también debe tener instalado lo siguiente.

- Una compilación del SDK del servidor para las versiones de la 3.1.5 a la 4.x. de Amazon GameLift Servers.
- Java 8

Prueba de un servidor de juegos

Si solo desea probar el servidor de juegos, puede utilizar la AWS CLI para simular llamadas del cliente de juego al servicio Amazon GameLift Servers Local. De este modo, se verifica que el servidor de juegos se comporta según lo esperado:

- El servidor de juegos se lanza correctamente e inicializa el SDK del servidor de Amazon GameLift Servers.
- Como parte del proceso de lanzamiento, el servidor de juegos notifica a Amazon GameLift Servers que el servidor está listo para alojar sesiones de juego.
- El servidor de juegos envía estados a Amazon GameLift Servers cada minuto mientras se ejecuta.
- El servidor de juegos responde a las solicitudes para iniciar una sesión de juego nueva.

1. Inicie Amazon GameLift Servers Local.

Abra una ventana de símbolo del sistema, vaya al directorio que contiene el archivo ***GameLiftLocal.jar*** y ejecútelo. De forma predeterminada, Local atiende a las solicitudes de clientes de juego en el puerto 8080. Para especificar un número de puerto diferente, utilice el parámetro `-p`, tal y como se muestra en el ejemplo siguiente:

```
java -jar GameLiftLocal.jar -p 9080
```


En cuanto Local arranque, verá los registros que indican que se iniciaron dos servidores locales, uno que atiende al servidor de juegos y otro al cliente de juego o a la AWS CLI. Los registros continúan informando sobre la actividad de los dos servidores locales, incluida la comunicación a y desde los componentes de juego.

2. Inicie el servidor de juegos.

Arranque el servidor de juegos integrado en Amazon GameLift Servers localmente. No es necesario cambiar el punto de enlace del servidor de juegos.

En la ventana de símbolo del sistema de Local, los mensajes de registro indican que el servidor de juegos se ha conectado al servicio Amazon GameLift Servers Local. Esto significa que el servidor de juegos ha inicializado correctamente el SDK del servidor de Amazon GameLift Servers (con `InitSDK()`). Ha llamado a `ProcessReady()` con las rutas de registro que se muestran y, en caso de éxito, está listo para alojar una sesión de juego. Mientras se ejecuta el servidor de juegos, Amazon GameLift Servers registra cada informe de estado del servidor de juegos. El siguiente ejemplo de mensajes de registro muestra un servidor de juegos integrado correctamente:

```
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
```

A continuación se presentan posibles mensajes de error y advertencia:

- Error: “ProcessReady no encontró un proceso con pID: *<ID de proceso>*. ¿Se ha invocado InitSDK()?”
- Advertencia: “El estado del proceso ya existe para el proceso con pID: *<ID de proceso>*. ¿Se ha invocado ProcessReady(...) más de una vez?»

3. Inicie AWS CLI.

En cuanto el servidor de juegos llame a `ProcessReady()` correctamente, podrá comenzar a realizar llamadas de cliente. Abra una ventana de símbolo del sistema y empiece con la herramienta AWS CLI. De forma predeterminada, la AWS CLI utiliza el punto de enlace de servicio web de Amazon GameLift Servers. Debe anularlo con el punto de enlace de Local en cada solicitud que emplee el parámetro `--endpoint-url`, tal y como se muestra en el siguiente ejemplo.

```
AWS gamelift describe-game-sessions --endpoint-url http://localhost:9080 --fleet-id fleet-123
```

En la ventana de símbolo del sistema de la AWS CLI, los comandos de `AWS gamelift` generan respuestas similares a las documentadas en la [Referencia de comandos de la AWS CLI](#).

4. Cree una sesión de juego.

Con la AWS CLI, envíe una solicitud [CreateGameSession\(\)](#). La solicitud debería seguir la sintaxis esperada. Para Local, el parámetro `FleetId` puede adaptarse a cualquier cadena válida (`^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
```

En la ventana de símbolo del sistema de Local, los mensajes de registro indican que Amazon GameLift Servers Local ha enviado una devolución de la llamada `onStartGameSession` al servidor de juegos. Si se crea una sesión de juego correctamente, el servidor de juegos responderá llamado a `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: gameSessionId:
    "arn:aws:gamelift:local::gamesession/
fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d/gsess-abcdef12-3456-7890-abcd-
ef1234567890"

```

En la ventana de la AWS CLI, Amazon GameLift Servers responde con un objeto de sesión de juego que incluye un ID de sesión de juego. Observe que el estado de la nueva sesión de juego es `Activating`. El estado cambia a `Active` cuando el servidor de juegos invoca a `ActivateGameSession`. Si desea ver el estado modificado, utilice la AWS CLI para llamar a `DescribeGameSessions()`.

```

{
  "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
  }
}

```

Prueba de un servidor y un cliente de juegos

Para comprobar la integración total del juego, incluidos los jugadores conectados a los juegos, puede ejecutar tanto el servidor como el cliente de juego localmente. De este modo podrá realizar llamadas

programáticas desde su cliente de juego a Amazon GameLift Servers Local. Puede verificar las siguientes acciones:

- El cliente de juego está realizando correctamente solicitudes del SDK de AWS al servicio Amazon GameLift Servers Local, incluida la creación de sesiones de juego, la recuperación de información sobre sesiones de juego existentes y la creación de sesiones de jugador.
- El servidor de juegos está validando jugadores correctamente al intentar conectarse a una sesión de juego. Para validar los jugadores, el servidor de juego puede recuperar datos de los jugadores (si están implementados).
- El servidor de juegos informa sobre la pérdida de la conexión cuando un jugador abandona el juego.
- El servidor de juegos informa sobre la finalización de una sesión de juego.

1. Inicie Amazon GameLift Servers Local.

Abra una ventana de símbolo del sistema, vaya al directorio que contiene el archivo *GameLiftLocal.jar* y ejecútelo. De forma predeterminada, Local atiende a las solicitudes de clientes de juego en el puerto 8080. Para especificar un número de puerto diferente, utilice el parámetro `-p`, tal y como se muestra en el ejemplo siguiente.

```
./gamelift-local -p 9080
```

En cuanto Local arranque, verá los registros que indican que se iniciaron dos servidores locales, uno que atiende al servidor de juegos y otro al cliente de juego o a la AWS CLI.

2. Inicie el servidor de juegos.

Arranque el servidor de juegos integrado en Amazon GameLift Servers localmente. Consulte [Prueba de un servidor de juegos](#) para obtener más información acerca de los registros de mensajes.

3. Configure su cliente de juego para Local e inícielo.

Para utilizar el cliente de juego con el servicio Amazon GameLift Servers Local, debe realizar los siguientes cambios en la configuración del cliente de juego, tal y como se describe en [Configuración de la API de Amazon GameLift Servers](#):

- Modifique el objeto `ClientConfiguration` para que apunte hacia el punto de enlace de Local, por ejemplo `http://localhost:9080`.

- Defina un valor de ID de la flota de destino. Para Local, no se requiere un ID de flota real; defina la flota de destino con cualquier cadena válida (`^fleet-\S+`), como `fleet-1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d`.
- Defina las credenciales de AWS. Para Local, no requiere credenciales de AWS reales, de modo que puede definir la clave de acceso y la clave secreta para cualquier cadena.

En la ventana de símbolo del sistema de Local, después de iniciar el cliente de juego, los mensajes de registro deberían indicar que ha inicializado `GameLiftClient` y que se comunica correctamente con el servicio Amazon GameLift Servers.

4. Pruebe las llamadas del cliente de juego al servicio Amazon GameLift Servers.

Compruebe que el cliente de juego realiza correctamente alguna o todas las llamadas a la API siguientes:

- [`CreateGameSession\(\)`](#)
- [`DescribeGameSessions\(\)`](#)
- [`CreatePlayerSession\(\)`](#)
- [`CreatePlayerSessions\(\)`](#)
- [`DescribePlayerSessions\(\)`](#)

En la ventana de símbolo del sistema de Local, solo las llamadas a `CreateGameSession()` resultan en mensajes de registro. Los mensajes de registro muestran el momento en que Amazon GameLift Servers Local solicita al servidor de juegos que inicie una sesión de juego (devolución de llamada `onStartGameSession`) y obtiene un `ActivateGameSession` correcto cuando el servidor de juegos lo llama. En la ventana de la AWS CLI, todas las llamadas a la API generan respuestas o mensajes de error tal y como se documenta.

5. Compruebe que el servidor de juegos valida las conexiones de jugadores nuevas.

Después de crear una sesión de juego y una sesión de jugador, establezca una conexión directa con la sesión de juego.

En la ventana de símbolo del sistema de Local, los mensajes de registro deberían mostrar que el servidor de juegos ha enviado una solicitud `AcceptPlayerSession()` para validar la conexión de jugadores nueva. Si utiliza la AWS CLI para llamar a `DescribePlayerSessions()`, el estado de la sesión de jugador debería cambiar de `Reserved` a `Active`.

6. Compruebe que el servidor de juegos informa sobre el estado del juego y del jugador al servicio Amazon GameLift Servers.

Para que Amazon GameLift Servers administre la demanda de los jugadores y notifique correctamente las métricas, el servidor de juegos debe informar sobre varios estados a Amazon GameLift Servers. Compruebe que Local registra los eventos relacionados con las acciones siguientes. Es posible que también desee utilizar la AWS CLI para realizar un seguimiento de los cambios de estado.

- El jugador se desconecta de una sesión de juego: los mensajes de registro de Amazon GameLift Servers local deberían mostrar que el servidor de juegos llama a `RemovePlayerSession()`. Una llamada de la AWS CLI a `DescribePlayerSessions()` debería mostrar un cambio de estado de `Active` a `Completed`. También puede llamar a `DescribeGameSessions()` para comprobar que el recuento de jugadores actual de la sesión de juego resulta en un jugador menos.
- La sesión de juego finaliza: los mensajes de registro de Amazon GameLift Servers local deberían mostrar que el servidor de juegos llama a `TerminateGameSession()`.

 Note

La guía anterior consistía en llamar a `TerminateGameSession()` al finalizar una sesión de juego. Este método ha quedado obsoleto en la versión 4.0.1 del SDK del servidor de Amazon GameLift Servers. Consulte [Finalización de una sesión de juego](#).

- El proceso del servidor ha finalizado: los mensajes de registro de Amazon GameLift Servers local deberían mostrar que el servidor de juegos llama a `ProcessEnding()`. Una llamada de la AWS CLI a `DescribeGameSessions()` debería mostrar un cambio de estado de `Active` a `Terminated` (o `Terminating`).

Variaciones de Local

Al utilizar Amazon GameLift Servers Local, tenga en cuenta lo siguiente:

- A diferencia del servicio web de Amazon GameLift Servers, Local no realiza el seguimiento del estado del servidor ni inicia la devolución de la llamada `onProcessTerminate`. Local simplemente deja de registrar los informes de estado del servidor de juegos.

- Para realizar llamadas al SDK de AWS, los ID de la flota no están validados y puede tratarse de cualquier valor de cadena que cumpla los requisitos de parámetro (^fleet-\S+).
- Los ID de sesión de juego creados con Local tienen una estructura diferente. Incluyen la cadena `local`, tal y como se muestra aquí:

```
arn:aws:gamelift:local::gamesession/fleet-123/gsess-56961f8e-  
db9c-4173-97e7-270b82f0daa6
```

Depuración de problemas relacionados con flotas de Amazon GameLift Servers

En este tema se proporciona orientación sobre cómo resolver los problemas con las EC2 flotas Amazon GameLift Servers gestionadas.

Problemas al crear una flota

Al crear una EC2 flota gestionada, el Amazon GameLift Servers servicio inicia un flujo de trabajo que crea la flota, despliega las EC2 instancias con la versión del servidor de juegos instalada e inicia los procesos del servidor de juegos en cada instancia. Para obtener una descripción detallada, consulte [Depuración de problemas de flotas de Amazon GameLift Servers](#). Una flota no puede albergar sesiones de juego ni jugadores hasta que alcance el estado Activo.

Para solucionar los problemas que impiden que las flotas se activen, es necesario identificar la fase de creación de flota en la que se produjo el problema y revisar los eventos y registros de creación de flota. Si los registros no aportan ninguna información útil, es posible que el problema se deba a un error de servicio interno. En ese caso, intente crear de nuevo la flota. Si el problema continúa, cargue de nuevo la compilación del juego para solucionar posibles daños en el archivo. También puede ponerse en contacto con el equipo de soporte de Amazon GameLift Servers o publicar una pregunta en el foro.

Descarga y validación de la compilación

Durante esta fase, Amazon GameLift Servers obtiene la compilación del servidor de juegos cargada, extrae los archivos y ejecuta los scripts de instalación. Si la creación de la flota falla durante estas fases, observe los eventos y registros de la flota para identificar el problema. Entre las causas posibles se incluyen las siguientes:

- Amazon GameLift Servers no puede obtener el archivo de compilación comprimido (evento `FLEET_BINARY_DOWNLOAD_FAILED`). Comprueba que se puede acceder a la ubicación de

almacenamiento de la compilación, que estás creando una flota similar a la de la compilación y que Amazon GameLift Servers tienes los permisos correctos para acceder a ella. Región de AWS

- Amazon GameLift Servers no puede extraer los archivos de compilación (evento `FLEET_CREATION_EXTRACTING_BUILD`).
- No se ha podido completar correctamente uno de los scripts de instalación incluido en los archivos de compilación (evento `FLEET_CREATION_FAILED_INSTALLER`).

Construcción de los recursos de flota

Normalmente, los problemas durante esta fase están relacionados con la asignación y la implementación de los recursos de flota. Entre las causas posibles se incluyen las siguientes:

- El tipo de instancia solicitado no está disponible.
- El tipo de flota solicitado (spot o bajo demanda) no está disponible.

Activación de los procesos del servidor de juegos

Durante esta fase, Amazon GameLift Servers intenta llevar a cabo distintas tareas y prueba elementos importantes, como la viabilidad del servidor de juegos, la configuración del tiempo de ejecución y la capacidad del servidor de juegos de conectarse al servicio de Amazon GameLift Servers mediante el SDK del servidor.

Note

En esta fase, puede acceder de forma remota a una instancia de flota para seguir analizando los problemas. Consulte [Conexión remota a instancias de flota de Amazon GameLift Servers](#).

Estos son algunos de los problemas que podría encontrar:

- Los procesos del servidor no comienzan a ejecutarse. Esto sugiere que hay un problema con los ajustes de configuración del tiempo de ejecución de la flota (eventos `FLEET_VALIDATION_LAUNCH_PATH_NOT_FOUND` o `FLEET_VALIDATION_EXECUTABLE_RUNTIME_FAILURE`). Compruebe si ha configurado correctamente la ruta de lanzamiento y los parámetros de lanzamiento opcionales.
- Los procesos del servidor empiezan a ejecutarse, pero la flota no se activa. Si los procesos del servidor arrancan y se ejecutan correctamente, pero la flota no pasa al estado Activo, es probable que la causa sea que el proceso del servidor no esté informando al servicio de

Amazon GameLift Servers. Compruebe que el servidor de juegos haga correctamente estas llamadas al SDK del servidor (consulte [Iniciación del proceso del servidor](#)):

- El proceso del servidor no se inicializa (evento `SERVER_PROCESS_SDK_INITIALIZATION_TIMEOUT`). El proceso del servidor no llama correctamente a `InitSdk()`.
- El proceso del servidor no envía una notificación a Amazon GameLift Servers cuando está listo para alojar una sesión de juego (evento `SERVER_PROCESS_PROCESS_READY_TIMEOUT`). El proceso del servidor se inicializa, pero no llama a `ProcessReady()` a tiempo.
- Hay un error en la solicitud de conexión de emparejamiento de VPC. Para las flotas que se crean con una interconexión de VPC (consulte [Para configurar la interconexión de VPC con una nueva flota](#)), la interconexión de VPC se realiza durante esta fase Activating (Activando). Si una interconexión de VPC devuelve un error por cualquier motivo, la flota nueva no podrá pasar al estado Activo. Puedes hacer un seguimiento del éxito o el fracaso de la solicitud de peering llamando [describe-vpc-peering-connections](#). Asegúrese de comprobar que existe una autorización de emparejamiento de VPC válida ([describe-vpc-peering-authorizations](#), ya que las autorizaciones solo son válidas durante 24 horas).

Problemas con los procesos del servidor

Los procesos del servidor arrancan pero dan error rápidamente o notifican que no están funcionando correctamente.

Aparte de los problemas con la compilación del juego, esto puede ocurrir al intentar ejecutar demasiados procesos del servidor simultáneamente en la instancia. La cantidad óptima de procesos simultáneos depende del tipo de instancia y de los requisitos de los recursos del servidor de juegos. Intente reducir el número de procesos simultáneos, que se define en la configuración del tiempo de ejecución de la flota, para ver si mejora el desempeño. Puedes cambiar la configuración de tiempo de ejecución de una flota mediante la Amazon GameLift Servers consola (edita los ajustes de asignación de capacidad de la flota) o llamando al comando AWS CLI [update-runtime-configuration](#)

Problemas de eliminación de la flota

La flota no se puede terminar debido a un recuento máximo de instancias.

El mensaje de error indica que la flota que se está eliminando todavía tiene instancias activas, lo que no está permitido. En primer lugar, debe reducir una flota a cero instancias activas. Esto se realiza configurando manualmente el recuento de instancia de la flota deseado a "0" y, a continuación, esperar a que la reducción de escala surta efecto. Asegúrese de desactivar el escalado automático, lo que contrarrestará los ajustes manuales.

Las acciones de VPC no están autorizadas.

Este problema solo se aplica a las flotas para las que haya creado específicamente conexiones de emparejamiento de VPC (consulte [Configuración del emparejamiento de VPC para Amazon GameLift Servers](#)). Este escenario se produce porque el proceso de eliminación de una flota también incluye la eliminación de la VPC de la flota y de cualquier conexión de emparejamiento de VPC. Primero debe obtener una autorización llamando a la API de servicio for Amazon GameLift Servers [CreateVpcPeeringAuthorization\(\)](#) o utilizando el comando AWS CLI `create-vpc-peering-authorization`. Una vez que tenga la autorización, puede eliminar la flota.

Conexión remota a instancias de flota de Amazon GameLift Servers

Puede conectarse a cualquier instancia de las flotas de contenedores o de EC2 administradas por Amazon GameLift Servers activas. Estas son algunas de las razones más habituales para acceder de forma remota a una instancia:

- Solucionar problemas con la integración del servidor de juegos.
- Afinar la configuración del tiempo de ejecución y otros ajustes específicos de la flota.
- Obtener datos de actividad en tiempo real del servidor de juegos, como el seguimiento del registro.
- Ejecutar herramientas de evaluación comparativa mediante el tráfico real de jugadores.
- Investigar problemas específicos relacionados con una sesión de juego o un proceso del servidor.

Al conectarse a una instancia, tenga en cuenta estos posibles problemas:

- Puede conectarse a cualquier instancia de una flota activa. Por lo general, no puede conectarse a flotas inactivas, como las que están en el proceso de activación o las que tienen un estado de error

(es posible que estas flotas tengan una disponibilidad limitada durante un corto período de tiempo). Si necesita ayuda en relación con los problemas de activación de la flota, consulte [Depuración de problemas relacionados con flotas de Amazon GameLift Servers](#).

- La conexión a una instancia activa no afecta a la actividad de alojamiento de la instancia. La instancia sigue iniciando y deteniendo los procesos del servidor en función de la configuración del tiempo de ejecución. Activa y ejecuta sesiones de juego. Es posible que la instancia se cierre en respuesta a un evento de reducción vertical o de otro tipo.
- Los cambios que haga en los archivos o la configuración de una instancia pueden afectar a las sesiones de juego activas de la instancia y a los jugadores conectados.

Acceso remoto a través de la consola

Puede conectarse a instancias de la flota directamente desde la consola de Amazon GameLift Servers mediante Amazon EC2 Systems Manager (SSM). Este método proporciona un acceso seguro sin necesidad de configuración adicional ni administración de credenciales.

1. En la consola de Amazon GameLift Servers, elija EC2 administrado o Contenedores administrados en el panel de navegación, seguido de Flotas.
2. Elija el ID de la flota que contiene la instancia a la que desea acceder.
3. En la página de detalles de la flota, seleccione la pestaña Instancias para ver todas las instancias de computación de la flota.
4. Seleccione la instancia a la que desea conectarse y, a continuación, elija Conectar. Esto abre el cuadro de diálogo Conectar a la instancia, que le informa de los detalles de la conexión y le permite ver el script que se utilizará para conectarse a la instancia. Para confirmar, seleccione Conectar nuevamente.
5. En el cuadro de diálogo de conexión, elija Ejecutar para crear una nueva sesión de SSM. El sistema autentica la sesión a través del Servicio de administración de AWS claves (AWS KMS) y abre una terminal en el navegador.

Note

El acceso remoto basado en consola está disponible para las flotas que ejecutan el SDK del servidor versión 5.x. Para las flotas que ejecutan versiones anteriores del SDK, utilice el método AWS CLI que se describe en la siguiente sección.

Acceso remoto con la AWS CLI

Las siguientes instrucciones describen cómo conectarse de forma remota a una instancia mediante la interfaz de la línea de comandos (CLI) de AWS . También puede realizar llamadas programáticas mediante el AWS SDK, tal y como se documenta en la [referencia de la API de servicio correspondiente Amazon GameLift Servers](#) a.

Recopilación de datos de instancias

Para conectarse a una instancia de flota de EC2 administrada por Amazon GameLift Servers, necesita la siguiente información:

- El ID de la instancia a la que desea conectarse. Puede utilizar el ARN o el ID de la instancia.
- La versión del SDK del servidor de Amazon GameLift Servers que se utiliza en la instancia. El SDK del servidor está integrado con la compilación del juego que se ejecuta en la instancia.

Las siguientes instrucciones describen cómo completar estas tareas mediante la AWS CLI. Debe conocer el ID de la flota de la instancia a la que desea conectarse.

1. Obtenga el nombre del recurso de computación. Obtenga una lista de todos los recursos de computación activos de la flota. Llame a [list-compute](#) con el ARN o ID de una flota. Si se trata de una flota de una sola ubicación, especifique únicamente el identificador de la flota. Si es una flota con múltiples ubicaciones, especifique el identificador de la flota y una ubicación. En el caso de las flotas de EC2 administradas, `list-compute` devuelve una lista de instancias de flota y la propiedad `ComputeName` es el ID de la instancia. Busque el recurso de computación al que desea acceder.

Solicitud

```
aws gamelift list-compute \  
  --fleet-id "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa" \  
  --location ""sa-east-1"
```

Respuesta

```
{  
  "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":
    "b08444ki909kvqu6zpw3is24x5pyz4b6m05i3jbxvpk9craztu0lqrbbrbnbkks.uwp57060n1k6dnlnw49b78hg1
west-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. Busque la versión del SDK del servidor. Para obtener esta información, debe buscar la compilación implementada en la flota. La versión del SDK del servidor es una propiedad de compilación.
 - a. Llame [describe-fleet-attributes](#) con un identificador de flota o un ARN para obtener el identificador de construcción y el ARN de la flota.
 - b. Llame a [describe-build](#) con el ARN o el ID de la compilación para obtener la versión del SDK del servidor de compilación.

Por ejemplo:

Solicitud

```

aws gamelift describe-fleet-attributes /
--fleet-ids "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"

```

Respuesta

```

{
  "FleetAttributes": [
    {
      "FleetId": "fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa",
      "ComputeType": "EC2",
      "BuildId": "build-3333cccc-44dd-55ee-66ff-00001111aa22",
      . . .
    }
  ]
}

```

```
}  
]  
}
```

Solicitud

```
aws gamelift describe-build /  
  --build-id "build-3333cccc-44dd-55ee-66ff-00001111aa22"
```

Respuesta

```
"Build": {  
  "BuildId": "build-1111aaaa-22bb-33cc-44dd-5555eeee66ff",  
  "Name": "My_Game_Server_Build_One",  
  "OperatingSystem": "AMAZON_LINUX_2023",  
  "ServerSdkVersion": "5.1.1",  
  . . .  
}
```

Conexión a una instancia (SDK 5 de servidor)

Si la instancia a la que quiere conectarse ejecuta una compilación de juego con la versión 5.x del SDK del servidor, conéctese a la instancia mediante Amazon EC2 Systems Manager (SSM). Puede acceder a las instancias remotas que se ejecuten en Windows o Linux.

Antes de comenzar:

Complete los pasos de configuración de SSM e instale el complemento de SSM en el equipo local. Para obtener más información, consulte [Configuración de SSM](#) e [instalación del complemento Session Manager para la AWS CLI en la](#) Guía del usuario de Amazon EC2 Systems Manager.

1. Solicite credenciales de acceso para la instancia. Llame [get-compute-access](#) con el ID de flota y el nombre de procesamiento de la instancia a la que desee conectarse. Amazon GameLift Servers devuelve un conjunto de credenciales temporales para acceder a la instancia. Por ejemplo:

Solicitud

```
aws gamelift get-compute-access \  
--compute-name i-11111111a222b333c \  
--fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \  
--region us-west-2
```

Respuesta

```
{  
  "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. Exporte las credenciales de acceso (opcional). Puede exportar las credenciales a variables de entorno y utilizarlas para configurar la AWS CLI para el usuario predeterminado. Para obtener más información, consulte [Variables de entorno para configurar la AWS CLI](#) en la Guía del AWS Command Line Interface usuario.

```
export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE  
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
```

3. Conéctese a la instancia de la flota. Inicie una sesión de SSM con la instancia a la que desea conectarse. Incluya la AWS región o la ubicación de la instancia. Para obtener más información, incluido cómo configurar SSM y el complemento SSM, consulte [Iniciar una sesión \(AWS CLI\)](#) en la Guía del usuario de Amazon EC2 Systems Manager.

La solicitud start-session utilizará automáticamente las credenciales que haya obtenido en el paso 1.

```
aws ssm start-session \  
--target i-11111111a222b333c \  

```

```
--region us-west-2 \
```

Note

Si recibe un error de acceso denegado, es posible que tenga una variable de `AWS_PROFILE` entorno configurada en un AWS perfil, lo que provoca que la AWS CLI utilice credenciales incorrectas para el acceso remoto. Para resolver este problema, desactive temporalmente la variable de entorno `AWS_PROFILE`. Como alternativa, puede crear un AWS perfil personalizado para sus credenciales de acceso remoto y añadir el parámetro de línea de `--profile` comandos a su `start-session` solicitud.

Conexión a una instancia (SDK 4.x, o anterior, de servidor)

Si la instancia a la que quiere conectarse ejecuta una compilación de juego con la versión 4, o anterior, del SDK del servidor, observe las siguientes instrucciones. Puede conectarse a instancias que se ejecuten en Windows o Linux. Para conectarse a una instancia de Windows, utilice un cliente de protocolo de escritorio remoto (RDP). Para conectarse a una instancia de Linux, utilice un cliente SSH.

1. Solicite credenciales de acceso para la instancia. Cuando tengas un ID de instancia, usa el comando [get-instance-access](#) para solicitar las credenciales de acceso. Si funciona correctamente, Amazon GameLift Servers devuelve el sistema operativo, la dirección IP y un conjunto de credenciales (nombre de usuario y clave secreta) de la instancia. El formato de las credenciales depende del sistema operativo de la instancia. Siga las siguientes instrucciones para recuperar las credenciales para RDP o SSH.
 - Para instancias Windows: para conectarse a una instancia Windows, RDP requiere un nombre de usuario y una contraseña. La solicitud `get-instance-access` devuelve estos valores en forma de cadenas simples, por lo que puede utilizar los valores devueltos tal cual. Credenciales de ejemplo:

```
"Credentials": {  
  "Secret": "aA1bBB2cCCd3EEE",  
  "UserName": "gl-user-remote"  
}
```


- Para instancias de Linux: para conectarse a una instancia de Linux, SSH requiere un nombre de usuario y una clave privada. Amazon GameLift Servers emite claves privadas RSA y las devuelve en forma de una única cadena, con el carácter de nueva línea (\n) para marcar los saltos de línea. Para poder usar la clave privada, (1) convierta la cadena en un archivo .pem y (2) establezca los permisos para el nuevo archivo. Credenciales de ejemplo devueltas:

```
"Credentials": {
  "Secret": "-----BEGIN RSA PRIVATE KEY-----
nEXAMPLEKEYKCAQEAY7WZhaDsra1W3mRlQtvhwy0RRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
\nvBoU7jLxx92pNHOfnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7i05dSrvC7dQkW2duV5QuUdEQW
\nZ/aNxMniGQE6XAgfwlnXVBwreerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
\nG50TCFe0zf18dqqCP6GzbPaIjiU19xX/az0R9V+tpU0zEL+wmXnZt3/nHPQ5xvD20JH67km6SuPW
\nnoPzev/D8V+x4+bHthfSjR9Y7DvQfjFBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrrqu
\n/uler7vgIn5m71N5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/
ufGxbL1\nmb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/
GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2\nbahyWyJNfjLe4M86yd2YK3V2CmK+X/
B0sShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/
tJWSD9\n81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA30zdXzMqexXVJ1TLZVEH0E7bh1Y9d801ozR
\nnoQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfq1
+1Ip1\nYkriL0DbLX1vRAH+yHPRit2hH0jtUNZh4Axx+cpq09qbUI3+43eEy24B7G/Uh
+GTfbjsXs0xQx/x\nnp9otyVwc7hsQ5TA5PZb
+mvkJ50BEKzet9XcKw0NBYELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
\nayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWV0Eihvm+xTtmaZlSp//lkq75XDwnU
\nWA8gkn603QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Un12ajLivWUt5pbBrKbUC
\nngYBjb0+0Zk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
\nnoMo7yykABY70zd5wQewBQ4AdS1WSX4nGDtsiFxiWi5sKuAAe0CbTosy1s8w8fxoJ5Tz1sdoxNeGs
\nArq6Wv/G16zQuAE9zK9vwwKBgF+09VI/1wJBirsDGz9whVWFFPrTkJNVJZzYt69qezx1sjgFKshy
\nWBhd4xHZtmCqpBP1AymEjr/T01bxyARmXMnIOWIANNXMBG4KGSyl1mzSVAoQ+fqR+cJ3d0dyP11j
\nnjbb0Ed/NY8fr1NDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0i0egLda
\nNWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvTW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
\nVRkAKKKYeGjKpUfVTrW0YFjXkfcrR/V+QFL50ndHAKJXjW7a4ejJLncTzmZSpYzwApc=\n-----END
RSA PRIVATE KEY-----",
  "UserName": "gl-user-remote"
}
```

Al usar la AWS CLI, puede generar automáticamente un .pem archivo al incluir los parámetros --query y --output en su get-instance-access solicitud.

Para establecer permisos en el nuevo archivo .pem, ejecute el siguiente comando:

```
$ chmod 400 MyPrivateKey.pem
```

2. Abra un puerto para la conexión remota. Puede acceder a las instancias en flotas de Amazon GameLift Servers mediante cualquier puerto autorizado en la configuración de la flota. Puede ver la configuración de los puertos de la flota mediante el comando [describe-fleet-port-settings](#).

Recomendamos abrir los puertos para el acceso remoto solo cuando se necesiten y cerrarlos cuando haya terminado. No puede actualizar la configuración de los puertos después de crear una flota y antes de que esté activa. Si no puede continuar, vuelva a crear la flota con la configuración de puertos abierta.

Utilice el comando [update-fleet-port-settings](#) para añadir una configuración de puerto para la conexión remota (como 22 para SSH o 3389 para RDP). Para el valor del rango de IP, especifique las direcciones IP de los dispositivos que desea utilizar para conectarse (convertidos al formato CIDR). Ejemplo:

```
$ 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"
```

En el siguiente ejemplo se abre el puerto 3389 en una flota de 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. Abra un cliente de conexión remoto. Utilice el escritorio remoto para Windows o SSH para instancias Linux. Conéctese a la instancia a través de la dirección IP, la configuración del puerto y las credenciales de acceso.

Ejemplo de SSH:

```
ssh -i MyPrivateKey.pem gl-user-remote@192.0.2.0
```

Visualización de archivos en instancias remotas

Independientemente de si se ha conectado a una instancia de forma remota a través de la consola o la AWS CLI, tiene acceso completo de usuario y administrador a la misma. Esto significa que también tendrá la capacidad de provocar errores y fallos al alojar juegos. Si la instancia aloja juegos con jugadores activos, existe el riesgo de que bloquee sesiones de juego y desconecte a jugadores, así como de interrumpir los procesos de cierre del juego y de provocar errores en los datos y registros de juego guardados.

Busque los siguientes recursos en una instancia de alojamiento:

- Archivos de compilación de juegos. Estos archivos son la compilación del juego que ha cargado en Amazon GameLift Servers. Incluyen uno o más ejecutables, activos y dependencias del servidor de juegos. Los archivos de la compilación del juego se encuentran en un directorio raíz llamado `game`:
 - En Windows: `c:\game`
 - En Linux: `/local/game`
- Archivos de log de juegos. Busque los archivos de registro generados por el servidor de juegos en el directorio raíz `game`, en la ruta de directorio que haya especificado.
- Recursos de alojamiento de Amazon GameLift Servers. El directorio raíz `Whitewater` contiene archivos usados por el servicio de Amazon GameLift Servers para administrar la actividad de alojamiento de juegos. No modifique estos archivos bajo ningún concepto.
- Configuración de tiempos de ejecución. No acceda a la configuración de tiempo de ejecución para instancias individuales. Para realizar cambios en una propiedad de configuración en tiempo de ejecución, actualice la configuración en tiempo de ejecución de la flota (consulte la operación del AWS SDK [UpdateRuntimeConfiguration](#) o la AWS CLI [update-runtime-configuration](#)).
- Datos de la flota. Un archivo JSON contiene información sobre la flota a la que pertenece la instancia, para que la usen los procesos del servidor que se ejecutan en la instancia. El archivo JSON se almacena en la siguiente ubicación:
 - En Windows: `C:\GameMetadata\gamelift-metadata.json`
 - En Linux: `/local/gamemetadata/gamelift-metadata.json`
- Certificados TLS. Si la instancia está en una flota que tiene habilitada la generación de certificados TLS, busque los archivos de certificados, como el certificado, la cadena de certificados, la clave privada y el certificado raíz, en la siguiente ubicación:
 - En Windows: `c:\GameMetadata\Certificates`

- En Linux: `/local/gamemetadata/certificates/`

Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers

La capacidad de alojamiento, medida en instancias, representa la cantidad de sesiones de juego que Amazon GameLift Servers puede alojar simultáneamente y la cantidad de jugadores simultáneos que pueden albergar esas sesiones de juego. Una de las tareas más complicadas del alojamiento de juegos es escalar la capacidad para satisfacer la demanda de los jugadores sin malgastar dinero en recursos innecesarios.

La capacidad se ajusta a nivel de ubicación de la flota. Todas las flotas tienen al menos una ubicación: la AWS región de origen de la flota. Al ver o escalar la capacidad, la información se muestra por ubicación, incluida la región de origen de la flota y cualquier otra ubicación remota.

Puede configurar manualmente el número de instancias que desee mantener, o puede configurar el escalado automático para ajustar la capacidad de forma dinámica a medida que la demanda de los jugadores cambia. Le recomendamos que comience activando la opción de escalado automático basado en objetivos. La función del escalado automático basado en objetivos es mantener suficientes recursos de alojamiento para dar cabida a los jugadores actuales y a algunos más con el fin de administrar los picos inesperados en la demanda de jugadores. Para la mayoría de los juegos, el escalado automático basado en objetivos ofrece una solución de escalado muy eficaz.

Puede realizar la mayoría de las actividades de escalado de la flota con la consola de Amazon GameLift Servers. También puedes usar un AWS SDK o el AWS Command Line Interface (AWS CLI) con la [API de servicio para Amazon GameLift Servers](#).

Temas

- [Para administrar la capacidad de la flota en la consola, realice el siguiente procedimiento:](#)
- [Establecimiento de los límites de capacidad de Amazon GameLift Servers](#)
- [Configurar manualmente la capacidad de una Amazon GameLift Servers flota](#)
- [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#)
- [Gestione el escalado y Amazon GameLift Servers Fleet To/From Zero](#)
- [Escalado de flotas de contenedores de Amazon GameLift Servers](#)

Para administrar la capacidad de la flota en la consola, realice el siguiente procedimiento:

1. Abra la [consola de Amazon GameLift Servers](#).
2. En el panel de navegación, elija Alojamiento y Flotas.
3. En la página Flotas, elija el nombre de una flota activa para abrir la página de detalles de la flota.
4. Elija la pestaña Escalado. En esa pestaña, podrá realizar las siguientes acciones:
 - Consultar las métricas de escalado históricas de toda la flota.
 - Ver y actualizar la configuración de capacidad de cada ubicación de la flota, incluidos los límites de escalado y la configuración de la capacidad actual.
 - Actualizar el escalado automático basado en objetivos, consultar las políticas de escalado automático basado en reglas que se aplican a toda la flota y suspender la actividad de escalado automático en cada ubicación.

Establecimiento de los límites de capacidad de Amazon GameLift Servers

Al escalar la capacidad de alojamiento de una ubicación de Amazon GameLift Servers flota, ya sea de forma manual o automática, tenga en cuenta los límites de escalado de la ubicación. Todas las ubicaciones de flota tienen un límite mínimo y máximo que define el rango permitido de capacidad de la ubicación. De forma predeterminada, los límites de las ubicaciones de la flota establecen en un mínimo de 0 instancias y un máximo de 1. Para poder escalar una ubicación de la flota, ajuste los límites.

Si utiliza el escalado automático, el límite máximo permite a Amazon GameLift Servers escalar verticalmente una ubicación de la flota para satisfacer la demanda de los jugadores, pero evita los costos de alojamiento descontrolados, como los que se originan durante un ataque DDoS. Configura una [CloudWatch alarma de Amazon](#) para que te notifique cuando la capacidad se acerque al límite máximo, de modo que puedas evaluar la situación y ajustarla manualmente según sea necesario. (También puedes [crear una alarma de facturación](#) para controlar AWS los costes). El límite mínimo resulta útil para mantener la disponibilidad del alojamiento, incluso cuando la demanda de jugadores es baja.

Puedes establecer los límites de capacidad para las ubicaciones de una flota en la [Amazon GameLift Serversconsola](#) o mediante el botón AWS Command Line Interface (AWS CLI).

Para establecer límites de capacidad.

Console

1. Abra la [consola de Amazon GameLift Servers](#).
2. En el panel de navegación, elija Alojamiento y Flotas.
3. En la página Flotas, elija el nombre de una flota activa para abrir la página de detalles de la flota.
4. En la pestaña Escalado, en Capacidad de escalado, seleccione una ubicación de la flota y, a continuación, seleccione Editar.
5. En el cuadro de diálogo Editar capacidad de escalado, establezca los recuentos de instancias para Tamaño mínimo, Instancias deseadas y Tamaño máximo.
6. Elija Confirmar.

AWS CLI

1. Compruebe la configuración de la capacidad actual. En una ventana de línea de comandos, usa el [describe-fleet-location-capacity](#) comando con el identificador de flota y la ubicación en la que desees cambiar la capacidad. Este comando devuelve un [FleetCapacity](#) objeto que incluye la configuración de capacidad actual de la ubicación. Determine si los nuevos límites de instancia se ajustarán a la configuración de instancias deseadas actual.

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
  --location <location name>
```

2. Actualice la configuración de límites. En una ventana de línea de comandos, utilice el [update-fleet-capacity](#) comando con los siguientes parámetros. Puede ajustar tanto los límites de instancias como el recuento de instancias deseadas con el mismo comando.

```
--fleet-id <fleet identifier>
--location <location name>
--max-size <maximum capacity for scaling>
--min-size <minimum capacity for scaling>
```

```
--desired-instances <fleet capacity goal>
```

Ejemplo:

```
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 la solicitud se realiza correctamente, Amazon GameLift Servers devuelve el ID de la flota. Si el valor `max-size` o `min-size` nuevo entra en conflicto con el ajuste `desired-instances` actual, Amazon GameLift Servers devuelve un error.

Configurar manualmente la capacidad de una Amazon GameLift Servers flota

Al crear una nueva flota, Amazon GameLift Servers establece automáticamente las instancias deseadas en una instancia en cada ubicación de la flota. A continuación, Amazon GameLift Servers implementa una instancia nueva en cada ubicación. Para cambiar la capacidad de la flota, puede añadir una política de escalado automático basada en objetivos o configurar manualmente la cantidad de instancias que desea para una ubicación.

Establecer la capacidad de una flota manualmente puede resultar útil cuando no necesita aplicar el escalado automático o cuando precisa mantener la capacidad en un nivel específico. La configuración manual de la capacidad solo funciona si no utiliza una política de escalado automático basada en objetivos. Si dispone de una política de escalado automático basada en objetivos, reiniciará automáticamente la capacidad deseada en función de sus reglas de escalado propias.

Puedes configurar la capacidad manualmente en la Amazon GameLift Servers consola o mediante AWS Command Line Interface (AWS CLI). El estado de la flota debe ser Activo.

Suspensión del escalado automático

Puede suspender toda la actividad de escalado automático de cada ubicación de la flota. Con la suspensión del escalado automático, el número deseado de instancias en la ubicación de la flota

permanece igual a menos que se cambie manualmente. Si suspende el escalado automático de una ubicación, afectará a las políticas actuales de la flota y a cualquier política que pueda definir en el futuro.

Para configurar manualmente la capacidad de la flota

Console

1. Abra la [consola de Amazon GameLift Servers](#).
2. En el panel de navegación, elija Alojamiento y Flotas.
3. En la página Flotas, elija el nombre de una flota activa para abrir la página de detalles de la flota.
4. En la pestaña Escalado, en Ubicaciones de escalado automático suspendidas, seleccione las ubicaciones para las que desee suspender el escalado automático y, a continuación, elija Suspend.
5. En Capacidad de escalado, seleccione una ubicación que desee configurar manualmente y, a continuación, elija Editar.
6. En el cuadro de diálogo Editar capacidad de escalado, defina el valor preferido para Instancias deseadas y, a continuación, seleccione Confirmar. Este valor indica a Amazon GameLift Servers cuántas instancias deben mantenerse en estado activo, listas para alojar sesiones de juego.

Amazon GameLift Servers responde a los cambios mediante la implementación de instancias adicionales o el cierre de las innecesarias. A medida que Amazon GameLift Servers completa este proceso, el número de instancias activas en la ubicación cambiará para coincidir con el valor de instancias actualizado deseado. Este proceso puede tardar cierto tiempo.

AWS CLI

1. Compruebe la configuración de la capacidad actual. En una ventana de línea de comandos, utilice el [describe-fleet-location-capacity](#) comando con el identificador de flota y la ubicación en la que desee cambiar la capacidad. Este comando devuelve un [FleetCapacity](#) objeto que incluye la configuración de capacidad actual de la ubicación. Determine si los límites de instancia se ajustarán a la nueva configuración de instancias deseada.

```
aws gamelift describe-fleet-location-capacity \
  --fleet-id <fleet identifier> \
```

```
--location <location name>
```

2. Actualice la capacidad deseada. Utilice el [update-fleet-capacity](#) comando con el identificador de flota, la ubicación y un nuevo valor para las instancias deseadas. Si este valor queda fuera del rango de límites actual, puede ajustar valores de ajuste de límites en el mismo comando.

```
--fleet-id <fleet identifier>
--location <location name>
--desired-instances <fleet capacity as an integer>
--max-size <maximum capacity> [Optional]
--min-size <minimum capacity> [Optional]
```

Ejemplo:

```
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 la solicitud se realiza correctamente, Amazon GameLift Servers devuelve el ID de la flota. Si la nueva configuración de instancias deseadas está fuera de los límites máximo y mínimo, Amazon GameLift Servers devuelve un error.

Escalado automático de la capacidad de la flota con Amazon GameLift Servers

Utilice el escalado automático de Amazon GameLift Servers para escalar de forma dinámica la capacidad de la flota en respuesta a la actividad del servidor de juegos. A medida que los jugadores se incorporen e inicien sesiones de juego, el escalado automático puede añadir nuevas instancias; y, a medida que la demanda de los jugadores se reduzca, el escalado automático irá cerrando las instancias innecesarias. El escalado automático es una forma eficaz de minimizar los recursos y los costos de alojamiento y, a la vez, de facilitar al jugador una experiencia rápida y fluida.

Para usar el escalado automático, debe crear políticas de escalado que indiquen a Amazon GameLift Servers cuándo escalar verticalmente la capacidad. Existen dos tipos de políticas de escalado: las basadas en objetivos y las basadas en reglas. El enfoque basado en objetivos (el seguimiento de objetivos) es una solución completa. Lo recomendamos como la opción más sencilla y eficaz. Las políticas de escalado basadas en reglas requieren que se defina cada aspecto del proceso de toma de decisiones de escalado automático, que son útiles para abordar cuestiones específicas. Esta solución funciona mejor como complemento del escalado automático basado en objetivos.

Puede administrar el escalado automático basado en objetivos mediante la Amazon GameLift Servers consola, el AWS Command Line Interface (AWS CLI) o un AWS SDK. Puede administrar el escalado automático basado en reglas mediante el uso del SDK AWS CLI o únicamente con un AWS SDK, aunque puede ver las políticas de escalado basado en reglas en la consola.

Temas

- [Escalado automático basado en objetivos](#)
- [Escalado automático con políticas basadas en reglas](#)

Escalado automático basado en objetivos

El escalado automático basado en objetivos de Amazon GameLift Servers ajusta los niveles de capacidad en función de la métrica `PercentAvailableGameSessions` de la flota. Esta métrica representa el búfer disponible de la flota frente a aumentos repentinos de la demanda de los jugadores.

El principal motivo para mantener un búfer de capacidad es el tiempo de espera de los jugadores. Cuando las ranuras de sesiones de juego están listas y en espera, se introducen los nuevos jugadores en las sesiones de juego en cuestión de segundos. Si no hay recursos disponibles, los jugadores deben esperar a que las sesiones de juego existentes terminen o a que queden disponibles nuevos recursos. El inicio de nuevas instancias y procesos del servidor puede tardar unos minutos.

Al configurar el escalado automático basado en objetivos, especifique el tamaño del búfer que desea que mantenga la flota. Dado que `PercentAvailableGameSessions` mide el porcentaje de recursos disponibles, el tamaño de búfer real es un porcentaje de la capacidad total de la flota. Amazon GameLift Servers añade o elimina instancias para mantener el tamaño de búfer objetivo. Con un búfer grande, reducirá el tiempo de espera, pero también pagará por recursos adicionales que podrían no utilizarse. Si los jugadores tienen más tolerancia a los tiempos de espera, puede reducir los costos estableciendo un búfer pequeño.

Para configurar el escalado automático basado en objetivos, realice el siguiente procedimiento:

Console

1. Abra la [consola de Amazon GameLift Servers](#).
2. En el panel de navegación, elija Alojamiento y Flotas.
3. En la página Flotas, elija el nombre de una flota activa para abrir la página de detalles de la flota.
4. Elija la pestaña Escalado. Esta pestaña muestra las métricas históricas de escalado de la flota y contiene controles para ajustar la configuración de escalado actual.
5. En Capacidad de escalado, compruebe que los límites de Tamaño mínimo y Tamaño máximo sean los adecuados para la flota. Con el escalado automático habilitado, la capacidad se ajustará entre esos dos límites.
6. En Política de escalado automático basada en objetivos, elija Editar.
7. En el cuadro de diálogo Editar la política de escalado automático basada en objetivos, en Porcentaje de sesiones de juego disponibles, establezca el porcentaje que desea mantener y, a continuación, elija Confirmar. Una vez que confirme la configuración, Amazon GameLift Servers añadirá una nueva política basada en objetivos en Política de escalado automático basada en objetivos.

AWS CLI

1. Establezca los límites de capacidad. Establezca los valores límite mediante el [update-fleet-capacity](#) comando. Para obtener más información, consulte [Establecimiento de los límites de capacidad de Amazon GameLift Servers](#).
2. Cree una política nueva. Abra una ventana de línea de comandos y utilice el [put-scaling-policy](#) comando con la configuración de los parámetros de la política. Para actualizar una política existente, especifique el nombre de la política y proporcione una versión completa de la política actualizada.

```
--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>
```

Ejemplo:

```
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"
```

Escalado automático con políticas basadas en reglas

Las políticas de escalado basadas en reglas de Amazon GameLift Servers ofrecen un control detallado al escalar automáticamente la capacidad de una flota en respuesta a la actividad de los jugadores. Para cada política, puede vincular el escalado a una métrica de la flota de entre varias, identificar un punto de desencadenamiento y personalizar el evento de ampliación o reducción de respuesta. Las políticas basadas en reglas son útiles para complementar el [escalado basado en destino](#) para afrontar circunstancias especiales.

Una política basada en reglas indica lo siguiente: “Si una métrica de una flota alcanza o supera un valor umbral durante un periodo determinado, se debe cambiar la capacidad de la flota en una cantidad determinada”. En este tema se describe la sintaxis utilizada para construir una instrucción de política y proporciona ayuda para crear y administrar sus políticas basadas en reglas.

Administración de políticas basadas en reglas

Cree, actualice o elimine políticas basadas en reglas mediante un AWS SDK o el AWS Command Line Interface (AWS CLI) con la [API de servicio](#) para Amazon GameLift Servers. Puede ver todas las políticas activas en la consola de Amazon GameLift Servers.

Para detener temporalmente todas las políticas de escalado de una flota, usa el AWS CLI comando. [stop-fleet-actions](#)

Para crear o actualizar una política de escalado basada en reglas (AWS CLI), realice el siguiente procedimiento:

1. Establezca los límites de capacidad. Establezca uno o ambos valores límite mediante el [update-fleet-capacity](#) comando. Para obtener más información, consulte [Establecimiento de los límites de capacidad de Amazon GameLift Servers](#).

2. Cree una política nueva. Abra una ventana de línea de comandos y utilice el [put-scaling-policy](#) comando con la configuración de los parámetros de la política. Para actualizar una política existente, especifique el nombre de la política y proporcione una versión completa de la política actualizada.

```
--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>
```

Ejemplo:

```
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
```

Para eliminar una política de escalado basada en reglas mediante la AWS CLI, realice el siguiente procedimiento:

- Abra una ventana de línea de comandos y utilice el [delete-scaling-policy](#) comando con el identificador de la flota y el nombre de la política.

Ejemplo:

```
aws gamelift delete-scaling-policy \
  --fleet-id fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
  --name "Scale up when AGS<50"
```

Sintaxis de las reglas de escalado automático

Para construir una instrucción de política de escalado automático basado en reglas, especifique seis variables:

Si *<metric name>* permanece así *<comparison operator>* *<threshold value>**<evaluation period>*, cambie la capacidad de la flota utilizando *<adjustment type>* to/by *<adjustment value>*.

Por ejemplo, esta instrucción de la política inicia un evento de escalado vertical cuando la capacidad adicional de la flota es inferior a lo que se necesita para administrar 50 nuevas sesiones de juego:

Si AvailableGameSessions se mantiene en less than 50 durante 10 minutes, entonces se debe cambiar la capacidad de la flota utilizando ChangeInCapacity en 1 instances.

Nombre de métrica

Para iniciar un evento de escalado, vincule una política de escalado automático a una de las siguientes métricas específicas de la flota. Consulte [Métricas de flotas de Amazon GameLift Servers](#) para obtener descripciones completas de las métricas.

- Activación de sesiones de juego
- Sesiones de juego activas
- Sesiones de juego disponibles
- Porcentaje de sesiones de juego disponibles
- Instancias activas
- Sesiones de jugador disponibles
- Sesiones de jugador actuales
- Instancias inactivas
- Porcentaje de instancias inactivas

Si la flota se encuentra en una cola de sesión de juego, puede utilizar las siguientes métricas:

- Profundidad de la cola: número de solicitudes de sesiones de juego pendientes para las que esta flota es la mejor ubicación de alojamiento disponible.
- Tiempo de espera: tiempo de espera específico de la flota. Periodo de tiempo que ha estado esperando la solicitud de sesión de juego más antigua hasta ser completada. El tiempo de espera de una flota es igual al tiempo en cola de la solicitud actual más vieja.

Operador de comparación

Indica a Amazon GameLift Servers cómo debe comparar los datos de las métricas con el valor de umbral. Los operadores de comparación válidos incluyen mayor que (>), menor que (<), mayor o igual que (>=) y menor o igual que (<=).

Valor umbral

Cuando el valor especificado de la métrica alcanza o supera el valor umbral, inicia un evento de escalado. Este valor siempre es un número entero positivo.

Periodo de evaluación

La métrica debe alcanzar o superar el valor umbral durante la totalidad del periodo de evaluación antes de iniciar un evento de escalado. La longitud del periodo de evaluación es consecutiva: si la métrica cae por debajo del umbral, el periodo de evaluación se inicia de nuevo.

Tipo y valor de ajuste

Este conjunto de variables opera conjuntamente para especificar el modo en que Amazon GameLift Servers debe ajustar la capacidad de la flota cuando se inicia un evento de escalado. Puede elegir de entre tres posibles tipos de ajuste:

- **Cambio de capacidad:** permite aumentar o disminuir la capacidad actual según el número de instancias especificado. Especifique el valor de ajuste según el número de instancias que desea añadir o quitar de la flota. Los valores positivos añaden instancias, mientras que los valores negativos eliminan instancias. Por ejemplo, un valor de "-10" reducirá verticalmente la flota en 10 instancias, independientemente del tamaño total de la flota.
- **Cambio porcentual de la capacidad:** permite aumentar o disminuir la capacidad actual según el porcentaje especificado. Especifique el valor de ajuste en función del porcentaje en que desea aumentar o reducir la capacidad de la flota. Los valores positivos añaden instancias, mientras que los valores negativos eliminan instancias. Por ejemplo, en el caso de una flota con 50 instancias, un cambio porcentual de "20" añadirá diez instancias a la flota.
- **Capacidad exacta:** permite aumentar o reducir la capacidad actual a un valor específico. Especifique el valor de ajuste en el número de instancias exacto que desea mantener en la flota.

Consejos para el escalado automático basado en reglas

Las siguientes sugerencias pueden ayudarle a sacar el máximo provecho del escalado automático con políticas basadas en reglas.

Utilice varias políticas

Puede emplear varias políticas de escalado automático para una flota al mismo tiempo. La situación más habitual consiste en tener una política basada en destino que administre la mayoría de las necesidades de escalado y utilizar políticas basadas en reglas para administrar casos extremos. No existen límites en cuanto al uso de varias políticas.

Con varias políticas, cada política se comporta de forma independiente. No se puede controlar la secuencia de eventos de escalado. Por ejemplo, si dispone de varias políticas de escalado vertical, es posible que la actividad de los jugadores inicie varios eventos de escalado de forma simultánea. Evite las políticas que se inician entre sí. Por ejemplo, puede crear un bucle infinito si crea políticas de escalado y reducción verticales que establecen una capacidad superior al umbral de cada una.

Defina una capacidad máxima y mínima

Cada flota tiene un límite de capacidad máxima y mínima. Esta característica es especialmente importante cuando se utiliza el escalado automático. El escalado automático nunca definirá un valor de la capacidad que no esté dentro de ese rango. De forma predeterminada, las flotas que se acaban de crear tienen un mínimo de 0 y un máximo de 1. Para que la política de escalado automático afecte a la capacidad según lo previsto, aumente el valor máximo.

La capacidad de la flota también está limitada por los límites del tipo de instancia de la flota y por las cuotas de servicio de su Cuenta de AWS empresa. No puede establecer un valor mínimo y uno máximo que se encuentren fuera de esos límites y de las cuotas de cuenta.

Realice el seguimiento de las métricas después de un cambio en la capacidad

Después de cambiar la capacidad en respuesta a una política de escalado automático, Amazon GameLift Servers espera 10 minutos antes de responder a los desencadenadores de dicha política. Esta espera concede a Amazon GameLift Servers tiempo para añadir las instancias nuevas, lanzar los servidores de juegos, conectar a los jugadores y empezar a recopilar datos de las instancias nuevas. Durante este tiempo, Amazon GameLift Servers evalúa la política en función de la métrica y realiza el seguimiento del periodo de evaluación de la política, que se reinicia después de que se produzca un evento de escalado. Esto significa que una política de escalado podría iniciar otro evento de escalado en cuanto acabe el tiempo de espera.

No hay tiempo de espera entre los eventos de escalado que inician diferentes políticas de escalado automático.

Gestione el escalado y Amazon GameLift Servers Fleet To/From Zero

Amazon GameLift Servers admite el escalado automático desde y hacia cero instancias en función de la actividad de la sesión de juego. Esta opción de capacidad gestionada permite que las ubicaciones de tu flota se amplíen hasta cero instancias tras un período definido sin actividad de juego y se amplíen automáticamente cuando se soliciten sesiones de juego.

El escalamiento desde y hacia cero instancias ofrece varias ventajas:

- Optimización de costos: elimine los costos de procesamiento durante los períodos de inactividad al ejecutar cero instancias cuando no haya actividad en la sesión de juego.
- Ampliación automática y reactiva: las ubicaciones de las flotas se escalan automáticamente a una sola instancia cuando se solicita una sesión de juego, lo que elimina la necesidad de intervención manual.
- Gestión simplificada: no es necesario ajustar manualmente la capacidad de la flota a to/from cero en función de la demanda prevista de los jugadores o de las necesidades de desarrollo.

Al activar Scale To/From Zero en una flota, Amazon GameLift Servers monitoriza la actividad de las sesiones de juego y ajusta automáticamente la capacidad de la flota:

- Escalar a cero: después de un período configurado sin actividad en las sesiones de juego, se Amazon GameLift Servers amplía la ubicación de la flota a cero instancias.
- Escalamiento horizontal desde cero: cuando se recibe una solicitud de creación de sesión de juego, reduce la ubicación de la flota a una instancia, lo que permite reanudar el Amazon GameLift Servers escalado automático.
- Ampliación continua: tras la ampliación horizontal, la flota vuelve a utilizar políticas de escalado automático configuradas para gestionar la capacidad.

Comportamiento de escalado

Amazon GameLift Servers comienza el proceso de escalado de una ubicación de flota una vez transcurrido el período de inactividad configurado sin actividad en la sesión de juego. Se define como un período en el que:

- No hay sesiones de juego activas en la ubicación de la flota.

- No se ha solicitado la creación de nuevas sesiones de juego en la ubicación de la Flota.

Durante la ampliación, Amazon GameLift Servers estableceremos en cero la capacidad mínima y deseada para la ubicación de la flota, y la ampliaremos rápidamente para ahorrar costes.

Comportamiento de escalado

Cuando se recibe una solicitud de creación de una sesión de juego cuando la ubicación de la flota está en cero instancias:

- Amazon GameLift Servers inicia inmediatamente la escalabilidad horizontal de una instancia.
- El intento de ubicación de la sesión de juego puede continuar con otras flotas o ubicaciones de flotas, según la configuración de las colas (si se utilizan).

Note

El aprovisionamiento y la inicialización de las instancias requieren tiempo para escalar desde cero. Los jugadores pueden experimentar tiempos de espera más largos para la primera sesión de juego después de un período de inactividad. Por este motivo, es mejor combinar esta función con las colas de flotas and/or en varias ubicaciones.

Configuración de Scale Zero To/From

Scaling To/From Zero se configura actualizando una flota existente.

Console

1. Abra la [consola de Amazon GameLift Servers](#).
2. En el panel de navegación, elija Alojamiento y Flotas.
3. En la página Flotas, elija el nombre de una flota activa para abrir la página de detalles de la flota.
4. En Scaling, seleccione cada ubicación para la que desee configurar Scale to/from Zero y, a continuación, elija Editar.
5. En el cuadro de diálogo Editar la capacidad de escalado, seleccione «Automático» para la estrategia de capacidad mínima, establezca su valor preferido en minutos para Establecer la capacidad mínima en 0 después y, a continuación, elija Confirmar.

Amazon GameLift Serversa continuación, ampliará las ubicaciones de la flota seleccionada a 0 instancias cuando no haya habido actividad en la sesión de juego durante el tiempo configurado. A continuación, cuando se solicite una sesión de juego en esta ubicación, Amazon GameLift Servers ampliaremos una instancia lo más rápido posible. Este proceso llevará un poco de tiempo.

AWS CLI

- Configure la escala to/from cero. En una ventana de línea de comandos, utilice el [update-fleet-capacity](#) comando con el identificador de flota, la ubicación y la configuración de capacidad gestionada para configurar la escala to/from cero.

```
aws gamelift update-fleet-capacity \
    --fleet-id <fleet identifier> \
    --location <location name> \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

Ejemplo:

```
aws gamelift update-fleet-capacity \
    --fleet-id
    fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa \
    --location us-west-2 \
    --desired-instances 5 \
    --max-size 10 \
    --managed-capacity-configuration
    ScaleInAfterInactivityMinutes=60,ZeroCapacityStrategy=SCALE_TO_AND_FROM_ZERO
```

Si la solicitud se realiza correctamente, Amazon GameLift Servers devuelve la configuración de capacidad actualizada. Amazon GameLift Serversa continuación, ampliará las ubicaciones de la flota seleccionada a 0 instancias cuando no haya habido actividad en la sesión de juego durante el tiempo configurado. A continuación, cuando se solicite una sesión de juego en esta ubicación, Amazon GameLift Servers ampliaremos una instancia lo más rápido posible. Este proceso llevará un poco de tiempo.

Prácticas recomendadas

Tenga en cuenta las siguientes recomendaciones cuando utilice Scale To/From Zero:

- Establezca los períodos de inactividad adecuados: equilibre los ahorros de costos con la frecuencia de los ciclos de escalamiento y escalamiento horizontal. Un menor tiempo de espera para escalar a cero puede maximizar los ahorros, pero se traduciría en arranques en frío más frecuentes.
- Úselo con cargas de trabajo predecibles: Scale To/From Zero funciona mejor para juegos con períodos claros de inactividad, como development/test entornos o juegos con distintos horarios de baja actividad.
- Supervise CloudWatch las métricas: realice un seguimiento de los eventos de escalado de la flota y de los tiempos de duración de las sesiones de juego para optimizar su configuración.
- Combínelo con políticas de escalado: utilice Scale To/From Zero junto con el escalado automático basado en objetivos o en reglas para una administración integral de la capacidad.

Escalado de flotas de contenedores de Amazon GameLift Servers

Una de las tareas más complicadas del alojamiento de juegos es escalar la capacidad para satisfacer la demanda de los jugadores sin malgastar dinero en recursos innecesarios. En una flota de contenedores administrada, la capacidad de la flota se escala añadiendo o eliminando instancias de flota.

Al crear una flota nueva, Amazon GameLift Servers establece la capacidad deseada de la flota en una instancia e implementa una instancia en la región de origen de la flota. En el caso de las flotas con múltiples ubicaciones, Amazon GameLift Servers implementa una instancia en la región de origen y en cada ubicación remota. Una vez que el estado de la flota pasa a ser ACTIVE, puede aumentar o disminuir la capacidad deseada para reducirla verticalmente.

Puede utilizar las características de escalado de Amazon GameLift Servers escalado para cambiar la capacidad manualmente o configurar el escalado automático en función de la demanda de los jugadores:

- Configuración del escalado automático con seguimiento de destino. Consulte [Escalado automático basado en objetivos](#).
- Cambio manual la capacidad de la flota. Consulte [Configurar manualmente la capacidad de una Amazon GameLift Servers flota](#).

A la hora de escalar una flota de contenedores, tenga en cuenta cómo la adición o eliminación de instancias afecta a la capacidad de la flota para alojar sesiones de juego y jugadores.

- Sesiones de juego por instancia
 - Cada proceso del servidor de juegos que se ejecuta en una instancia representa la capacidad para alojar una sesión de juego.
 - Utilice esta fórmula para calcular el número de sesiones de juego que se ejecutan simultáneamente en una instancia de flota de contenedores:

```
[Game sessions per instance] = [# of game server processes per game server container] * [# of game server container groups per instance]
```

Si la arquitectura de contenedores ejecuta un proceso de servidor de juegos simultáneamente en el contenedor del servidor de juegos, las sesiones de juego por instancia equivalen al número de grupos de contenedores de servidor de juegos por instancia.

- En el caso de los grupos de contenedores de servidor de juegos por instancia, realice una llamada a [DescribeContainerFleet](#) para obtener el valor `GameServerContainerGroupsPerInstance` o `MaximumGameServerContainerGroupsPerInstance`.
- Jugadores por instancia
 - Usted decide el número de espacios de jugadores que desea permitir en cada sesión de juego. En función de cómo administre su solución de alojamiento la ubicación de las sesiones de juego, puede definir los jugadores por sesión de juego en la configuración de emparejamiento o en las llamadas para iniciar la ubicación de una sesión de juego.
 - Utilice esta fórmula para calcular el número de jugadores que pueden jugar al juego simultáneamente en una instancia de flota de contenedores:

```
[Players per instance] = [# of game sessions per instance] * [# of player slots per game session]
```

Para obtener la capacidad total actual de una flota de contenedores, realice una llamada a [DescribeFleetCapacity](#) o a [DescribeFleetLocation Capacity](#) para obtener el número de grupos de contenedores de servidor de juegos de la flota. Los grupos activos son aquellos que están alojando sesiones de juego. Los grupos inactivos están listos para alojar una nueva sesión de juego.

Multiplique estos valores por el número de procesos de servidor por grupo de contenedores de servidor de juegos.

Supervisión de Amazon GameLift Servers

La supervisión es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de Amazon GameLift Servers sus demás AWS soluciones. Amazon GameLift Servers emplea las métricas de tres maneras: para supervisar el estado del sistema y configurar alertas, para realizar el seguimiento del uso y del desempeño del servidor de juegos y para administrar la capacidad mediante el escalado automático o manual.

AWS proporciona las siguientes herramientas de monitoreo para observar Amazon GameLift Servers, informar cuando algo anda mal y tomar medidas automáticas cuando sea apropiado:

- **Consola de Amazon GameLift Servers:** utilice la interfaz gráfica para administrar sus recursos de Amazon GameLift Servers y monitorizar la actividad de alojamiento de juegos.
- **Métricas de telemetría del servidor:** Amazon GameLift Servers le permite emitir métricas personalizadas directamente desde sus servidores de juegos mediante la integración de SDK y complementos. Puede definir y monitorizar sus propias métricas específicas del juego, así como los datos integrados de rendimiento, red, memoria y temporización. Todas las métricas pueden publicarse en [Amazon Managed Service para Prometheus](#) y supervisarse en Amazon Grafana a través de paneles totalmente personalizables. También puedes publicar en Amazon CloudWatch para integrarlo con otros AWS servicios. El sistema de telemetría es totalmente personalizable: puede crear consultas personalizadas de Prometheus para obtener otras métricas aparte de las incorporadas. Para obtener instrucciones de implementación específicas de su pila de tecnologías, consulte [the section called “Supervisión con métricas de telemetría de servidor”](#).
- **Amazon CloudWatch:** puede monitorear Amazon GameLift Servers las métricas en tiempo real, así como las métricas de otros AWS recursos y aplicaciones que ejecuta en AWS los servicios. Amazon CloudWatch ofrece un conjunto de funciones de monitoreo, que incluyen herramientas para crear paneles personalizados y la capacidad de configurar alarmas que notifican o toman medidas cuando una métrica alcanza un umbral específico.
- **AWS CloudTrail—** captura todas las llamadas a la API y los eventos relacionados realizados por o en nombre de tu AWS cuenta Amazon GameLift Servers y otros AWS servicios. Los datos se entregan como archivos de registro en el bucket de Amazon S3 que especifique. Puedes identificar qué usuarios y cuentas llamaron AWS, la dirección IP de origen desde la que se realizaron las llamadas y cuándo se produjeron.
- **Registros de sesiones de juego:** puede enviar mensajes de servidor personalizados para sus sesiones de juego a archivos de registro almacenados en Amazon S3.

Comparación de métricas de diferentes fuentes de supervisión

Amazon GameLift Servers proporciona métricas a través de tres fuentes principales: las métricas de actividad de la flota de la Amazon GameLift Servers consola, las métricas de telemetría del servidor y las métricas de Amazon CloudWatch Amazon GameLift Servers. Comprender el solapamiento y las capacidades únicas de cada fuente le será de ayuda a la hora de elegir el enfoque de supervisión adecuado según sus necesidades.

Disponibilidad de las métricas por fuente

En las siguientes tablas se muestran las métricas que están disponibles en las diferentes fuentes de supervisión, organizadas por categoría de métricas.

Métricas de la instancia

Métricas de instancia para la supervisión de la capacidad y el estado de la flota:

Disponibilidad de las métricas de instancia

Métrica	Consola	CloudWatch	Telemetría
Instancias activas	✓	✓	✓
Instancias inactivas	✓	✓	✓*
Porcentaje de instancias inactivas	✓	✓	✓*
Instancias deseadas	✓	✓	
Instancias máximas	✓	✓	
Instancias mínimas	✓	✓	
Utilización de la CPU		✓	✓
E/S de red		✓	✓
Disk/Storage read/write		✓	✓
Interrupciones de instancia	✓	✓	
Instancias recicladas	✓	✓	

Métrica	Consola	CloudWatch	Telemetría
Instancias defectuosas reemplazadas		✓	

*Disponible mediante consultas personalizadas de Prometheus con datos de telemetría.

Métricas del servidor de juegos

Métricas de procesos del servidor y sesiones de juego:

Disponibilidad de las métricas del servidor de juegos

Métrica	Consola	CloudWatch	Telemetría
Sesiones de juego activas	✓	✓	✓*
Activación de sesiones de juego	✓	✓	
Sesiones de juego disponibles	✓	✓	
Porcentaje de sesiones de juego disponibles	✓	✓	
Sesiones de juego activables simultáneas	✓	✓	
Interrupciones de sesiones de juego	✓	✓	
Procesos activos del servidor			✓
Servidores de juegos en buen estado			✓
Sesiones de juego bloqueadas			✓

*Disponible mediante consultas personalizadas de Prometheus con datos de telemetría.

Métricas de uso de los jugadores

Métricas de sesiones de jugador y usuarios simultáneos:

Disponibilidad de las métricas de uso de los jugadores

Métrica	Consola	CloudWatch	Telemetría
Sesiones de jugador actuales	✓		
Activaciones de sesiones de jugador	✓		
Usuarios simultáneos globales (CCU)			✓
Usuarios simultáneos por ubicación (CCU)			✓
Capacidad de ubicación			✓

Métricas de flotas de contenedores

Métricas específicas de contenedores (disponibles solo CloudWatch para flotas de contenedores):

Disponibilidad de las métricas de flotas de contenedores

Métrica	Consola	CloudWatch	Telemetría
Grupos de contenedores activos		✓	✓
Grupos de contenedores inactivos		✓	✓
CPU/Memory Utilización de contenedores		✓	✓
Tráfico de red del contenedor		✓	✓
Operaciones de almacenamiento del contenedor		✓	✓

Elección de la fuente de supervisión adecuada

Seleccione su enfoque de supervisión en función de sus necesidades específicas:

- Utilice las métricas de telemetría del servidor para supervisar datos exhaustivos de rendimiento de sus servidores de juegos y emita métricas personalizadas específicas del juego. Estas métricas ofrecen información detallada sobre el rendimiento del servidor, la actividad de la red, el uso de la

memoria y los datos de temporización de todos los motores de juegos y servidores SDKs. Puede definir métricas personalizadas de eventos de juego, rendimiento de la lógica empresarial y puntos de datos específicos de aplicación. Todos los paneles son totalmente personalizables y puede crear consultas personalizadas de Prometheus para obtener métricas adicionales a partir de los datos recopilados. Para obtener más información, consulte [the section called “Supervisión con métricas de telemetría de servidor”](#).

- Utilice la consola de Amazon GameLift Servers para administrar flotas, planificar la capacidad y supervisar las operaciones en general. La consola proporciona una vista integrada del estado de la flota y de la actividad de los jugadores. Para obtener más información, consulte [the section called “Administración de los recursos de alojamiento de juegos”](#).
- Usa Amazon CloudWatch para la supervisión, las alertas y la integración automatizadas con otros AWS servicios. CloudWatch permite paneles personalizados y automatización basada en alarmas. Para obtener más información, consulte [the section called “Supervise con CloudWatch”](#).
- Utilice [Amazon Managed Service para Prometheus](#) para recopilar y almacenar métricas de alto rendimiento con las capacidades de consulta de PromQL. Prometheus proporciona almacenamiento de datos de series temporales escalable para las métricas de telemetría del servidor.
- Utilice [Amazon Managed Grafana](#) para disfrutar de funciones de visualización avanzadas y paneles de control totalmente personalizables. Grafana proporciona GameLift paneles prediseñados que puede personalizar y ampliar, además admite la creación de paneles completamente personalizados con múltiples fuentes de datos. Puede crear consultas y visualizaciones personalizadas para monitorizar cualquier métrica que sea importante para el juego.
- Utilice varias fuentes para una supervisión exhaustiva. Combine la supervisión de la consola, las métricas de telemetría del servidor y las alertas automatizadas para obtener una visibilidad completa de la infraestructura de alojamiento de juegos.

Para habilitar capacidades de supervisión adicionales, también puede utilizar:

- [the section called “Registro de llamadas a la API”](#): permite monitorizar las llamadas a la API y los eventos relacionados con fines de auditoría y conformidad.
- [the section called “Registro de mensajes del servidor”](#): permite capturar mensajes personalizados del servidor y registros de sesiones de juego.

Temas

- [the section called “Administración de los recursos de alojamiento de juegos”](#)
- [the section called “Supervise con CloudWatch”](#)
- [the section called “Supervisión con métricas de telemetría de servidor”](#)
- [the section called “Registro de llamadas a la API”](#)
- [the section called “Registro de mensajes del servidor”](#)

Monitoriza Amazon GameLift Servers con Amazon CloudWatch

Puedes monitorizar Amazon GameLift Servers con Amazon CloudWatch, un AWS servicio que recopila datos sin procesar y los procesa para convertirlos en métricas legibles prácticamente en tiempo real. Estas estadísticas se conservan durante quince meses para proporcionar una perspectiva histórica sobre el rendimiento del alojamiento del servidor de juegos con Amazon GameLift Servers. Puede establecer alarmas que vigilen determinados umbrales y enviar notificaciones o realizar acciones cuando se alcancen dichos umbrales. Para obtener más información, consulta la [Guía del CloudWatch usuario de Amazon](#).

Para ofrecer una observabilidad mejorada con métricas de telemetría detalladas, Amazon GameLift Servers también admite la supervisión mediante [Amazon Managed Service para Prometheus](#) y [Amazon Managed Grafana](#). Estas métricas de telemetría proporcionan datos detallados sobre el rendimiento del servidor, que incluyen:

- Tiempos del servidor: tiempo diferencial del servidor (constancia de la frecuencia de graduación del servidor), frecuencia de graduación del servidor (actualizaciones por segundo), tiempo de graduación del servidor (tiempo de procesamiento por ciclo) y tiempo de graduación del mundo del juego del servidor (tiempo de actualización del mundo del juego)
- Métricas de red: conexiones (conexiones de red totales), red I/O en bytes y paquetes (estadísticas de transferencia de datos) y pérdida de paquetes (porcentaje de errores de transmisión)
- Métricas de memoria: uso de memoria en unidades (consumo total de memoria) y uso de memoria física como porcentaje (uso de memoria)
- Uso de CPU: porcentaje de uso de la CPU (uso de recursos por proceso del servidor de juegos)

Para ver las guías de implementación específicas de su SDK o complemento, consulte [the section called “Supervisión con métricas de telemetría de servidor”](#).

En las siguientes tablas se muestran las métricas y dimensiones de Amazon GameLift Servers. Todas las métricas disponibles en también CloudWatch están disponibles en la Amazon GameLift Servers consola, que proporciona los datos en forma de un conjunto de gráficos personalizables. Para acceder a CloudWatch las métricas de tus juegos, usa la Consola de administración de AWS AWS CLI, la o la CloudWatch API.

Si una métrica no tiene una ubicación, utiliza la ubicación de inicio.

Dimensiones de las métricas de Amazon GameLift Servers

Amazon GameLift Servers permite filtrar métricas por las siguientes dimensiones.

Dimensión	Description (Descripción)
Location	Permite filtrar las métricas para una ubicación de implementación de la flota. Si una métrica no tiene una ubicación, utiliza la ubicación de inicio.
FleetId	Filtrar métricas para una sola flota. Esta dimensión se puede usar con todas las métricas de instancias, procesos de servidor, sesiones de juego y sesiones de jugador.
MetricGroup	Filtrar métricas para una colección de flotas. Añada una flota a un grupo de métricas añadiendo el nombre del grupo de métricas a los atributos de la flota (consulte UpdateFleetAttributes()). Esta dimensión se puede usar con todas las métricas de instancias, procesos de servidor, sesiones de juego y sesiones de jugador.
QueueName	Filtrar métricas para una sola cola. Esta dimensión se usa únicamente con métricas para las colas de sesiones de juego.
ConfigurationName	Filtrar métricas para una única configuración de emparejamiento. Esta dimensión se usa con métricas de configuraciones de emparejamiento.

Dimensión	Description (Descripción)
ConfigurationName-RuleName	Filtrar métricas para una intersección de una configuración de emparejamiento y una regla de emparejamiento. Esta dimensión solo se usa con métricas de reglas de emparejamiento.
InstanceType	Filtrar métricas para una designación de tipo de instancia EC2, como "c4.large". Esta dimensión se usa con métricas de instancias de spot.
OperatingSystem	Filtrar métricas para el sistema operativo de una instancia. Esta dimensión se utiliza con métricas de instancias de spot.
GameServerGroup	Filtrar métricas de FleetIQ para un grupo de servidores de juegos.
ContainerGroupType	Filtre las métricas de la flota de contenedores para distinguir entre GameServer ContainerGroup las métricas sobre la ubicación de la flota y PerInstance ContainerGroup las métricas sobre la ubicación de la flota. Esta dimensión se incluye para determinar las métricas de la flota de contenedores (por ejemplo ContainerNetworkIn o ContainerStorageReadBytes) y le permite distinguir entre las métricas de distintos tipos de grupos de contenedores.

Métricas de flotas de Amazon GameLift Servers

El espacio de nombres `AmazonGameLift` incluye las siguientes métricas relativas a la actividad de una flota o grupo de flotas. Las flotas se utilizan con una solución de Amazon GameLift Servers administrada. El Amazon GameLift Servers servicio envía métricas a CloudWatch cada minuto.

instancias

Métrica	Description (Descripción)
ActiveInstances	Instancias con el estado ACTIVE, lo que indica que se trata de procesos del servidor activos. El recuento incluye las instancias inactivas y las que alojan una o varias sesiones de juego. Esta métrica mide la capacidad total actual de las instancias. Se puede usar con el escalado automático. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
DesiredInstances	Número de destino de instancias activas que utiliza Amazon GameLift Servers para mantener la flota. Con el escalado automático, este valor se determina en función de las políticas de escalado aplicadas actualmente. Sin el escalado automático, este valor se establece manualmente. Esta métrica no está disponible cuando se consultan datos de grupos de métricas de flotas. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
IdleInstances	Instancias activas que alojan actualmente cero (0) sesiones de juego. Esta métrica mide la capacidad disponible pero sin usar. Se puede usar con el escalado automático.

Métrica	Description (Descripción)
	Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
MaxInstances	Número máximo de instancias permitidas para la flota. El número máximo de instancias de una flota determina el límite máximo de capacidad durante el escalado ascendente manual o automático. Esta métrica no está disponible cuando se consultan datos de grupos de métricas de flotas. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
MinInstances	Número mínimo de instancias permitidas para la flota. El número mínimo de instancias de una flota determina el límite inferior de capacidad durante el escalado descendente manual o automático. Esta métrica no está disponible cuando se consultan datos de grupos de métricas de flotas. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
PercentIdleInstances	Porcentaje de todas las instancias activas que se encuentran inactivas (calculado como $\text{IdleInstances} / \text{ActiveInstances}$). Esta métrica se puede usar para el escalado automático. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
RecycledInstances	Número de instancias de spot que se han reciclado y reemplazado. Amazon GameLift Servers recicla las instancias de spot que actualmente no alojan sesiones de juego y que tienen una alta probabilidad de interrupción. Esta métrica solo está disponible para las flotas de EC2 administradas. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
RecycledSpotInstances	Número de instancias de spot que se han reciclado y reemplazado. Amazon GameLift Servers recicla las instancias de spot que actualmente no alojan sesiones de juego y que tienen una alta probabilidad de interrupción. Esta métrica solo está disponible para las flotas de contenedores. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación
InstanceSpotInterruptions	Número de las instancias de spot que se han interrumpido. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
CPUUtilization	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. El porcentaje de tiempo de CPU física que Amazon EC2 utiliza para ejecutar la instancia, que incluye el tiempo dedicado a ejecutar tanto el código de usuario como el código de Amazon EC2. Las herramientas de su sistema operativo pueden mostrar un porcentaje diferente CloudWatch al que se debe a factores como la simulación de dispositivos antiguos, la configuración de dispositivos no antiguos, las cargas de trabajo con muchas interrupciones, la migración en tiempo real y las actualizaciones en tiempo real. Unidad: porcentaje
NetworkIn	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. El número de bytes recibidos en todas las interfaces de red por la instancia. Esta métrica identifica el volumen de tráfico de red entrante para una aplicación en una sola instancia. Unidades: bytes
NetworkOut	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. El número de bytes enviados en todas las interfaces de red por la instancia. Esta métrica identifica el volumen de tráfico de red saliente para una aplicación en una sola instancia. Unidades: bytes

Métrica	Description (Descripción)
DiskReadBytes	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. Bytes leídos de todos los volúmenes del almacén de instancias disponibles para la instancia. Esta métrica se usa para determinar el volumen de datos que la aplicación lee del disco duro de la instancia. Se puede usar para determinar la velocidad de la aplicación. Unidades: bytes
DiskWriteBytes	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. Bytes escritos en todos los volúmenes del almacén de instancias disponibles para la instancia. Esta métrica se usa para determinar el volumen de datos que la aplicación escribe en el disco duro de la instancia. Se puede usar para determinar la velocidad de la aplicación. Unidades: bytes
DiskReadOps	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. Operaciones de lectura completadas de todos los volúmenes del almacén de instancias disponibles para la instancia en un periodo de tiempo especificado. Para calcular el promedio de I/O operaciones por segundo (IOPS) del período, divida el total de operaciones del período por el número de segundos de ese período. Unidades: recuento

Métrica	Description (Descripción)
DiskWriteOps	Métricas de EC2. En el caso de Amazon GameLift Servers, esta métrica representa el rendimiento del hardware en todas las instancias activas de una ubicación de flota. Operaciones de escritura completadas en todos los volúmenes del almacén de instancias disponibles para la instancia en un periodo de tiempo especificado. Para calcular el promedio de I/O operaciones por segundo (IOPS) del período, divida el total de operaciones del período por el número de segundos de ese período. Unidades: recuento

Procesos del servidor

Métrica	Description (Descripción)
ActiveServerProcesses	Procesos del servidor con el estado ACTIVE, lo que indica que se están ejecutando y pueden alojar sesiones de juego. El recuento incluye los procesos de servidor inactivos y los que alojan sesiones de juego. Esta métrica mide la capacidad total actual de los procesos del servidor. Unidades: recuento CloudWatch Estadísticas relevantes: media, mínima y máxima Dimensiones: ubicación
HealthyServerProcesses	Procesos del servidor activos registrados como en buen estado. Esta métrica es útil para realizar un seguimiento del estado general de los servidores de juego de la flota.

Métrica	Description (Descripción)
	Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
PercentHealthyServerProcesses	Porcentaje de todos los procesos del servidor activos registrados como en buen estado (calculado como $\text{HealthyServerProcesses} / \text{ActiveServerProcesses}$). Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
ServerProcessAbnormalTerminations	Procesos del servidor que se cerraron debido a circunstancias anormales desde el último informe. Esta métrica incluye las terminaciones iniciadas por el servicio de Amazon GameLift Servers. Esto ocurre cuando un proceso del servidor deja de responder , informa de forma sistemática de errores en las comprobaciones de estado o no finaliza correctamente (llamando a ProcessEnding()). Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
ServerProcessActivations	Procesos del servidor que pasaron correctamente del estado ACTIVATING a ACTIVE desde el último informe. Los procesos del servidor no pueden alojar sesiones de juego hasta que están activos. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación
ServerProcessTerminations	Procesos del servidor que se cerraron desde el último informe. Esto incluye todos los procesos del servidor que por algún motivo pasaron a tener el estado TERMINATED, incluidas las terminaciones del proceso normales y anormales. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Sesiones de juego

Métrica	Description (Descripción)
ActivatingGameSessions	Sesiones de juego con el estado ACTIVATING, lo que indica que se están iniciando. Las sesiones de juego no pueden alojar jugadores hasta que están activas. Un número elevado durante un periodo de tiempo prolongado puede indicar que las sesiones de juego no están pasando del estado ACTIVATING.

Métrica	Description (Descripción)
	G al estado ACTIVE. Se puede usar con el escalado automático. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
ActiveGameSessions	Sesiones de juego con el estado ACTIVE, lo que indica que pueden alojar jugadores y que están alojando cero o más jugadores. Esta métrica mide el número total de sesiones de juego alojadas actualmente. Se puede usar con el escalado automático. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
AvailableGameSessions	Procesos de servidor activos y en buen estado que no se utilizan actualmente para alojar una sesión de juego y que pueden iniciar una nueva sesión de juego sin retrasos para activar nuevos procesos o instancias del servidor. Se puede usar con el escalado automático.  Note Para las flotas que limitan las activaciones simultáneas de sesiones de juego, utilice la métrica <code>ConcurrentAvailableGameSessions</code>. Esa métrica representa con mayor precisión el número de sesiones de juego nuevas que se pueden iniciar sin ningún tipo de retraso. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
ConcurrentActivatableGameSessions	Procesos de servidor activos y en buen estado que no se utilizan actualmente para alojar una sesión de juego y que pueden iniciar inmediatamente una nueva sesión de juego. Esta métrica se diferencia de AvailableGameSessions de la siguiente manera: no incluye los procesos del servidor que actualmente no pueden activar una nueva sesión de juego debido a los límites en las activaciones de las sesiones de juego. (Consulte la configuración RuntimeConfiguration opcional de la flotaMaxConcurrentGameSessionActivations). En el caso de las flotas que no limitan las activaciones de las sesiones de juego, la métrica es idéntica a AvailableGameSessions . Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
PercentAvailableGameSessions	Porcentaje de slots de sesiones de juego en todos los procesos del servidor activos (en buen estado o en mal estado) que no se están usando actualmente (calculado como $\text{AvailableGameSessions} / [\text{ActiveGameSessions} + \text{AvailableGameSessions} + \text{unhealthy server processes}]$). Se puede usar con el escalado automático. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio Dimensiones: ubicación
GameSessionInterruptions	Número de sesiones de juego en instancias de spot que se han interrumpido. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Sesiones de jugador

Métrica	Description (Descripción)
CurrentPlayerSessions	Sesiones de jugador con el estado ACTIVE (el jugador está conectado a una sesión de juego activa) o con el estado RESERVED (al jugador se le ha asignado un slot en una sesión de juego pero aún no se ha conectado). Se puede usar con el escalado automático.

Métrica	Description (Descripción)
	Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo
PlayerSessionActivations	Sesiones de jugador que pasaron correctamente del estado RESERVED a ACTIVE desde el último informe. Esto ocurre cuando un jugador se conecta correctamente a una sesión de juego activa. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo

Métricas de flotas de contenedores

Las métricas de flota de contenedores proporcionan datos de supervisión del alojamiento de juegos en contenedores, como el estado del grupo de contenedores, el uso de los recursos y las métricas de rendimiento.

Note

Hay paneles mejorados para la supervisión de flotas de contenedores disponibles a través de [Amazon Managed Grafana](#) cuando se utiliza la solución de métricas de telemetría. Consulte [the section called “Supervisión con métricas de telemetría de servidor”](#) para obtener instrucciones de configuración.

Métrica	Description (Descripción)
ActiveGameServerContainerGroups	Grupos de contenedores que actualmente están activos y preparados para alojar sesiones de juego. Unidades: recuento

Métrica	Description (Descripción)
	CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
IdleGameServerContainerGroups	Grupos de contenedores que están activos, pero que actualmente no están alojando sesiones de juego. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
PendingGameServerContainerGroups	Grupos de contenedores que están en proceso de iniciarse. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
TerminatingGameServerContainerGroups	Grupos de contenedores que están en proceso de cerrarse. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
UnhealthyGameServerContainerGroupsReplaced	Grupos de contenedores que se han reemplazado debido a errores en las comprobaciones de estado. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación
ContainerCPUUtilizationPerInstance	Uso promedio de la CPU en todos los contenedores de una instancia. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
ContainerCPUReservation	Porcentaje de recursos de CPU reservados por contenedores. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
ContainerMemoryReservation	Porcentaje de recursos de memoria reservados por contenedores. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
ContainerMemoryUtilization	Porcentaje de memoria reservada que está en uso por parte de contenedores. Unidad: porcentaje CloudWatch Estadísticas relevantes: promedio, mínimo, máximo Dimensiones: ubicación
ContainerNetworkIn	Bytes de red recibidos por contenedores. Unidades: bytes CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación
ContainerNetworkOut	Bytes de red enviados por contenedores. Unidades: bytes CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación
ContainerStorageReadBytes	Bytes leídos del almacenamiento por contenedores. Unidades: bytes CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

Métrica	Description (Descripción)
ContainerStorageWriteBytes	Bytes escritos al almacenamiento por contenedores. Unidades: bytes CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: ubicación

DDoMétricas de protección S (puerta de entrada al jugador)

Amazon GameLift Servers Las métricas de Player Gateway rastrean el tráfico UDP que fluye a través de la capa de protección DDo S cuando PlayerGatewayMode está configurada en ENABLED oREQUIRED. Estas métricas le ayudan a monitorear los patrones de tráfico, identificar posibles ataques DDo S y realizar un seguimiento del rendimiento de los repetidores.

Métrica	Description (Descripción)
PlayerGatewayPacketsIn	Número de paquetes UDP recibidos de los jugadores y reenviados a los servidores del juego a través de la capa de protección DDo S. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: FleetId, Ubicación
PlayerGatewayBytesIn	Número de bytes recibidos de los jugadores y reenviados a los servidores del juego a través de la capa de protección DDo S. Unidades: bytes CloudWatch Estadísticas relevantes: suma, media, mínima y máxima

Métrica	Description (Descripción)
	Dimensiones: FleetId, Ubicación
PlayerGatewayPacketsOut	Número de paquetes UDP recibidos de los servidores del juego y reenviados a los jugadores a través de la capa de protección DDoS. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: FleetId, Ubicación
PlayerGatewayBytesOut	Número de bytes recibidos de los servidores del juego y reenviados a los jugadores a través de la capa de protección DDoS. Unidades: bytes CloudWatch Estadísticas relevantes: suma, media, mínima y máxima Dimensiones: FleetId, Ubicación
PlayerGatewayPacketsThrottled	Número de paquetes entrantes descartados debido a la limitación de velocidad impuesta por la capa de protección DDoS. Unidades: recuento CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: FleetId, Ubicación

Métrica	Description (Descripción)
PlayerGatewayBytesThrottled	Se ha reducido el número de bytes entrantes debido a la limitación de velocidad impuesta por la capa de protección DDo S. Unidades: bytes CloudWatch Estadísticas relevantes: suma, promedio, mínimo, máximo Dimensiones: FleetId, Ubicación
PlayerGatewayPlayerSessions	Sesiones de jugadores que actualmente están protegidas por la capa de protección DDo S. Unidades: recuento CloudWatch Estadísticas relevantes: media, mínima y máxima Dimensiones: FleetId, Ubicación

Métricas de Amazon GameLift Servers para colas

El espacio de nombres Amazon GameLift incluye las siguientes métricas relativas a la actividad en una cola de ubicación de sesión de juego. Las colas se utilizan con una solución de Amazon GameLift Servers administrada. El Amazon GameLift Servers servicio envía métricas a CloudWatch cada minuto.

Métrica	Description (Descripción)
AverageWaitTime	Promedio de tiempo que las solicitudes de ubicación de sesión de juego en la cola con el estado PENDING han esperado a ser atendidas. Unidades: segundos

Métrica	Description (Descripción)
	CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma Dimensiones: ubicación
FirstChoiceNotViable	Sesiones de juego que se transfirieron correctamente pero NO a la primera flota elegida, ya que dicha flota no se consideraba viable (por ejemplo, una flota de spot con un gran número de interrupciones). Esta métrica se basa en el costo, no en la latencia. La primera flota elegida es la primera flota que aparece en la cola o, si una solicitud de ubicación contiene datos sobre la latencia de los jugadores, la primera flota elegida por la priorización de FleetIQ. Si no hay flotas de spot viables, se puede seleccionar cualquier flota en esa región. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
FirstChoiceOutOfCapacity	Sesiones de juego que se transfieren correctamente pero NO a la primera flota elegida, ya que dicha flota no tiene recursos disponibles. La primera flota elegida es la primera flota que aparece en la cola o, si una solicitud de ubicación contiene datos sobre la latencia de los jugadores, la primera flota elegida por la priorización de FleetIQ. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma

Métrica	Description (Descripción)
LowestLatencyPlacement	Sesiones de juego que se transfirieron correctamente a una región que ofrece la menor latencia posible en la cola para los jugadores. Esta métrica solamente se emite solo cuando los datos de latencia de los jugadores se incluyen en la solicitud de ubicación. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
LowestPricePlacement	Sesiones de juego que se transfirieron correctamente a una flota que tiene el menor precio posible de la cola para la región elegida. Esta flota puede ser una flota de spot o, si la cola no tiene instancias de spot, una instancia bajo demanda. Esta métrica solamente se emite solo cuando los datos de latencia de los jugadores se incluyen en la solicitud de ubicación. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
Placement <region name>	Sesiones de juego que se transfirieron correctamente a las flotas situadas en la región especificada. Esta métrica desglosa el valor PlacementSucceeded por regiones. Unidades: recuento CloudWatch Estadísticas relevantes: Suma

Métrica	Description (Descripción)
PlacementsCanceled	Solicitudes de ubicación de sesión de juego que se cancelaron porque se agotó el tiempo de espera desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
PlacementsFailed	Solicitudes de ubicación de sesión de juego que no se realizaron correctamente por alguna razón desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
PlacementsStarted	Solicitudes de ubicación de sesión de juego nuevas que se añadieron a la cola desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
PlacementsSucceeded	Solicitudes de ubicación de sesión de juego que dieron lugar a una nueva sesión de juego desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma

Métrica	Description (Descripción)
PlacementsTimedOut	Solicitudes de ubicación de sesión de juego que alcanzaron el límite de tiempo de espera de la cola sin ser atendidas desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
QueueDepth	Número de solicitudes de ubicación de sesión de juego de la cola con el estado PENDING. Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma Dimensiones: ubicación

Métricas de Amazon GameLift Servers para el emparejamiento

El espacio de nombres Amazon GameLift incluye métricas sobre la actividad de FlexMatch relativa a las configuraciones de emparejamiento y reglas de emparejamiento. El emparejamiento de FlexMatch se utiliza con una solución de Amazon GameLift Servers administrada. El Amazon GameLift Servers servicio envía métricas a CloudWatch cada minuto.

Para obtener más información sobre la secuencia de la actividad de emparejamiento, consulte [Funcionamiento de Amazon GameLift Servers FlexMatch](#).

Configuraciones de emparejamiento

Métrica	Description (Descripción)
CurrentTickets	En la actualidad, se están procesando solicitudes de emparejamiento o están a la espera de ser procesadas.

Métrica	Description (Descripción)
	Unidades: recuento CloudWatch Estadísticas relevantes: promedio, mínimo, máximo, suma
MatchAcceptancesTimedOut	Para las configuraciones de emparejamiento que requieren aceptación, los posibles emparejamientos que agotaron su tiempo durante la aceptación desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
MatchesAccepted	Para las configuraciones de emparejamiento que requieren aceptación, los posibles emparejamientos que se aceptaron desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
MatchesCreated	Emparejamientos potenciales creados desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
MatchesPlaced	Emparejamientos que se ubicaron en una sesión de juego desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma

Métrica	Description (Descripción)
MatchesRejected	Para las configuraciones de emparejamiento que requieren aceptación, los posibles emparejamientos que rechazó al menos un jugador desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
PlayersStarted	Jugadores en incidencias de emparejamiento que se añadieron desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
TicketsFailed	Solicitudes de emparejamiento con error desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
TicketsStarted	Nuevas solicitudes de emparejamiento creadas desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma
TicketsTimedOut	Solicitudes de emparejamiento que alcanzaron el límite de tiempo de espera desde el último informe. Unidades: recuento CloudWatch Estadísticas relevantes: Suma

Métrica	Description (Descripción)
TimeToMatch	Para las solicitudes de emparejamiento que se pusieron en un emparejamiento potencial antes del último informe, la cantidad de tiempo entre la creación de la incidencia y los posibles emparejamientos. Unidades: segundos CloudWatch Estadísticas relevantes: muestras de datos, promedio, mínimo, máximo
TimeToTicketCancel	Para las solicitudes de emparejamiento que se cancelaron antes del último informe, la cantidad de tiempo entre la creación de la incidencia y la cancelación. Unidades: segundos CloudWatch Estadísticas relevantes: muestras de datos, promedio, mínimo, máximo
TimeToTicketSuccess	Para las solicitudes de emparejamiento con éxito antes del último informe, la cantidad de tiempo entre la creación de la incidencia y la ubicación de emparejamiento con éxito. Unidades: segundos CloudWatch Estadísticas relevantes: muestras de datos, promedio, mínimo, máximo

Reglas de emparejamiento

Métrica	Description (Descripción)
RuleEvaluationsPassed	Evaluaciones de reglas durante el proceso de emparejamiento superadas desde el último informe. Esta métrica se limita a las 50 primeras reglas. Unidades: recuento CloudWatchEstadísticas relevantes: suma
RuleEvaluationsFailed	Evaluaciones de reglas durante el emparejamiento que no se superaron desde el último informe. Esta métrica se limita a las 50 primeras reglas. Unidades: recuento CloudWatch Estadísticas relevantes: Suma

Amazon GameLift Servers Métricas de para FleetIQ

El espacio de nombres Amazon GameLift incluye métricas para el grupo de servidores de juegos y la actividad del servidor de juegos de FleetIQ como parte de una solución de FleetIQ independiente para el alojamiento de juegos. El Amazon GameLift Servers servicio envía métricas a CloudWatch cada minuto. Consulte también [Monitorización de sus grupos e instancias de Auto Scaling mediante Amazon CloudWatch](#) en la Guía del usuario de Amazon EC2 Auto Scaling.

Métrica	Description (Descripción)
AvailableGameServers	Servidores para videojuegos que están disponibles para ejecutar una ejecución de juego y en los que actualmente no hay actividad de juego. Este número incluye servidores para videojuegos que se han reclamado pero todavía se encuentran en estado AVAILABLE. Unidades: recuento

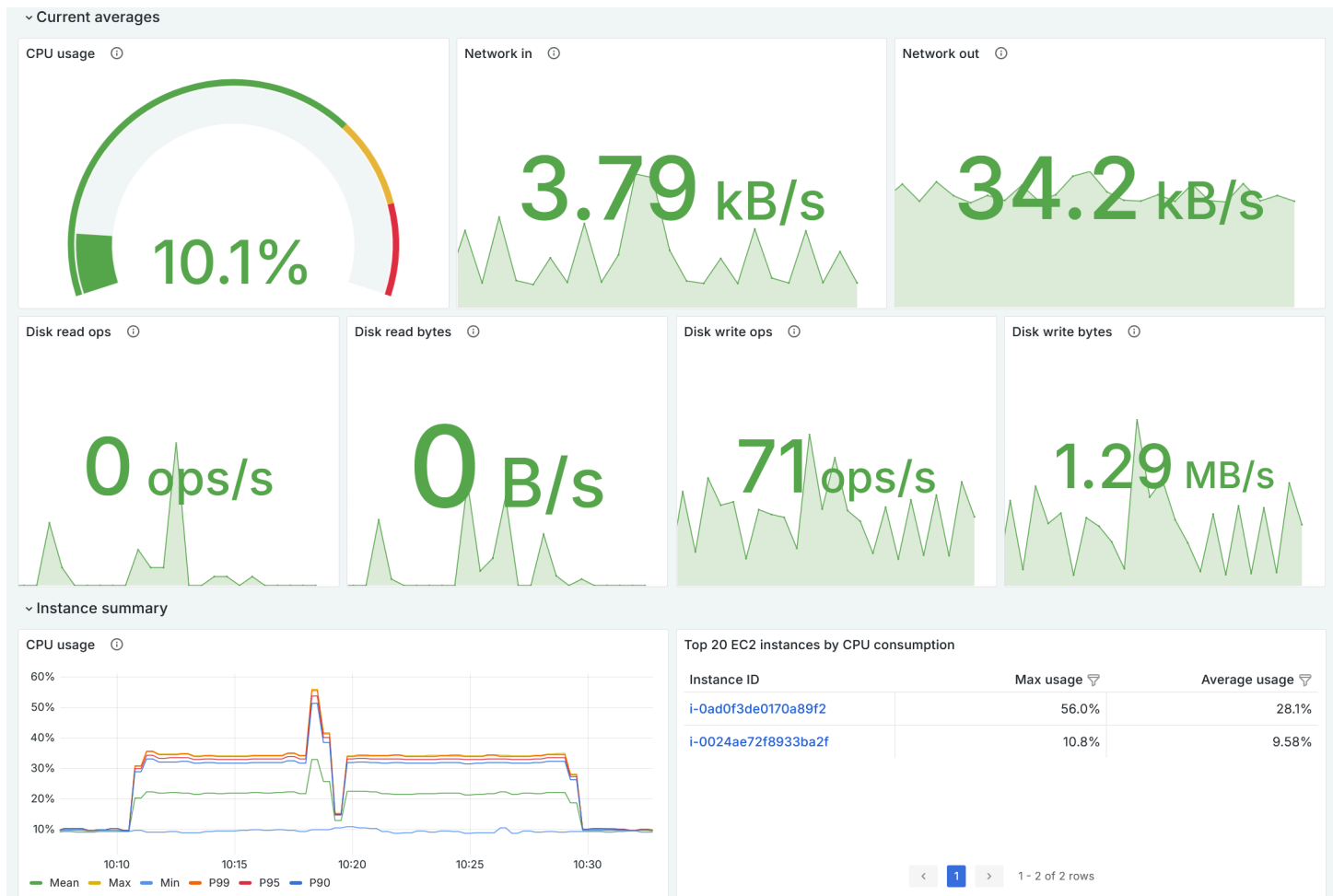
Métrica	Description (Descripción)
	CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup
UtilizedGameServers	Servidores para videojuegos en los que actualmente hay actividad de juego. Este número incluye los servidores de juegos que están en estado UTILIZADOS. Unidades: recuento CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup
DrainingAvailableGameServers	Servidores para videojuegos en instancias programadas para la terminación que actualmente no son compatibles con la actividad de juego. Estos servidores para videojuegos tienen la prioridad más baja de reclamación en respuesta a una nueva solicitud de reclamación. Unidades: recuento CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup

Métrica	Description (Descripción)
DrainingUtilizedGameServers	Servidores para videojuegos en instancias programadas para la terminación que actualmente admiten la actividad de juego. Unidades: recuento CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup
PercentUtilizedGameServers	Parte de los servidores para videojuegos que actualmente admiten ejecuciones de juegos. Esta métrica indica la cantidad de capacidad del servidor para videojuegos que se está utilizando actualmente. Es útil para habilitar una política de Auto Scaling que agregue y elimine dinámicamente instancias para satisfacer la demanda de jugadores. Unidad: porcentaje CloudWatch Estadísticas relevantes de Amazon: promedio, mínimo, máximo Dimensiones: GameServerGroup
GameServerInterruptions	Servidores para videojuegos en instancias de spot que se interrumpieron debido a una disponibilidad de spot limitada. Unidades: recuento CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup, InstanceType

Métrica	Description (Descripción)
InstanceInterruptions	Instancias de spot que se interrumpieron debido a una disponibilidad limitada. Unidades: recuento CloudWatch Estadísticas relevantes de Amazon: suma Dimensiones: GameServerGroup, InstanceType

Supervisión con métricas de telemetría de servidor

Amazon GameLift Servers se puede configurar para recopilar y publicar métricas de telemetría para servidores de juegos que se ejecutan en flotas gestionadas de Amazon EC2 y Container. Estas métricas están disponibles después de implementar el recopilador de telemetría con la compilación del servidor. El sistema de métricas es compatible con todos SDKs (C++, C#, Go), con todos los complementos (Unreal, Unity) y con el Game Server Wrapper. Amazon GameLift Servers Los datos de las métricas fluyen a los paneles de [Amazon Managed Service para Prometheus](#), [the section called “Supervise con CloudWatch”](#) y [Amazon Managed Grafana](#) (recomendado para la visualización).



Supervisa el rendimiento de tus servidores de juegos mediante paneles prediseñados en Amazon Managed Grafana o Amazon CloudWatch.

Ventajas de las métricas de telemetría

El sistema de métricas de telemetría ofrece cinco ventajas clave:

- **Métricas específicas del motor de juegos:** los complementos de motores de juegos (Unreal, Unity) proporcionan una integración nativa con métricas de rendimiento específicas del motor, como el tiempo de ciclo del servidor, la velocidad de fotogramas y el uso de recursos por motor, que son fundamentales para la optimización del rendimiento de los juegos.
- **Compatibilidad con métricas personalizadas:** define y realice un seguimiento de sus propias métricas específicas del juego mediante llamadas a funciones del SDK del servidor para supervisar los eventos de juego personalizados, el rendimiento de la lógica empresarial y los puntos de datos específicos de la aplicación que son importantes para el juego.

- Recopilación automatizada: las métricas fluyen automáticamente tras la implementación del recopilador de telemetría sin necesidad de instrumentación adicional y con una integración directa con [Amazon Managed Service para Prometheus](#) y Amazon. CloudWatch
- Supervisión a varios niveles: métricas de capacidad y escalado por flota, métricas de uso de recursos por instancia y métricas de las sesiones de juego para monitorizar el rendimiento.
- Compatibilidad universal: funciona con todos los entornos de desarrollo Amazon GameLift Servers compatibles, está integrado con todos los servidores y es compatible de forma nativa con los complementos de los SDKs motores de juegos.

Note

Las métricas de telemetría están disponibles para las flotas gestionadas de Amazon GameLift Servers Amazon EC2 o de contenedores que ejecutan Amazon Linux 2023 o Windows.

Antes de empezar

Recursos necesarios AWS

- AWS cuenta configurada para Amazon GameLift Servers.
- Flota administrada que se ejecuta en:
 - Amazon EC2 con sistemas operativos compatibles, o
 - Contenedores con Amazon Linux 2023
- Permisos adecuados de IAM

Requisitos de IAM

Los siguientes permisos de IAM solo son necesarios si tiene previsto utilizar el servicio correspondiente:

- [Amazon Managed Service para Prometheus](#) (necesario solo si se van a publicar métricas en Prometheus)
 - Permiso de `aps:RemoteWrite`
 - Acceso a su espacio de trabajo de Prometheus

- Amazon CloudWatch (obligatorio solo si publicas las métricas en Amazon CloudWatch)
 - Permiso de `cloudwatch:PutMetricData`
 - Acceso a los espacios de nombres de métricas
- [Amazon Managed Grafana](#) (necesario solo si se utilizan paneles de Grafana)
 - Permiso de `grafana:Read`
 - Configuración del SSO para el acceso al panel

Implementación

Seleccione la ruta de implementación en función del entorno de desarrollo:

Implementación del SDK

Tipo de SDK	Configuración del SDK	Métricas personalizadas	referencia de la API
SDK para Go	Guía de configuración completa	API de métricas de Go	Acciones y tipos de datos de Go
SDK para C#	Guía de configuración completa	API de métricas de C#	Acciones y tipos de datos de C#
SDK para C++	Guía de configuración completa	API de métricas de C++	Acciones y tipos de datos de C++

Implementación del complemento

Complemento	Configuración del complemento	Métricas personalizadas	referencia de la API
Irreal	Guía de configuración completa	API de métricas de Unreal	Acciones y tipos de datos de Unreal
Unity	Guía de configuración completa	API de métricas de Unity	Acciones y tipos de datos de C#

Flujo de trabajo de implementación

Cada implementación sigue un proceso consistente en dos pasos:

1. Guía de configuración completa (Metrics.md): despliegue de infraestructura, configuración de AWS infraestructura, configuración de flota y configuración del panel de control de Grafana.
2. Guía de implementación de la API (CUSTOM_METRICS.md): uso del SDK específico del lenguaje, tipos de métricas, creación de métricas personalizadas y configuración avanzada.

Verificación

1. Valide el flujo de métricas consultando los datos de telemetría entrantes en su espacio de trabajo de [Amazon Managed Service for Prometheus](#) o en la consola de CloudWatch Amazon.
2. Compruebe la visibilidad de los paneles en [Amazon Managed Grafana](#) con ayuda de los paneles prediseñados.
3. Pruebe las métricas personalizadas verificando que aparecen en los paneles de monitoreo.

Note

Una vez completada la implementación, regrese a esta página y vaya a la página [Métricas disponibles](#).

Métricas disponibles

Las métricas se dividen en tres categorías:

- Métricas recopiladas automáticamente
- Métricas proporcionadas por el SDK
- Métricas personalizadas

Recopilación automática de métricas

No se requieren cambios en el código para estas métricas:

Métricas de la instancia

Tipo de métrica	Description (Descripción)	Caso de uso
CPU	Porcentaje de uso por instancia	Monitoreo de recursos
Memoria	Uso de memoria física y porcentaje	Planificación de la capacidad
E/S de red	Bytes y paquetes enviados/recibidos	Estado de la conexión
E/S de disco	Operaciones de lectura/escritura y rendimiento	Rendimiento del almacenamiento

Métricas de flota

Tipo de métrica	Description (Descripción)	Caso de uso
Instancias activas	Recuento de instancias en ejecución	Escalado de flotas
Sesiones de juego	Sesiones activas y disponibles	Administración de la capacidad
Sesiones de juego bloqueadas	Sesiones de juego que se han bloqueado	Supervisión de errores

Métricas proporcionadas por el SDK

Requiere llamadas a funciones del SDK en el código:

Métricas de tiempo del servidor

Métrica	Description (Descripción)	Implementación
Tiempo diferencial del servidor	Diferencia de tiempo entre el tiempo de graduación del servidor actual y el anterior. Mide la constancia de la frecuencia de graduación del servidor	Llame GetDeltaTime ()

Métrica	Description (Descripción)	Implementación
Frecuencia de graduación del servidor	Muestra el número de veces por segundo que el servidor procesa las actualizaciones	Se calcula automáticamente
Tiempo de graduación del servidor	La cantidad de tiempo que tarda el servidor en procesar una sola graduación o actualización	Llamar GetTickTime ()
Tiempo de graduación del servidor mundial	La cantidad de tiempo que tarda el servidor en actualizar el mundo del juego con cada graduación	Llamar GetWorldUpdateTime ()

Implementación: en el caso de los distintos motores SDKs (C++, C#, Go), estas métricas se implementan llamando a las funciones del SDK desde el bucle de juego con valores de tiempo calculados. En el caso de los complementos de motor (Unreal o Unity), estas métricas se capturan automáticamente mediante la integración del motor.

Métricas de red

Métrica	Description (Descripción)	Implementación
Connections	Número total de conexiones de red establecidas por el agente	Automático después de () InitMetrics
Red I/O (bytes)	Número total de bytes que envía y recibe el nodo a través de la red	Automático después de InitMetrics ()
Red I/O (paquetes)	Cantidad total de paquetes de red enviados y recibidos por el servidor	Automático después de InitMetrics ()
Pérdida de paquetes	Porcentaje de paquetes de red que se pierden durante la transmisión	Automático después de InitMetrics ()

Implementación: integre las llamadas a las funciones del SDK con la biblioteca de redes. El SDK proporciona directrices para diferentes implementaciones de red.

Métricas de proceso

Métrica	Description (Descripción)	Implementación
Uso de la CPU (%)	Porcentaje de recursos de CPU que utiliza el proceso del servidor de juegos	Automático después de InitMetrics ()
Uso de memoria (unidades)	Cantidad total de memoria que consumen los procesos del servidor	Automático después de InitMetrics ()
Uso de memoria física (%)	Porcentaje de memoria física total del servidor que está en uso actualmente	Automático después de InitMetrics ()
Estado del servidor	Estado del servidor de juegos	Automático después de InitMetrics ()

Implementación: el SDK recopila automáticamente estas métricas para cada proceso de sesión de juego.

Organización del panel por proceso

Las métricas por proceso están disponibles en dos paneles especializados:

- Panel de rendimiento del servidor: contiene los tiempos del servidor (tiempo delta, tasa de tics, tiempo de tic, tiempo de tic mundial), métricas de red (conexiones, I/O bytes/paquetes, pérdida de paquetes), uso de memoria y uso de CPU para sesiones de juego individuales.
- Panel de rendimiento de las instancias: incluye las tablas “Las N sesiones de juego que más memoria consumen” y las “N sesiones de juego que más CPU consumen”, que ayudan a identificar qué procesos contribuyen en mayor medida al consumo de recursos de la instancia. Haga clic en los enlaces de las sesiones de juego para investigar más a fondo las métricas detalladas.

Casos de uso de las métricas por proceso

Las métricas o procesos por proceso admiten los siguientes escenarios de supervisión per-game-session :

- Investigación exhaustiva del rendimiento: cuando un rendimiento se host/instance ha reducido debido a procesos o sesiones de juego específicos, las métricas por proceso ayudan a identificar

qué proceso causó el problema en las tablas de sesiones de juego que más memoria y CPU consumen.

- Investigación de bloqueos del servidor de juegos: cuando una sesión de juego se bloquea, estas métricas ayudan a determinar si el bloqueo se debe a una falta de memoria, a una sobrecarga de la CPU o a problemas de ancho de banda de la red.
- Investigación de problemas notificados por los jugadores: cuando los jugadores informan de retrasos o interrupciones durante el juego, las métricas por proceso ayudan a identificar los cuellos de botella en la CPU, la memoria, la red, el tiempo de graduación o el tiempo de actualización del mundo de juego.
- Identificación de las variaciones de rendimiento en las diferentes compilaciones: las métricas del tiempo de graduación, frecuencia de graduación y tiempo de actualización del mundo de juego permiten a los desarrolladores medir las variaciones de rendimiento del juegos en las distintas compilaciones del servidor.
- Detección de retrasos y ralentización del juego: las métricas de tiempo de graduación, frecuencia de graduación y tiempo de actualización del mundo de juego reflejan la rapidez con la que el servidor actualiza el juego, lo que repercute directamente en la experiencia del usuario.
- Análisis comparativo: identifique cómo los diferentes escenarios del juego afectan al rendimiento del servidor en función de factores como el número de jugadores, el modo de juego y otras variables.

Organización en paneles

Las métricas se organizan en paneles especializados en [Amazon Managed Grafana](#) para ofrecer diferentes posibilidades de supervisión. Los paneles disponibles dependen del tipo de flota:

EC2 Paneles de control de flota

- EC2 Panel de control general de la flota: capacidad de la flota de alto nivel, información sobre la escalabilidad, jugadores simultáneos (CCU), instancias, capacidad de los jugadores y sesiones de juego interrumpidas.
- Panel de información general de las instancias: métricas agregadas de nivel de host de todas las instancias, incluidos el uso promedio de la CPU, la memoria, la red y el disco.
- Panel de rendimiento de las instancias: métricas detalladas de las instancias individuales con las tablas “N sesiones de juego que consumen más memoria” y “N más sesiones de juego que consumen CPU” para identificar los procesos que consumen muchos recursos.

- Panel de rendimiento del servidor (EC2): cronometría de los bucles de juego, rendimiento de la red, memoria y CPU para sesiones de juego individuales en EC2 instancias.

Paneles de la flota de contenedores

- Panel de información general sobre la flota de contenedores: descripción general de alto nivel de la utilización de los recursos de la flota de contenedores, incluida la reserva de CPU, la utilización de la memoria y el estado del grupo de contenedores.
- Panel de rendimiento de contenedores: métricas detalladas de los contenedores individuales dentro de tareas específicas de ECS, como el uso de la CPU, el uso de la memoria, las E/S de la red y el rendimiento del almacenamiento.
- Panel de rendimiento del servidor (contenedor): medición de la duración de los ciclos de juego, el rendimiento de la red, la memoria y la CPU de las sesiones de juego individuales en contenedores.

Para obtener más información e instrucciones detalladas, consulte [the section called “Organización y uso de los paneles”](#).

Funcionamiento

El sistema de métricas de telemetría sigue un flujo de datos sencillo de cuatro etapas que va desde los servidores de juegos hasta los paneles de visualización.

Recopilación: el servidor de juegos, integrado con el SDK o el complemento GameLift del servidor, emite automáticamente las métricas a un recopilador de telemetría local que se ejecuta en la misma instancia. El SDK captura tanto las métricas automáticas (ciclo de vida del servidor y uso de recursos) como las métricas personalizadas que defina en el código.

Procesamiento: el recopilador de telemetría agrega las métricas procedentes del servidor de juegos y las combina con datos de rendimiento de nivel de instancia (uso de la CPU, de la memoria, de la red y del disco). Esto proporciona una imagen completa tanto del rendimiento del juego como de la infraestructura subyacente.

Almacenamiento: las métricas procesadas se exportan al almacén de métricas que elijas: [Amazon Managed Service para Prometheus para](#) un almacenamiento de series temporales de alto rendimiento, CloudWatch AWS Amazon para la integración de servicios, o ambos. Todas las transmisiones de datos son objeto de autenticación y cifrado.

Visualización: [Grafana gestionada por Amazon](#) se conecta a tu almacén de métricas para mostrar cuadros de mando GameLift prediseñados. Estos paneles proporcionan información general de las flotas, detalles del rendimiento de los servidores y vistas de supervisión de contenedores que le ayudan a supervisar y solucionar los problemas de la infraestructura de alojamiento de juegos.

 Note

Por motivos de seguridad, toda la transmisión de métricas entre el servidor de juegos y el recopilador de telemetría se produce de forma local en la instancia. Solo el recopilador se comunica con los AWS servicios mediante la autenticación adecuada.

Organización y uso de los paneles

Consulta tus métricas en paneles completos de [Amazon Managed Grafana](#). Los paneles disponibles dependen del tipo de flota:

Disponibilidad del panel de control por tipo de flota

En la siguiente tabla se muestran los paneles disponibles para cada tipo de flota:

Panel de control	Tipo de flota	Description (Descripción)
EC2 Descripción general de la flota	EC2 Flota	Muestra información sobre los jugadores simultáneos (CCU), las instancias y la capacidad de los jugadores
Información general de instancias	EC2 Flota	Muestra el uso promedio de CPU, memoria y red en todas las instancias de la flota
Rendimiento de las instancias	EC2 Flota	Muestra métricas detalladas (CPU, memoria, disco, red) de una instancia individual
Información general sobre la flota de contenedores	Flota de contenedores	Muestra la utilización media de los recursos de todos los contenedores

Panel de control	Tipo de flota	Description (Descripción)
		res de una flota de contenedores administrada
Rendimiento de los contenedores	Flota de contenedores	Muestra métricas detalladas de los contenedores individuales dentro de una tarea de ECS específica
Rendimiento del servidor	Ambos	Muestra el rendimiento de la red, la memoria y el tiempo de ejecución de un proceso de servidor de juegos específico (versiones independientes para EC2 las flotas de contenedores)

EC2 Flotas gestionadas:

- EC2 La descripción general de la flota proporciona información de alto nivel sobre la capacidad y el escalamiento de la flota
- Utilice la descripción general de las instancias y los paneles de rendimiento de las instancias para la supervisión a nivel de host.
- Métricas recopiladas a través del receptor hostmetrics para obtener visibilidad a nivel del sistema.
- Céntrese en la utilización de los recursos y el rendimiento de las EC2 instancias.
- Server Performance (EC2) monitorea las métricas de las aplicaciones del servidor de juegos independientemente de la infraestructura subyacente.

Flotas de contenedores administradas:

- Utilice los paneles de información general sobre la flota de contenedores y del rendimiento de los contenedores para supervisar las tareas de ECS y a nivel de los contenedores.
- Métricas recopiladas mediante ECS Container Receiver para obtener visibilidad de la carga de trabajo en contenedores.
- Céntrese en la agregación a nivel de tareas y en el aislamiento de los recursos de los contenedores.
- Rendimiento del servidor (contenedor) supervisa las métricas de las aplicaciones de los servidores de juegos con independencia de la infraestructura subyacente.

EC2 Panel de información general de la flota

En este panel de control se ofrece información general de alto nivel sobre el uso y la capacidad de su flota a nivel mundial y por ubicación. Incluye gráficos que muestran el número de paradas, inicios y bloqueos de los servidores de juegos, así como el porcentaje de servidores de juegos en buen estado. Puede filtrar por FleetID y ubicación.

Métricas de resumen de flota

En la siguiente tabla se muestran las métricas disponibles en el panel Información general de la flota:

Panel de control	Nombre de métrica	Definición
Capacidad y CCU globales	CCU	Muestra el número de usuarios simultáneos en todos los servidores del juego en todas las instancias de todas las ubicaciones
	CCU global	Muestra el número de usuarios simultáneos en todos los servidores del juego en todas las instancias del mundo
	Instancias activas	Muestra el número total de instancias de la flota que están activas
	Procesos activos	Muestra el número de procesos del servidor de juegos activos que están listos para alojar una sesión de juego
	Inicios del servidor de juegos	Muestra el número de sesiones de juego que se han iniciado en toda la flota
Capacidad y CCU globales	Servidores de juegos en buen estado	Muestra el porcentaje medio de servidores de juegos que informan de un buen estado a Amazon GameLift Servers en toda la flota.
	Sesiones de juego bloqueadas	Muestra la sesión IDs de juego de las sesiones de juego bloqueadas. Haga clic en el enlace al ARN de la sesión del juego para ir al panel de rendimiento del servidor.

Panel de control	Nombre de métrica	Definición
CCU de ubicación	Jugadores de ubicación	Muestra el número de jugadores simultáneos en una ubicación, incluidos todos los servidores del juego en todas las instancias de la ubicación
	Capacidad de ubicación	Muestra el uso de la capacidad (%) en una ubicación y el porcentaje de servidores de juegos en uso en esa ubicación

 Note

Las métricas de la CCU deben implementarse en el código del servidor de juegos. Estas métricas no se recopilan automáticamente y tu aplicación debe implementarlas e informarlas.

Panel de Información general de instancias

Este panel proporciona métricas agregadas a nivel de host en todas las instancias de su flota. Los promedios actuales muestran el estado general de las instancias. Cuando el rendimiento disminuya, compruebe si hay cuellos de botella en el uso de la CPU, el consumo de memoria y el consumo de red y disco. Puede filtrar por FleetID y ubicación.

Métricas de información general de las instancias

En la siguiente tabla se muestran las métricas disponibles en el panel Información general de instancias:

Panel de control	Nombre de métrica	Definición
Resumen de instancia	Uso promedio de la CPU	El resumen de instancias hace un seguimiento del uso medio de recursos por instancia. Porcentaje medio de recursos de CPU que se utilizan en todas las instancias de la flota. Los patrones de alta utilización requieren

Panel de control	Nombre de métrica	Definición
		comprobar las métricas de las instancias individuales y la distribución de los recursos
	Uso máximo de la CPU	El uso de CPU más alto registrado en cualquier instancia de la flota durante ese período
	Uso de la CPU por instancia	Utilización individual de la CPU para cada instancia de la flota, lo que permite identificar los hotspots de recursos
Resumen de instancia	Uso promedio de memoria	Porcentaje medio de RAM que se utiliza en todas las instancias de la flota. Los patrones de alta utilización requieren comprobar las métricas de las instancias individuales y la distribución de los recursos, como el consumo de memoria
	Uso máximo de memoria	El uso de memoria más alto registrado en cualquier instancia de la flota durante ese período
	Uso de memoria por instancia	Utilización de memoria individual para cada instancia de la flota, lo que ayuda a identificar las cargas de trabajo que consumen mucha memoria
Resumen de red	Red I/O (bytes)	El resumen de la red refleja la conectividad media de las instancias. Volumen medio de tráfico de red (enviado y recibido) en todas las instancias de la flota. En caso de problemas de latencia o conectividad, comprueba las métricas de cada instancia individual, como las velocidades de red I/O y paquetes
	Red I/O (paquetes)	Velocidad media de paquetes de red (enviados y recibidos) en todas las instancias de la flota

Panel de control	Nombre de métrica	Definición
	Uso de la red por instancia	Utilización de la red individual para cada instancia, útil para identificar los cuellos de botella de la red
Resumen de disco	I/O Operaciones de disco	El resumen del disco indica el rendimiento medio del disco de las instancias. Promedio de read/write operaciones de disco en todas las instancias de la flota. Los tiempos de respuesta lentos sugieren examinar las operaciones de disco, I/O los tiempos de espera y las operaciones pendientes a nivel de instancia individual
	I/O Rendimiento del disco	read/write Rendimiento medio de disco en todas las instancias de la flota
	Uso de disco por instancia	Utilización del disco individual para cada instancia, lo que ayuda a identificar los problemas de rendimiento del almacenamiento

Note

Las métricas a nivel de instancia se recopilan a través del receptor hostmetrics y proporcionan visibilidad a nivel de sistema del rendimiento de la infraestructura de su flota. Utilice este panel para identificar las tendencias generales del estado de la flota y analizar en detalle las instancias individuales cuando se detecten problemas de rendimiento.

Panel de rendimiento de instancias

Este panel proporciona métricas de rendimiento detalladas para instancias individuales. Los promedios actuales muestran el estado general de la instancia. Cuando el rendimiento disminuya, compruebe si hay cuellos de botella en el uso de la CPU, el consumo de memoria y el consumo del sistema de archivos. Incluye las tablas “Las N sesiones de juego que más memoria consumen” y las “N sesiones de juego que más CPU consumen”, que ayudan a identificar qué procesos contribuyen

en mayor medida al consumo de recursos de la instancia. Haga clic en los enlaces de las sesiones de juego para investigar más a fondo las métricas detalladas. Puede filtrar por un ID de instancia específico.

Métricas de rendimiento de las instancias

En la siguiente tabla se muestran las métricas disponibles en el panel de control de rendimiento de instancias:

Panel de control	Nombre de métrica	Definición
Resumen de cargas	Las N sesiones de juego que más memoria consumen	El resumen de carga rastrea el uso de recursos por parte de los servidores de juegos. Lista clasificada de las sesiones de juego que consumen más recursos de memoria en esta instancia. Los patrones de alta utilización requieren comprobar las métricas de las sesiones individuales y la distribución de los recursos. Haga clic en los enlaces de las sesiones de juego para investigar más a fondo las métricas detalladas
	Las N sesiones de juego que más consumen CPU	Lista clasificada de las sesiones de juego que consumen más recursos de CPU en esta instancia. Los patrones de alta utilización requieren comprobar las métricas de las sesiones individuales y la distribución de los recursos, como la CPU y la memoria. Haga clic en los enlaces de las sesiones de juego para investigar más a fondo las métricas detalladas
	Uso de CPU por sesión de juego	Desglose del uso individual de la CPU que muestra el consumo de recursos por cada sesión de juego activa
	Uso de memoria por sesión de juego	Desglose del uso de la memoria individual que muestra el consumo de recursos por cada sesión de juego activa

Panel de control	Nombre de métrica	Definición
Promedios actuales	Uso de CPU de la instancia	Utilización general de la CPU de la instancia seleccionada a lo largo del tiempo
	Uso de memoria de instancia	Utilización general de la memoria de la instancia seleccionada a lo largo del tiempo
	Uso del sistema de archivos de la instancia	Consumo del sistema de archivos de la instancia seleccionada, útil para identificar problemas de capacidad de almacenamiento
Network	E/S de red de instancias	La red refleja la conectividad de la instancia . Volumen de tráfico de red y velocidades de paquetes para la instancia seleccionada. Si tiene problemas de latencia o conectividad, investigue las E/S de la red, las velocidades de paquetes y los recuentos de errores
	Conexiones de red	Número de conexiones de red activas en la instancia seleccionada
	Recuentos de errores de red	Estadísticas de errores de red para identificar problemas de conectividad
Disk	Operaciones de disco I/O	El disco indica el rendimiento del disco. read/write Actividad y utilización del disco para la instancia seleccionada. Los tiempos de respuesta lentos sugieren examinar las operaciones del disco, I/O los tiempos de espera y las operaciones pendientes
	Tiempos de I/O espera del disco	Tiempo dedicado a esperar a que finalicen las operaciones del disco
	Operaciones de disco pendientes	Número de operaciones de disco en espera de ser procesadas

Note

Las tablas de las N sesiones de juego que más memoria y CPU consumen son esenciales para identificar los cuellos de botella en el rendimiento y los procesos que consumen muchos recursos y que pueden afectar al rendimiento general de la instancia. Estas clasificaciones permiten identificar rápidamente las sesiones de juego problemáticas para investigarlas más a fondo.

Panel de información general de la flota de contenedores

En este panel de control se ofrece información general de alto nivel sobre la capacidad y el uso de los recursos de su flota de contenedores. Muestra la utilización media de los recursos de todos los contenedores de una flota de contenedores administrada, incluida la reserva de CPU, el uso de la memoria y el estado del grupo de contenedores. Puede filtrar por FleetID y ubicación.

Métricas de información general de la flota de contenedores

En la siguiente tabla se muestran las métricas disponibles en el panel de control de información general de la flota de contenedores:

Panel de control	Nombre de métrica	Definición
Estado del grupo de contenedores	Grupos de contenedores de servidores de juegos activos	Grupos de contenedores que actualmente están activos y preparados para alojar sesiones de juego
	Grupos de contenedores de servidor de juegos activos	Grupos de contenedores que están activos, pero que actualmente no están alojando sesiones de juego
	Grupos de contenedores de servidor de juegos pendientes	Grupos de contenedores que están en proceso de iniciarse
	Finalización de los grupos de contenedores de servidores de juegos	Grupos de contenedores que están en proceso de cerrarse

Panel de control	Nombre de métrica	Definición
Uso de los recursos	Utilización de la CPU de contenedor	Uso promedio de CPU en todos los contenedores de la flota
	Utilización de la memoria de contenedores	Utilización media de memoria en todos los contenedores de la flota
	Reserva de CPU de contenedor	Porcentaje de recursos de CPU reservados por contenedores de toda la flota
Actividad de red	Entrada de red de contenedores	Promedio de bytes de red recibidos por los contenedores de la flota
	Salida de red de contenedores	Promedio de bytes de red enviados por los contenedores en toda la flota

Note

Las métricas de la flota de contenedores se recopilan a través de ECS Container Receiver y proporcionan visibilidad de la carga de trabajo en contenedores, centrándose en la agregación a nivel de tareas y el aislamiento de los recursos de los contenedores.

Panel de rendimiento de contenedores

Este panel proporciona métricas de rendimiento detalladas para contenedores individuales dentro de tareas específicas de ECS. Muestra métricas detalladas de los contenedores individuales, incluida la utilización de la CPU, el uso de la memoria, las E/S de la red y el rendimiento del almacenamiento. Puede filtrar por ID de contenedor específico o por tarea de ECS.

Métricas de rendimiento de contenedores

En la siguiente tabla se muestran las métricas disponibles en el panel de control de rendimiento de contenedores:

Panel de control	Nombre de métrica	Definición
Rendimiento de los recursos	Utilización de la CPU de contenedor	Utilización de la CPU del contenedor seleccionado a lo largo del tiempo
	Utilización de la memoria de contenedores	Utilización de la memoria del contenedor seleccionado a lo largo del tiempo
	Reserva de memoria del contenedor	Porcentaje de recursos de memoria reservados por el contenedor seleccionado
Rendimiento de la red	E/S de red de contenedores	Volumen de tráfico de red (bytes de entrada/salida) del contenedor seleccionado
	Conexiones de red	Número de conexiones de red activas para el contenedor seleccionado
Rendimiento del almacenamiento	Lectura y escritura de almacenamiento de contenedores	Actividad de lectura y escritura de almacenamiento para el contenedor seleccionado
	I/O Operaciones de almacenamiento en contenedores	Número de I/O operaciones de almacenamiento realizadas por el contenedor seleccionado

Note

Las métricas de rendimiento de los contenedores proporcionan una visibilidad detallada de las características de rendimiento y consumo de recursos individuales de los contenedores en las tareas de ECS.

Panel de rendimiento del servidor

El panel de rendimiento del servidor muestra métricas relacionadas con los tiempos de los servidores, la actividad de la red, la memoria y el uso de la CPU en sesiones de juego individuales.

Puedes filtrar por ID de sesión de juego y exportar las métricas directamente a Amazon CloudWatch o a [Grafana gestionada por Amazon](#).

Métricas de rendimiento del servidor

En la siguiente tabla se muestran las métricas disponibles en el panel de control de rendimiento del servidor:

Panel de control	Nombre de métrica	Definición
Horarios de los servidores	Tiempo diferencial del servidor	Diferencia de tiempo entre el tiempo de graduación del servidor actual y el anterior. Mide la constancia de la frecuencia de graduación del servidor.
	Frecuencia de graduación del servidor	Muestra el número de veces por segundo que el servidor procesa las actualizaciones
	Tiempo de graduación del servidor	La cantidad de tiempo que tarda el servidor en procesar una sola graduación o actualización
	Tiempo de graduación del servidor mundial	La cantidad de tiempo que tarda el servidor en actualizar el mundo del juego con cada graduación
Network	Connections	Número total de conexiones de red establecidas por el agente
	Red I/O (bytes)	Número total de bytes que envía y recibe el nodo a través de la red
	Red I/O (paquetes)	Cantidad total de paquetes de red enviados y recibidos por el servidor
	Pérdida de paquetes	Porcentaje de paquetes de red que se pierden durante la transmisión
Memoria	Uso de memoria (unidades)	Cantidad total de memoria que consumen los procesos del servidor

Panel de control	Nombre de métrica	Definición
	Uso de memoria física (%)	Porcentaje de memoria física total del servidor que está en uso actualmente
Uso de la CPU	Uso de la CPU (%)	Porcentaje de recursos de CPU que utiliza el proceso del servidor de juegos

Escenarios de supervisión habituales

Investigación del rendimiento en profundidad

Escenario: A host/instance tiene un rendimiento degradado debido a procesos o sesiones de juego específicos

Pasos de investigación:

- Acceda al panel Rendimiento de las instancias.
- Consulte la tabla “Las N sesiones de juego que más memoria consumen” para identificar qué procesos contribuyen en mayor medida al consumo de memoria de la instancia.
- Consulte la tabla “Las N sesiones de juego que más CPU consumen” para identificar qué procesos contribuyen en mayor medida al consumo de CPU de la instancia.
- Haga clic en los enlaces a las sesiones de juego para poder investigar más a fondo las métricas detalladas.
- Analice los tiempos de los servidores (tiempo diferencial del servidor, velocidad de graduación del servidor, tiempo de graduación del servidor, tiempo de graduación del servidor mundial) para identificar los cuellos de botella en el rendimiento.

Investigación de bloqueos del servidor de juegos

Escenario: una sesión de juego se ha bloqueado y necesita identificar la causa raíz

Pasos de investigación:

- Acceda al panel de rendimiento del servidor de la sesión de juego que se ha bloqueado.
- Consulte el uso de memoria (unidades) y el uso de memoria física (%) para determinar si el bloqueo se debió a una falta de memoria.

- Revise el uso de la CPU (%) para identificar si fue la sobrecarga de la CPU lo que provocó el bloqueo.
- Analice la red I/O (bytes) y la red I/O (paquetes) para determinar si los problemas de ancho de banda de la red contribuyeron al bloqueo.
- Examine el porcentaje de pérdida de paquetes para identificar problemas relacionados con la red.

Investigación de problemas notificados por los jugadores

Escenario: los jugadores informan de un retraso o una interrupción durante el juego

Pasos de investigación:

- Acceda al panel de rendimiento del servidor de la sesión de juego afectada.
- Revise el tiempo de graduación del servidor y el tiempo de graduación del servidor mundial para identificar los retrasos en las actualizaciones del juego.
- Compruebe la velocidad de graduación del servidor para asegurarse de que la frecuencia de actualización del servidor es constante.
- Analice el uso de la CPU (%) para identificar cuellos de botella en el procesamiento.
- Revise las métricas de uso de la memoria para identificar posibles problemas de rendimiento relacionados con la memoria.
- Compruebe I/O las métricas de la red y la pérdida de paquetes para identificar los cuellos de botella de la red.

Identificación de las variaciones de rendimiento en las diferentes compilaciones del servidor de juegos

Escenario: desea medir cómo varía el rendimiento de los juegos en las distintas compilaciones del servidor

Pasos de investigación:

- Compare las métricas de tiempo de graduación del servidor de las diferentes compilaciones para medir las variaciones en la eficiencia de procesamiento.
- Analice la coherencia de la velocidad de graduación del servidor en las diferentes compilaciones para identificar regresiones en el rendimiento.

- Revise el tiempo de graduación del servidor mundial para medir las variaciones en el rendimiento de las actualizaciones mundiales del juego.
- Compare los patrones de uso de memoria de diferentes compilaciones para identificar mejoras o regresiones en la optimización de la memoria.
- Supervise las tendencias de uso de la CPU para evaluar las variaciones en la eficiencia computacional.

Detección de retrasos y ralentización del juego

Escenario: necesita supervisar la capacidad de respuesta del servidor y la velocidad de actualización del juego

Pasos de investigación:

- Supervise el tiempo de graduación del servidor para medir la rapidez con la que el servidor procesa cada ciclo de actualización.
- Realice un seguimiento de la velocidad de graduación del servidor para asegurarse de que las actualizaciones del estado del juego por segundo son constantes.
- Analice el tiempo de graduación del servidor mundial para medir la velocidad de actualización mundial del juego, que repercute directamente en la experiencia del cliente.
- Configure alertas de variaciones en el tiempo diferencial del servidor a fin de detectar incoherencias en el rendimiento del servidor.

Evaluación comparativa de diferentes escenarios de juego

Escenario: desea identificar cómo afectan los diferentes escenarios de juego al rendimiento del servidor

Pasos de investigación:

- Compare las métricas de rendimiento del servidor con distintos números de jugadores para comprender el impacto del escalado.
- Analice las diferencias de rendimiento de los distintos modos de juego utilizando las métricas de tiempo de graduación del servidor y uso de la CPU.
- Supervise los patrones de uso de la memoria en diferentes escenarios de juego para identificar las características que consumen una gran cantidad de recursos.

- Realiza un seguimiento de I/O las métricas de la red para comprender los requisitos de ancho de banda en los diferentes escenarios de juego.
- Utilice el panel de rendimiento de las instancias para identificar qué escenarios de juego producen las sesiones de juego con un consumo de recursos más elevado.

Respuesta ante un uso de recursos elevado

Escenario: picos inusuales de uso de recursos (CPU > 85 %, memoria > 90 %)

Pasos de investigación:

Identifique los recursos afectados.

- Usa DescribeGameSessionDetails la API.
- Filtre por estado si es necesario.
- Documente las instancias afectadas.

Análisis del uso de recursos

- Revise el panel de información general de las instancias.
- Compare el uso en toda la flota.
- Compruebe los patrones históricos.

Supervisión del impacto del servidor de juegos

- Compruebe las métricas de rendimiento del servidor.
- Revise los tiempos de graduación y la pérdida de paquetes.
- Supervise las pérdidas de memoria.

Pasos de resolución

- Descargue los registros de las sesiones.
- Aborde los problemas de compilación.
- Supervise las mejoras.

Análisis de bloqueos del servidor de juegos

Escenario: existencia de varias sesiones de juego con estado de error en la flota

Pasos de investigación:

Evaluación inicial

- Acceda al panel de información general de la flota.
- Revise la tabla de sesiones bloqueadas.
- Observe los patrones de tiempos y ubicación.

Análisis del rendimiento

- Compruebe las métricas de tiempo del servidor.
- Revise el uso de recursos.
- Supervise el rendimiento de la red.

Revisión de la infraestructura

- Verifique la capacidad de la flota.
- Consulte el estado de las instancias.
- Revise las políticas de escalado.

Ruta de resolución

- Analice los registros del servidor.
- Revise la optimización del código.
- Implemente correcciones.

Optimización de la capacidad de la flota

Escenario: lanzamiento o estudio comparativo del juego

Pasos de análisis:

Uso de los recursos

- Filtre por ubicación.
- Revise las métricas P50/P95/P99.
- Analice los patrones de uso.

Análisis de los tipos de instancia

- Compare el rendimiento por tipo de instancia.
- Identifique las instancias aptas para el escalado.
- Patrones de uso de documentos.

Acciones de optimización

- Ajuste las políticas de escalado.
- Modifique los tipos de instancia.
- Actualice la configuración de la flota.

Guía para solucionar problemas

Problemas comunes y pasos de resolución

Métricas faltantes o incompletas

Síntomas

- No aparecen métricas en los paneles.
- Se recopila solo una parte de las métricas.
- Las métricas se actualizan con retraso.

Pasos de resolución

A. Verificación del estado del recopilador

Compruebe el servicio systemd:

```
sudo systemctl status gamelift-telemetry-collector
```

Revise los registros del recopilador:

```
sudo journalctl -u gamelift-telemetry-collector
```

- Confirme la configuración del recopilador.

B. Verificación del permiso de IAM

- Compruebe los permisos de los roles de instancia.
- Verifique las políticas necesarias:
 - `aps:RemoteWrite`
 - `cloudwatch:PutMetricData`
- Valide las relaciones de confianza de los roles.

C. Conectividad de la red

- Verifique el acceso al punto de conexión.
- Compruebe las reglas del grupo de seguridad.
- Red de reseñas ACLs.

Errores de autenticación

Síntomas

- Errores de autenticación SigV4.
- Mensajes de acceso denegado.
- Problemas con la actualización de credenciales.

Pasos de resolución

A. Autenticación SigV4

- Verifique las credenciales temporales.

- Compruebe la rotación de credenciales.
- Valide el perfil de instancia.

B. Acceso a AMP

- Revise la configuración del espacio de trabajo.
- Verifique la URL de escritura remota.
- Compruebe los enlaces de rol de IAM.

Problemas con los paneles

Síntomas

- Paneles vacíos.
- Puntos de datos faltantes.
- Errores de autenticación.

Pasos de resolución

A. Configuración del origen de datos

- Verifique la conexión de Prometheus.
- Compruebe la CloudWatch integración de Amazon.
- Pruebe los permisos de orígenes de datos.

B. Acceso a Grafana

- Confirme la configuración de SSO.
- Verifique la configuración de la 2FA si es necesario.
- Compruebe los permisos de usuario.

Problemas específicos de Windows

Síntomas

- Errores en el inicio del servicio.

- Brechas en la recopilación de métricas.
- Errores de permisos.

Pasos de resolución

- Verifique el estado del servicio de Windows.
- Compruebe los registros de eventos de Windows.
- Revise la configuración del recopilador.
- Valide las rutas específicas de Windows.

Registrar llamadas a la Amazon GameLift Servers API con AWS CloudTrail

Amazon GameLift Servers está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon GameLift Servers. CloudTrail captura todas las llamadas a la API Amazon GameLift Servers como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Amazon GameLift Servers y las llamadas desde el código a las operaciones de la API de Amazon GameLift Servers. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para Amazon GameLift Servers. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar a Amazon GameLift Servers qué dirección IP se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la [Guía AWS CloudTrail del usuario](#).

Amazon GameLift Servers información en CloudTrail

CloudTrail está habilitada en tu cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en Amazon GameLift Servers, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su Cuenta de AWS. Para obtener más información, consulte [Visualización de eventos con el historial de CloudTrail eventos](#).

Para mantener un registro continuo de eventos en la Cuenta de AWS, incluidos los eventos de Amazon GameLift Servers, cree un registro de seguimiento. Un rastro permite CloudTrail entregar

archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- [Introducción a la creación de registros de seguimiento](#)
- [CloudTrail servicios e integraciones compatibles](#)
- [Configuración de las notificaciones de Amazon SNS para CloudTrail](#)
- [Recibir archivos de CloudTrail registro de varias regiones](#) y [recibir archivos de CloudTrail registro de varias cuentas](#)

Todas Amazon GameLift Servers las acciones se registran CloudTrail y se documentan en la [referencia de la Amazon GameLift Servers API](#). Por ejemplo `CreateGameSession`, las llamadas `CreatePlayerSession` y `UpdateGameSession` las acciones generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el [elemento `userIdentity` de CloudTrail](#).

Descripción de las entradas de los archivos de registro de Amazon GameLift Servers

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud,

etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que muestra las `DescribeFleetAttributes` acciones `CreateFleet` y.

```
{
  "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",
        "e2InboundPermissions": [
          {
            "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"
    },
]
}
```

Registro de mensajes del servidor en Amazon GameLift Servers

Puede capturar mensajes de servidor personalizados de sus servidores de Amazon GameLift Servers en archivos de registro. La forma de configurar el registro depende de si utiliza servidores personalizados o Amazon GameLift Servers Realtime de si utiliza EC2 flotas gestionadas o flotas de contenedores (consulte las subsecciones correspondientes de este capítulo).

Temas

- [Diferencias de registro: flotas gestionadas EC2 y flotas de contenedores](#)
- [Registro de mensajes del servidor \(servidores personalizados\)](#)

Diferencias de registro: flotas gestionadas EC2 y flotas de contenedores

El comportamiento de registro difiere considerablemente entre las EC2 flotas gestionadas y las flotas de contenedores. Comprender estas diferencias es crucial a la hora de migrar EC2 a contenedores o elegir el tipo de flota adecuado para sus necesidades de registro.

Resumen de las diferencias clave

Característica	Flotas gestionadas EC2	Flotas de contenedores
Almacenamiento de registros	Se carga automáticamente en Amazon S3 al finalizar la sesión de juego	Debe configurar Amazon CloudWatch Logs o una solución de registro personalizada
GetGameSessionLogUrl API	Disponible: devuelve la URL de Amazon S3 para descargar el registro	No disponible: los registros no se almacenan automáticamente en Amazon S3

Característica	Flotas gestionadas EC2	Flotas de contenedores
Retención de registros	14 días en Amazon S3 (automático)	Depende de la configuración de registro
Supervisión en tiempo real	Limitada: los registros solo están disponibles una vez que finaliza la sesión de juego	Disponible con la integración de Amazon CloudWatch Logs
Complejidad de la configuración	Automática: no se requiere ninguna configuración adicional	Requiere una configuración de registro explícita

Registro de EC2 flota gestionado

En el caso de EC2 las flotas gestionadas, Amazon GameLift Servers proporciona una gestión automática de los registros:

- Carga automática: los registros del servidor se cargan automáticamente en Amazon S3 cuando finaliza una sesión de juego.
- GetGameSessionLogUrl API: utilice esta API para recuperar una URL prefirmada para descargar registros de Amazon S3
- Conservación durante 14 días: los registros se retienen en Amazon S3 durante 14 días antes de su eliminación automática.
- Límites de tamaño: los archivos de registro tienen límites de tamaño por sesión de juego (consulte [Puntos de conexión y cuotas de Amazon GameLift Servers](#)).

Para obtener más información, consulte [the section called “Registro para servidores personalizados”](#) la [referencia de la GetGameSessionLogUrl API](#).

Registro de flotas de contenedores

Las flotas de contenedores requieren una configuración explícita del registro:

- Sin carga automática en Amazon S3: los registros no se cargan automáticamente en Amazon S3.
- GetGameSessionLogUrl no disponible: esta API no funciona con flotas de contenedores
- Integración con Amazon CloudWatch Logs: configure su contenedor para enviar registros a Amazon CloudWatch Logs para un registro centralizado

- Soluciones de registro personalizadas: implemente su propia infraestructura de registro mediante controladores de registro o contenedores asociados.
- Acceso en tiempo real: con la configuración adecuada, es posible acceder a los registros en tiempo real durante las sesiones de juego.

Para ver opciones detalladas de registro de contenedores, consulte [Funcionamiento de las flotas de contenedores](#).

Consideraciones sobre la migración

Al migrar de una flota gestionada EC2 a una de contenedores, tenga en cuenta estos cambios en el registro:

- Actualizar el código de recuperación de registros: sustituye las llamadas a la `GetGameSessionLogUrl` API por consultas de Amazon CloudWatch Logs o tu solución de registro personalizada
- Configurar la retención de registros: configura las políticas de retención adecuadas en Amazon CloudWatch Logs o en tu sistema de registro
- Implementación de la supervisión en tiempo real: utilice el acceso a los registros en tiempo real para beneficiarse de una mejor observabilidad.
- Revise el volumen y los costes de los CloudWatch registros: los precios de Amazon Logs difieren del almacenamiento de Amazon S3 incluido en EC2 las flotas gestionadas

Configuración del registro de contenedores recomendada

Para las flotas de contenedores, recomendamos lo siguiente:

1. Integración de Amazon CloudWatch Logs: configura tu definición de contenedor para usar el controlador de `awslogs` registro
2. Registro estructurado: utilice formatos de registro estructurados (JSON) para mejorar la capacidad de búsqueda y el análisis.
3. Niveles de registro: implemente niveles de registro adecuados para controlar el grado de detalle y los costos.
4. Políticas de conservación: establezca periodos de conservación en función de sus necesidades de conformidad y depuración.

Registro de mensajes del servidor (servidores personalizados)

Puede capturar mensajes de servidor personalizados de sus servidores personalizados de Amazon GameLift Servers en archivos de registro.

Important

El tamaño de un archivo de registro está sujeto a un límite por sesión de juego (consulte [Puntos de conexión y cuotas de Amazon GameLift Servers](#) en la Referencia general de AWS). Cuando finaliza una sesión de juego, Amazon GameLift Servers carga los registros del servidor en Amazon Simple Storage Service (Amazon S3). Amazon GameLift Servers no cargará los registros que superen el límite. Los registros pueden crecer muy rápido y superar el límite de tamaño. Debe supervisar los registros y limitar la salida del registro a los mensajes necesarios únicamente.

Configuración del registro para servidores personalizados

Con servidores personalizados de Amazon GameLift Servers, usted escribe su propio código para realizar el registro, que configura como parte de la configuración del proceso del servidor. Amazon GameLift Servers utiliza su configuración de registro para identificar los archivos que debe cargar en S3 al final de cada sesión de juego.

Las siguientes instrucciones muestran cómo configurar el registro mediante ejemplos de código simplificados:

C++

Configuración del registro (C++)

1. Cree un vector de cadenas que sean rutas de directorio a los archivos de registro del servidor de juegos.

```
std::string serverLog("serverOut.log");           // Example server log file
std::vector<std::string> logPaths;
logPaths.push_back(serverLog);
```

2. Proporcione su vector como el [LogParameters](#) de su [ProcessParameters](#) objeto.

```
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));
```

- Proporcione el [ProcessParameters](#) objeto cuando llame a [ProcessReady\(\)](#).

```
Aws::GameLift::GenericOutcome outcome =
    Aws::GameLift::Server::ProcessReady(processReadyParameter);
```

Para ver un ejemplo más completo, consulte [ProcessReady\(\)](#).

C#

Para configurar el registro (C#), realice el siguiente procedimiento:

- Cree una lista de cadenas que sean rutas de directorio a los archivos de registro del servidor de juegos.

```
List<string> logPaths = new List<string>();
logPaths.Add("C:\\game\\serverOut.txt");    // Example of a log file that the
game server writes
```

- Proporcione su lista como la [LogParameters](#) de su [ProcessParameters](#) objeto.

```
var processReadyParameter = new ProcessParameters(
    this.OnGameSession,
    this.OnProcessTerminate,
    this.OnHealthCheck,
    this.OnGameSessionUpdate,
    port,
    new LogParameters(logPaths));
```

- Proporcione el [ProcessParameters](#) objeto cuando llame a [ProcessReady\(\)](#).

```
var processReadyOutcome =
```

```
GameLiftServerAPI.ProcessReady(processReadyParameter);
```

Para ver un ejemplo más completo, consulte [ProcessReady\(\)](#).

Escritura en los registros

Los archivos de registro existirán después de que comience el proceso del servidor. Puede escribir en los registros mediante cualquier método para escribir en los archivos. Para capturar todos los resultados estándar y de errores del servidor, reasigne los flujos de salida a los archivos de registro, como en los ejemplos siguientes:

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"));
```

Acceso a los registros del servidor

El acceso a los registros varía en función del tipo de flota:

Flotas de EC2 administradas

Cuando una sesión de juego finaliza, Amazon GameLift Servers almacena automáticamente los registros en un bucket de S3 y los retiene durante 14 días. Para obtener la ubicación de los registros de una sesión de juego, puedes usar la operación de la [GetGameSessionLogUrl](#) API. Para descargar los registros, utilice la URL que devuelve la operación.

Como alternativa, puede configurar su propia solución de registro configurando el servidor de juegos para que envíe los registros directamente al servicio de registro o la ubicación de almacenamiento de su preferencia. Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Flotas de contenedores

Las flotas de contenedores capturan flujos de error y salida estándar de todos los contenedores. Puede configurar el registro mediante una de estas opciones:

- Guarda la salida del contenedor como flujos de CloudWatch registro en un grupo de registros específico
- Guardar la salida del contenedor en un bucket de almacenamiento de S3
- Desactivar el registro (la salida del contenedor no se guarda)

Para obtener información detallada sobre la configuración de las opciones de registro al crear flotas de contenedores, consulte [LogConfiguration](#) la referencia de la Amazon GameLift Servers API.

Seguridad en Amazon GameLift Servers

Si utiliza Amazon GameLift Servers FleetIQ como característica independiente con Amazon EC2, consulte [Seguridad en Amazon EC2](#) en la Guía del usuario de Amazon EC2.

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

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

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

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida a la hora de utilizar Amazon GameLift Servers. Los siguientes temas le mostrarán cómo configurar Amazon GameLift Servers para satisfacer sus objetivos de seguridad y de conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus Amazon GameLift Servers recursos.

Temas

- [Protección de los datos en Amazon GameLift Servers](#)
- [Administración de identidades y accesos en Amazon GameLift Servers](#)
- [Registro y supervisión con Amazon GameLift Servers](#)
- [Validación de conformidad en Amazon GameLift Servers](#)
- [Resiliencia en Amazon GameLift Servers](#)
- [Seguridad de la infraestructura en Amazon GameLift Servers](#)

- [Configuración y análisis de vulnerabilidades en Amazon GameLift Servers](#)
- [Prácticas recomendadas de seguridad para Amazon GameLift Servers](#)

Protección de los datos en Amazon GameLift Servers

Si utilizas Amazon GameLift Servers FleetIQ una función independiente con Amazon EC2, consulta [Seguridad en Amazon EC2 en](#) la Guía del EC2 usuario de Amazon.

El modelo de [responsabilidad AWS compartida modelo](#) se aplica a la protección de datos en Amazon GameLift Servers. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el [Modelo de responsabilidad compartida de AWS y GDPR](#) en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte [Cómo trabajar con CloudTrail senderos](#) en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información

sobre los puntos de conexión de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 140-3](#).

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con Amazon GameLift Servers o Servicios de AWS utiliza la consola, la API o. AWS CLI AWS SDKs Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Los datos específicos de Amazon GameLift Servers se administran de la siguiente manera:

- Las compilaciones y los scripts del servidor de juegos que cargue en Amazon GameLift Servers se almacenan en Amazon S3. El cliente no tiene acceso directo a estos datos una vez cargados. Un usuario autorizado puede obtener acceso temporal para cargar archivos, pero no puede verlos ni actualizarlos en Amazon S3 directamente. Para eliminar scripts y compilaciones, utilice la consola de Amazon GameLift Servers o la API de servicio.
- Los datos de registro de la sesión de juego se almacenan en Amazon S3 durante un periodo limitado una vez completada la sesión de juego. Para obtener acceso a los datos del registro, los usuarios autorizados pueden descargarlos a través de un enlace en la consola de Amazon GameLift Servers o mediante llamadas a la API de servicio.
- Los datos de las métricas y eventos se almacenan en Amazon GameLift Servers y se puede obtener acceso a ellos a través de la consola de Amazon GameLift Servers o mediante llamadas a la API de servicio. Los datos pueden recuperarse en flotas, instancias, ubicaciones de sesiones de juego, tickets de emparejamiento, sesiones de juego y sesiones de jugador. También se puede acceder a los datos a través de Amazon CloudWatch y CloudWatch Events.
- Los datos suministrados por el cliente se almacenan en Amazon GameLift Servers. Los usuarios autorizados pueden obtener acceso a ellos mediante llamadas a la API de servicio. La información potencialmente confidencial puede incluir datos del jugador, datos de sesiones de jugador y de juego (incluida la información de conexión) y datos del emparejador, entre otros.

 Note

Si incluyes un jugador personalizado IDs en tus solicitudes, se espera que estos valores sean anónimos UUIDs y no contengan información que identifique al jugador.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al [modelo de responsabilidad compartida de AWS y GDPR](#) en el blog de seguridad de AWS .

Cifrado en reposo

El cifrado en reposo de los datos específicos de Amazon GameLift Servers se administra de la siguiente manera:

- Las compilaciones y los scripts del servidor de juegos se almacenan en buckets de Amazon S3 con cifrado del servidor.
- Los datos suministrados por el cliente se almacenan en Amazon GameLift Servers en formato cifrado.

Cifrado en tránsito

Las conexiones Amazon GameLift Servers APIs se realizan a través de una conexión segura (SSL) y se autentican mediante la [versión 4 de AWS Signature](#) (cuando se conecta a través de la AWS CLI o el AWS SDK, la firma se gestiona automáticamente). La autenticación se administra mediante las políticas de acceso definidas por IAM para las credenciales de seguridad que se utilizan a fin de realizar la conexión.

La comunicación directa entre los clientes y servidores de juego es la siguiente:

- Para los servidores de juegos personalizados alojados en recursos de Amazon GameLift Servers, la comunicación no implica el servicio Amazon GameLift Servers. El cifrado de esta comunicación es responsabilidad del cliente. Puede utilizar flotas habilitadas con TLS para que sus clientes de juegos autenticuen el servidor de juegos al conectarse y para cifrar toda comunicación entre el cliente y el servidor de juegos.
- Los certificados para las flotas con TLS se crean al mismo tiempo que la flota y sus fechas de caducidad se basan en la fecha de creación de la flota.

- Para Amazon GameLift Servers Realtime con la generación de certificados TLS habilitada, el tráfico entre el cliente de juegos y los servidores Realtime que utilizan el SDK del cliente de Realtime se cifra en tránsito. El tráfico TCP se cifra con TLS 1.2 y el tráfico UDP se cifra con DTLS 1.2.

Note

Para cumplir con los requisitos de la autoridad de certificación para los certificados TLS, Amazon GameLift Servers ajustará la duración máxima de los certificados para las flotas configuradas para generar un certificado. La vida útil de los certificados comienza con la creación de la flota y se modificará según el siguiente calendario:

- Hasta el 11 de marzo de 2026, la vida útil máxima de un certificado TLS emitido es de 398 días.
- A partir del 1 de marzo de 2026, la vida útil máxima de un certificado TLS emitido será de 200 días.
- A partir del 1 de marzo de 2027, la vida útil máxima de un certificado TLS emitido será de 100 días.
- A partir del 1 de marzo de 2029, la vida útil máxima de un certificado TLS emitido será de 47 días.

Para garantizar la renovación de sus certificados y mantener los entornos de ejecución de los servidores de up-to-date juegos, Amazon GameLift Servers le recomienda sustituir periódicamente sus flotas de servidores de juegos.

Privacidad del tráfico entre redes

Puede obtener acceso remoto a sus instancias de Amazon GameLift Servers de forma segura. Para las instancias que utilizan Linux, SSH proporciona un canal de comunicaciones seguro para el acceso remoto. Para las instancias que ejecutan Windows, utilice un cliente de protocolo de escritorio remoto (RDP). Con Amazon GameLift ServersFleetIQ, el acceso remoto a sus instancias mediante AWS Systems Manager Session Manager y Run Command se cifra mediante TLS 1.2, y las solicitudes para crear una conexión se firman mediante SiGv4. Para obtener ayuda con la conexión a una instancia administrada de Amazon GameLift Servers, consulte [Conexión remota a instancias de flota de Amazon GameLift Servers](#).

Administración de identidades y accesos en Amazon GameLift Servers

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Amazon GameLift Servers. La IAM es un Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)
- [Administración del acceso con políticas](#)
- [Cómo Amazon GameLift Servers funciona con IAM](#)
- [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#)
- [Solución de problemas de identidades de Amazon GameLift Servers y accesos](#)
- [AWS políticas gestionadas para Amazon GameLift Servers](#)

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte [Solución de problemas de identidades de Amazon GameLift Servers y accesos](#)).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte [Cómo Amazon GameLift Servers funciona con IAM](#)).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#)).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte [Cómo iniciar sesión en la Cuenta de AWS](#) en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte [AWS Signature Version 4 para solicitudes de API](#) en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear un Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso completo a todos Servicios de AWS los recursos. Se recomienda encarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte [Tareas que requieren credenciales de usuario raíz](#) en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, del proveedor de identidades web o al Directory Service que se accede Servicios de AWS mediante credenciales de una fuente de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, se recomienda AWS IAM Identity Center. Para obtener más información, consulte [¿Qué es el Centro de identidades de IAM?](#) en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un [usuario de IAM](#) es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte [Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales](#) en la Guía del usuario de IAM.

Un [grupo de IAM](#) especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte [Casos de uso para usuarios de IAM](#) en la Guía del usuario de IAM.

Roles de IAM

Un [Rol de IAM](#) es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol [cambiando de un rol de usuario a uno de IAM \(consola\)](#) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte [Métodos para asumir un rol](#) en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.

Administración del acceso con políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte [Información general de políticas de JSON](#) en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte [Definición de permisos de IAM personalizados con políticas administradas por el cliente](#) en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte [Selección entre políticas administradas y políticas insertadas](#) en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe [especificar una entidad principal](#) en una política basada en recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que conceden los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte [Límites de permisos para las entidades de IAM](#) en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte [Políticas de control de servicios](#) en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte [Políticas de control de recursos \(RCPs\)](#) en la Guía del AWS Organizations usuario.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte [Políticas de sesión](#) en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la [lógica de evaluación de políticas](#) en la Guía del usuario de IAM.

Cómo Amazon GameLift Servers funciona con IAM

Antes de utilizar IAM para administrar el acceso a Amazon GameLift Servers, conozca qué características de IAM se pueden utilizar con Amazon GameLift Servers.

Características de IAM que puede utilizar con Amazon GameLift Servers

Característica de IAM	Compatibilidad con Amazon GameLift Servers
Políticas basadas en identidad	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACLs	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo Amazon GameLift Servers funcionan otros AWS servicios con la mayoría de las funciones de IAM, consulte [AWS los servicios que funcionan con IAM en la Guía del usuario de IAM](#).

Políticas de Amazon GameLift Servers basadas en identidades

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte [Definición de permisos de IAM personalizados con políticas administradas por el cliente](#) en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte [Referencia de los elementos de la política de JSON de IAM](#) en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades de Amazon GameLift Servers

Para ver ejemplos de políticas basadas en identidad de Amazon GameLift Servers, consulte [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#).

Políticas basadas en recursos de Amazon GameLift Servers

Admite políticas basadas en recursos: no

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política basada en recursos. Los directores pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte [Cross account resource access in IAM](#) en la Guía del usuario de IAM.

Acciones de política para Amazon GameLift Servers

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para obtener una lista de las acciones de Amazon GameLift Servers, consulte [Acciones definidas por Amazon GameLift Servers](#) en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Amazon GameLift Servers utilizan el siguiente prefijo antes de la acción:

```
gamelift
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [  
    "gamelift:action1",  
    "gamelift:action2"  
]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones. Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "gamelift:Describe*"
```

Para ver ejemplos de políticas basadas en identidad de Amazon GameLift Servers, consulte [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#).

Recursos de políticas para Amazon GameLift Servers

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el [Nombre de recurso de Amazon \(ARN\)](#). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para obtener una lista de los tipos de Amazon GameLift Servers recursos y sus tipos ARNs, consulte [los recursos definidos Amazon GameLift Servers](#) en la Referencia de autorización de servicios.

Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte [Acciones definidas por Amazon GameLift Servers](#).

Algunos recursos de Amazon GameLift Servers tienen valores ARN, lo que permite administrar su acceso mediante políticas de IAM. El recurso de flota de Amazon GameLift Servers tiene un ARN con la sintaxis siguiente:

```
arn:${Partition}:gamelift:${Region}:${Account}:fleet/${FleetId}
```

Para obtener más información sobre el formato de ARNs, consulte [Amazon Resource Names \(ARNs\)](#) en Referencia general de AWS.

Por ejemplo, para especificar la flota `fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa` en su instrucción, utilice el siguiente ARN:

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/  
fleet-2222bbbb-33cc-44dd-55ee-6666ffff77aa"
```

Para especificar todas las flotas que pertenecen a una cuenta específica, utilice el carácter comodín (*):

```
"Resource": "arn:aws:gamelift:us-west-2:123456789012:fleet/*"
```

Para ver ejemplos de políticas basadas en identidad de Amazon GameLift Servers, consulte [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#).

Claves de condición de políticas para Amazon GameLift Servers

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan [operadores de condición](#), tales como

igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición AWS globales, consulte las claves de [contexto de condición AWS globales](#) en la Guía del usuario de IAM.

Para obtener una lista de las claves de condición de Amazon GameLift Servers, consulte [Claves de condición para Amazon GameLift Servers](#) en la Referencia de autorizaciones de servicio. Para obtener más información sobre las acciones y los recursos con los que puede utilizar una clave de condición, consulte [Acciones definidas por Amazon GameLift Servers](#).

Para ver ejemplos de políticas basadas en identidad de Amazon GameLift Servers, consulte [Ejemplos de políticas basadas en identidades de Amazon GameLift Servers](#).

ACLs en Amazon GameLift Servers

Soportes ACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Amazon GameLift Servers

Admite ABAC (etiquetas en las políticas): sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos denominados etiquetas. Puede adjuntar etiquetas a las entidades y AWS los recursos de IAM y, a continuación, diseñar políticas de ABAC para permitir las operaciones cuando la etiqueta del director coincida con la etiqueta del recurso.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el [elemento de condición](#) de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte [Definición de permisos con la autorización de ABAC](#) en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte [Uso del control de acceso basado en atributos \(ABAC\)](#) en la Guía del usuario de IAM.

Para obtener un ejemplo de política basada en identidad que limita el acceso a un recurso basado en las etiquetas de ese recurso, consulte [Ver las flotas de Amazon GameLift Servers basadas en etiquetas](#).

Uso de credenciales temporales con Amazon GameLift Servers

Compatibilidad con credenciales temporales: sí

Las credenciales temporales proporcionan acceso a AWS los recursos a corto plazo y se crean automáticamente cuando se utiliza la federación o se cambia de rol. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte [Credenciales de seguridad temporales en IAM](#) y [Servicios de AWS que funcionan con IAM](#) en la Guía del usuario de IAM.

Permisos de entidades principales entre servicios de Amazon GameLift Servers

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos del principal que llama y los que solicitan Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Para obtener información detallada sobre las políticas a la hora de realizar solicitudes de FAS, consulte [Forward access sessions](#).

Roles de servicio para Amazon GameLift Servers

Compatible con roles de servicio: sí

Un rol de servicio es un [rol de IAM](#) que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte [Crear un rol para delegar permisos a un Servicio de AWS](#) en la Guía del usuario de IAM.

Warning

Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Amazon GameLift Servers. Edite los roles de servicio sólo cuando Amazon GameLift Servers proporcione orientación para hacerlo.

Permita que sus servidores Amazon GameLift Servers de juegos alojados accedan a otros AWS recursos, como una AWS Lambda función o una base de datos de Amazon DynamoDB. Como

los servidores de juegos están alojados en flotas que las Amazon GameLift Servers administran, necesitas un rol de servicio que te dé acceso Amazon GameLift Servers limitado al resto de tus recursos. AWS Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Roles vinculados a servicios de Amazon GameLift Servers

Compatibilidad con roles vinculados al servicio: no

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para obtener más información acerca de cómo crear o administrar roles vinculados a servicios, consulte [Servicios de AWS que funcionan con IAM](#) en la Guía de usuario de IAM. Busque un servicio en la tabla que incluya Yes en la columna Roles vinculados al servicio. Elija el vínculo Sí para ver la documentación sobre el rol vinculado a un servicio en cuestión.

Ejemplos de políticas basadas en identidades de Amazon GameLift Servers

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de Amazon GameLift Servers. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte [Creación de políticas de IAM \(consola\)](#) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos Amazon GameLift Servers, incluido el formato de cada uno de los tipos de recursos, consulte [las claves de condición, recursos y acciones de la Referencia Amazon GameLift Servers](#) de autorización de servicios. ARNs

Temas

- [Prácticas recomendadas sobre las políticas](#)
- [Mediante la consola de Amazon GameLift Servers](#)
- [Cómo permitir a los usuarios consultar sus propios permisos](#)
- [Permitir acceso a los jugadores a las sesiones de juego](#)
- [Habilitación del acceso a una cola de Amazon GameLift Servers](#)

- [Ver las flotas de Amazon GameLift Servers basadas en etiquetas](#)
- [Acceso a un archivo de compilación de un juego en Amazon S3](#)

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de Amazon GameLift Servers de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las [políticas administradas por AWS](#) o las [políticas administradas por AWS para funciones de tarea](#) en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte [Políticas y permisos en IAM](#) en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte [Elementos de la política de JSON de IAM: Condición](#) en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte [Validación de políticas con el Analizador de acceso de IAM](#) en la Guía del usuario de IAM.

- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte [Acceso seguro a la API con MFA](#) en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte [Prácticas recomendadas de seguridad en IAM](#) en la Guía del usuario de IAM.

Mediante la consola de Amazon GameLift Servers

Para acceder a la consola de Amazon GameLift Servers, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de su cuenta. Amazon GameLift Servers Cuenta de AWS Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

Para asegurarse de que esas entidades todavía pueden utilizar la consola de Amazon GameLift Servers, añada permisos para los usuarios y grupos con la sintaxis en los siguientes ejemplos y en [Ejemplos de permisos de administración](#). Para obtener más información, consulte [Configuración de permisos de usuario para Amazon GameLift Servers](#).

Los usuarios que trabajan con operaciones AWS CLI directas o Amazon GameLift Servers mediante AWS API no requieren permisos mínimos de consola. En su lugar, puede limitar el acceso únicamente a las operaciones que el usuario debe realizar. Por ejemplo, un usuario jugador que actúa en nombre de los clientes de juegos necesita acceso para solicitar sesiones de juego, ubicar a los jugadores en los juegos y realizar otras tareas.

Para obtener información sobre los permisos necesarios para utilizar todas las características de la consola de Amazon GameLift Servers, consulte la sintaxis de permisos para administradores en [Ejemplos de permisos de administración](#).

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o 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": "*"
  }
]
}

```

Permitir acceso a los jugadores a las sesiones de juego

Para colocar a los jugadores en las sesiones de juego, los clientes de juegos y los servicios de backend necesitan permisos. Para ver ejemplos de políticas para estos escenarios, consulte [Ejemplos de permisos de usuario de un jugador](#).

Habilitación del acceso a una cola de Amazon GameLift Servers

El siguiente ejemplo proporciona a un usuario acceso a colas específicas de Amazon GameLift Servers.

Esta política otorga al usuario permisos para añadir, actualizar y eliminar destinos de colas con las siguientes acciones: `gamelift:UpdateGameSessionQueue`, `gamelift>DeleteGameSessionQueue` y `gamelift:DescribeGameSessionQueues`. Tal y como se muestra, esta política utiliza el elemento `Resource` para limitar el acceso a una sola cola: `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"
    }
  ]
}
```

Ver las flotas de Amazon GameLift Servers basadas en etiquetas

Puede utilizar las condiciones de su política basada en identidad para controlar el acceso a los recursos de Amazon GameLift Servers basados en etiquetas. Este ejemplo muestra cómo puede crear una política que permita ver una flota si la etiqueta `Owner` coincide con el nombre de usuario del usuario. Esta política también proporciona los permisos necesarios para completar esta operación en la consola.

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}"}
      }
    }
  ]
}
```

Acceso a un archivo de compilación de un juego en Amazon S3

Después de integrar el servidor de juegos con Amazon GameLift Servers, cargue los archivos de compilación en Amazon S3. Para que Amazon GameLift Servers acceda a los archivos de compilación, utilice la siguiente política.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],

```

```
    "Resource": "arn:aws:s3::bucket-name/object-name"
  }
]
}
```

Para obtener más información sobre la carga de archivos de juego de Amazon GameLift Servers, consulte, [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#).

Solución de problemas de identidades de Amazon GameLift Servers y accesos

Usa la siguiente información para ayudarte a diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con Amazon GameLift Servers y AWS Identity and Access Management (IAM).

Temas

- [No tengo autorización para realizar una acción en Amazon GameLift Servers](#)
- [No estoy autorizado a realizar lo siguiente: PassRole](#)
- [Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis Amazon GameLift Servers recursos](#)

No tengo autorización para realizar una acción en Amazon GameLift Servers

Si Consola de administración de AWS te indica que no estás autorizado a realizar una acción, ponte en contacto con el administrador de tu AWS cuenta para obtener ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM, mateojackson, intenta utilizar la consola para ver detalles sobre una cola, pero no tiene permisos de `gamelift:DescribeGameSessionQueues`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
gamelift:DescribeGameSessionQueues on resource: examplequeue123
```

En este caso, Mateo pide a su administrador que actualice sus políticas para poder tener acceso de lectura al recurso `examplequeue123` mediante la acción `gamelift:DescribeGameSessionQueues`.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a Amazon GameLift Servers.

Algunos Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir la función al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon GameLift Servers. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis Amazon GameLift Servers recursos

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si Amazon GameLift Servers admite estas características, consulte [Cómo Amazon GameLift Servers funciona con IAM](#).

- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte [Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario](#) de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo [proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros](#) en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte [Proporcionar acceso a usuarios autenticados externamente \(identidad federada\)](#) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.

AWS políticas gestionadas para Amazon GameLift Servers

Una política AWS administrada es una política independiente creada y administrada por AWS. Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir [políticas administradas por el cliente](#) específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte [Políticas administradas por AWS](#) en la Guía del usuario de IAM.

AWS política gestionada: GameLiftContainerFleetPolicy

Puede asociar GameLiftContainerFleetPolicy a sus roles de IAM.

La política otorga permisos para realizar acciones de computación en una flota de contenedores de Amazon GameLift Servers. Una flota de contenedores es un conjunto de recursos de alojamiento que

se Amazon GameLift Servers gestiona por usted. Amazon GameLift Serversnecesita permisos para conectarse al Amazon GameLift Servers servicio y a otros AWS servicios en su nombre.

Al crear una flota de contenedores conAmazon GameLift Servers, proporciona una función de servicio de IAM con la política GameLiftContainerFleetPolicy gestionada adjunta. Para obtener instrucciones sobre cómo crear este rol de servicio, consulte [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Para obtener más información, consulte [GameLiftContainerFleetPolicy](#).

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `cloudwatch`— Permite Amazon GameLift Servers escribir los registros de las sesiones del juego en una secuencia de registros de Amazon CloudWatch Events de tu AWS cuenta.
- `cloudwatch` — Permite Amazon GameLift Servers crear un grupo de CloudWatch registros para organizar los datos de las sesiones de juego en el flujo de registro.
- `s3`— Permite Amazon GameLift Servers escribir los registros de las sesiones del juego en un depósito de Amazon Simple Storage Service de tu AWS cuenta.
- `s3`— Permite Amazon GameLift Servers recuperar el Región de AWS lugar donde reside un bucket de Amazon S3 específico mediante la acción de la `APIs3:GetBucketLocation`.
- `gamelift` : permite a Amazon GameLift Servers recuperar un token de autenticación que permite a un servidor de juegos alojado comunicarse con el servicio Amazon GameLift Servers a través de su cuenta de AWS .

Amazon GameLift Serversactualizaciones de las políticas AWS administradas

Consulte los detalles sobre las actualizaciones de las políticas AWS administradas Amazon GameLift Servers desde que este servicio comenzó a rastrear estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la [página de notas de la versión de Amazon GameLift Servers](#).

Cambio	Descripción	Fecha
GameLiftContainerFleetPolicy : cambio	Amazon GameLift Serversse han añadido nuevos permisos	5 de febrero de 2024

Cambio	Descripción	Fecha
	para recuperar el Región de AWS de un bucket de Amazon S3.	
GameLiftContainerFleetPolicy: política nueva	Amazon GameLift Servers ha añadido nuevos permisos para permitir que los contenedores de servidores de juegos se ejecuten en flotas administradas por Amazon GameLift Servers.	12 de noviembre de 2024
Amazon GameLift Servers comenzó el seguimiento de los cambios	Amazon GameLift Servers comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas.	12 de noviembre de 2024

Registro y supervisión con Amazon GameLift Servers

La supervisión es un aspecto importante a la hora de mantener la fiabilidad, la disponibilidad y el rendimiento de Amazon GameLift Servers y de las soluciones de AWS. Debe recopilar datos de supervisión de todas las partes de la solución de AWS para que, dado el caso, le resulte más sencillo depurar un error multipunto.

AWS y Amazon GameLift Servers proporcionan varias herramientas para supervisar los recursos de alojamiento de juegos y responder a posibles incidentes.

Alarmas de Amazon CloudWatch

Con las alarmas de Amazon CloudWatch, puede observar una métrica determinada durante el periodo que especifique. Si la métrica supera un determinado umbral, se envía una notificación a un tema de Amazon SNS o a una política de AWS Auto Scaling. Las alarmas de CloudWatch se activan cuando su estado cambia y se mantiene durante un número de periodos especificado, no por estar en un estado determinado. Para obtener más información, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

Registros de AWS CloudTrail

CloudTrail proporciona un registro de las acciones que realiza un usuario, un rol o un servicio de AWS en Amazon GameLift Servers. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon GameLift Servers, la dirección IP desde la que se realizó, quién la realizó y cuándo, etc. Para obtener más información, consulte [Registrar llamadas a la Amazon GameLift Servers API con AWS CloudTrail](#).

Validación de conformidad en Amazon GameLift Servers

Amazon GameLift Servers no está en el ámbito de los programas de conformidad de AWS.

Para saber si un Servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte [Servicios de AWS incluidos por programa de conformidad](#) y escoja el programa de conformidad que le interese. Para obtener información general, consulte [Programas de conformidad de AWS](#).

Puedes descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte [Descarga de informes en AWS Artifact](#).

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la confidencialidad de los datos, los objetivos de conformidad de su empresa, así como de la legislación y los reglamentos aplicables. Para obtener más información sobre la responsabilidad de conformidad al usar Servicios de AWS, consulte la [Documentación de seguridad de AWS](#).

Resiliencia en Amazon GameLift Servers

Si utiliza Amazon GameLift Servers FleetIQ como característica independiente con Amazon EC2, consulte [Seguridad en Amazon EC2](#) en la Guía del usuario de Amazon EC2.

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. Las regiones de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad y las regiones de AWS, consulte [Infraestructura global de AWS](#).

Además de la infraestructura global de AWS, Amazon GameLift Servers ofrece las siguientes características que le ayudan con sus necesidades de resiliencia de los datos:

- **Colas multirregionales:** las colas de sesiones de juego de Amazon GameLift Servers se utilizan para ubicar nuevas sesiones de juego con los recursos de alojamiento disponibles. Las colas que abarcan varias regiones pueden redirigir las ubicaciones de sesiones de juego en caso de que se produzca una interrupción regional. Para obtener más información y conocer las prácticas recomendadas sobre la creación de colas de sesiones de juego, consulte [Personalización de colas de sesiones de juego](#).
- **Escalado automático de la capacidad:** mantenga el estado y la disponibilidad de sus recursos de alojamiento con ayuda de las herramientas de escalado de Amazon GameLift Servers. Estas herramientas ofrecen una amplia gama de opciones que le permiten ajustar la capacidad de la flota para adaptarla a las necesidades del juego y de los jugadores. Para obtener más información sobre el escalado, consulte [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#).
- **Distribución entre instancias:** Amazon GameLift Servers distribuye el tráfico entrante entre varias instancias según el tamaño de la flota. Como práctica recomendada, los juegos en producción deben tener varias instancias para mantener la disponibilidad en caso de que una instancia deje de estar en buen estado o no responda.
- **Almacenamiento en Amazon S3:** las compilaciones y los scripts de los servidores de juegos que se cargan en Amazon GameLift Servers se almacenan en Amazon S3 mediante la clase de almacenamiento estándar, que utiliza varias replicasiones de centro de datos para aumentar la resiliencia. Los registros de sesiones de juego también se almacenan en Amazon S3 con la clase de almacenamiento estándar.

Seguridad de la infraestructura en Amazon GameLift Servers

Si utiliza Amazon GameLift Servers FleetIQ como característica independiente con Amazon EC2, consulte [Seguridad en Amazon EC2](#) en la Guía del usuario de Amazon EC2.

Al tratarse de un servicio administrado, Amazon GameLift Servers está protegido por los procedimientos de seguridad de red globales de AWS que se describen en el documento técnico [Amazon Web Services: información general sobre procesos de seguridad](#).

Puede utilizar llamadas a la API publicadas en AWS para obtener acceso a Amazon GameLift Servers a través de la red. Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.2 o con una versión posterior. Recomendamos TLS 1.3 o una versión posterior. Los clientes

también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

El servicio de Amazon GameLift Servers coloca todas las flotas en nubes virtuales privadas (VPC) de Amazon para que cada flota exista en un área aislada lógicamente en la nube de AWS. Puede utilizar las políticas de Amazon GameLift Servers para controlar el acceso desde puntos de enlace de la VPC o desde VPC específicas. Este proceso aísla con eficacia el acceso de red a un recurso de Amazon GameLift Servers determinado únicamente desde la VPC específica de la red de AWS. Al crear una flota, se especifica un intervalo de números de puerto y direcciones IP. Estos intervalos limitan el modo en que el tráfico entrante puede obtener acceso a los servidores de juegos alojados en la VPC de una flota. Utilice las prácticas recomendadas de seguridad estándar a la hora de elegir la configuración de acceso a la flota.

Configuración y análisis de vulnerabilidades en Amazon GameLift Servers

Si utiliza Amazon GameLift Servers FleetIQ como característica independiente con Amazon EC2, consulte [Seguridad en Amazon EC2](#) en la Guía del usuario de Amazon EC2.

La configuración y los controles de TI son una responsabilidad compartida entre AWS y usted, nuestro cliente. Para obtener más información, consulte el [modelo de responsabilidad AWS compartida](#). AWS gestiona las tareas de seguridad básicas, como la aplicación de parches al sistema operativo (SO) huésped y a las bases de datos, la configuración del firewall y la recuperación ante desastres. Estos procedimientos han sido revisados y certificados por los terceros pertinentes. Para obtener más información, consulte el recurso siguiente: [Amazon Web Services: información general de procesos de seguridad](#) (documento técnico).

Las siguientes prácticas recomendadas sobre seguridad también evalúan la configuración y los análisis de vulnerabilidades en Amazon GameLift Servers:

- Los clientes son responsables de la administración del software que se implementa en las instancias de Amazon GameLift Servers para el alojamiento de juegos. En concreto:

- El software de las aplicaciones del servidor de juegos que proporciona el cliente requiere un mantenimiento, que incluye actualizaciones y parches de seguridad. Para actualizar el software del servidor de juegos, cargue una nueva compilación en Amazon GameLift Servers o proporciónela en una nueva imagen de contenedor, cree una nueva flota para ella y redirija el tráfico a la nueva flota.
- La imagen de máquina de Amazon (AMI) base, que incluye el sistema operativo, solo se actualiza cuando se crea una nueva flota. Para aplicar parches, actualizar y proteger el sistema operativo y otras aplicaciones que forman parte de la AMI, sustituya las flotas periódicamente, independientemente de las actualizaciones del servidor de juegos. Como mínimo, las flotas de producción deben sustituirse al menos una vez al año. Consulte [Mantener actualizados los entornos de tiempo de ejecución de la flota](#) para obtener más detalles.
- Los clientes deberían considerar la posibilidad de actualizar sus juegos con regularidad con las versiones más recientes del SDK, incluidos el AWS SDK, el SDK para Amazon GameLift Servers servidores para servidores en tiempo real y el SDK para Amazon GameLift Servers clientes.

Prácticas recomendadas de seguridad para Amazon GameLift Servers

Si utiliza Amazon GameLift Servers FleetIQ como característica independiente con Amazon EC2, consulte [Seguridad en Amazon EC2](#) en la Guía del usuario de Amazon EC2.

Amazon GameLift Servers proporciona un número de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no suponen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, plantéese las como consideraciones útiles en lugar de como normas.

Mantener actualizados los entornos de tiempo de ejecución de la flota

Amazon GameLift Servers recomienda que sustituya periódicamente las flotas administradas (incluidas las flotas de EC2 administradas y las flotas de contenedores administradas) para mantener entornos de tiempo de ejecución seguros para los servidores de juegos. Las flotas que se ejecutan durante periodos de tiempo prolongados sin actualizaciones de tiempo de ejecución pueden contener dependencias obsoletas y vulnerabilidades de seguridad que podrían poner en peligro los servidores de juegos. Para obtener más información sobre cómo se comparte la responsabilidad del

software implementado en flotas de Amazon GameLift Servers, consulte [Configuración y análisis de vulnerabilidades en Amazon GameLift Servers](#).

El entorno de tiempo de ejecución de una flota administrada viene determinado por su versión de la imagen de máquina de Amazon (AMI). Cuando se crea una nueva flota, Amazon GameLift Servers le asigna la versión de la AMI más reciente disponible, y todas las instancias de computación de esa flota se implementan con esa versión. Para actualizar la versión de la AMI, debe crear una nueva flota. Para obtener más información sobre las versiones actuales de la AMI, consulte [Versiones de AMI de Amazon GameLift Servers](#).

Prácticas recomendadas:

- Supervise la antigüedad de la flota y sustituya las flotas que tengan más de 30 días de antigüedad: puede monitorizar la fecha de creación de una flota en la consola de Amazon GameLift Servers o utilizar la CLI para recuperar los atributos de la flota. Amazon GameLift Servers muestra en la consola advertencias para las flotas que tienen más de 90 días de antigüedad, y notifica a los titulares de las cuentas por correo electrónico para las flotas que tienen más de un año de antigüedad.

 Note

La actualización de una flota (por ejemplo, usar [UpdateFleetAttributes](#) o [UpdateContainerFleet](#)) no cambia la versión de la AMI. Debe crear una flota nueva.

- Sustituya las flotas periódicamente en función de su estado de seguridad: establezca un calendario regular para crear flotas nuevas y retirar las antiguas. Considere la posibilidad de utilizar un servicio como [Amazon Q](#) para revisar el código del juego con la versión actual de la AMI, detectar problemas de seguridad y sugerir medidas correctivas.
- Pruebe las compilaciones de servidor con las versiones más recientes de la AMI antes de la implementación: es posible que deba modificar la compilación del servidor y cargarla en Amazon GameLift Servers antes de crear una flota nueva.
- Gestione las cuotas de flota para su AWS cuenta: puede solicitar aumentos de los límites si es necesario para crear flotas de reemplazo. Para obtener más información, consulte [Cuotas y puntos de conexión de Amazon GameLift Servers](#).
- Considere la posibilidad de automatizar la sustitución de las flotas: puede automatizar los procesos para crear nuevas flotas y migrar el tráfico de jugadores de las flotas más antiguas. Por ejemplo:

- Úselo AWS CloudFormation para automatizar la creación y la gestión de flotas. Mantenga las configuraciones de su flota como CloudFormation plantillas y utilícelas para lanzar pilas de recursos.
- Aprovecha la función de Amazon GameLift Servers alias para resumir una flota IDs específica. Los alias de flota permiten cambiar fácilmente el tráfico de jugadores de una flota existente a una nueva sin interrumpir las sesiones de juego en curso. Para obtener más información, consulte [Abstracción de una designación de flota de Amazon GameLift Servers con un alias](#).
- Utilice estrategias blue/green de despliegue para reducir el riesgo de migración y evitar el tiempo de inactividad. Con dos entornos de producción idénticos, puede beneficiarse de entorno de prueba similar un entorno de producción completo, ejercer un mayor control del proceso de migración y garantizar las reversiones instantáneas.

Protección de las configuraciones de puerto

Recomendamos encarecidamente no abrir puertos a Internet, ya que eso supone un riesgo para la seguridad. Por ejemplo, la siguiente configuración abre un puerto de escritorio remoto que permite a cualquier usuario de Internet acceder a la instancia:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "0.0.0.0/0",
      "Protocol": "RDP",
      "ToPort": 3389
    }
  ]
}
```

En su lugar, [UpdateFleetPortSettings](#) úselo para abrir un puerto con una dirección IP específica o un rango de direcciones, como se muestra en este ejemplo:

```
{
  "FleetId": "<fleet identifier>",
  "InboundPermissionAuthorizations": [
    {
      "FromPort": 3389,
      "IpRange": "54.186.139.221/32",

```

```
    "Protocol": "TCP",  
    "ToPort": 3389  
  }  
]  
}
```

Recursos de seguridad adicionales

Para obtener más información sobre cómo puede hacer que el uso de Amazon GameLift Servers sea más seguro, consulte el [Pilar de seguridad de AWS Well-Architected Tool](#).

Planificación de precios y costos de Amazon GameLift Servers

Comprender la estructura de precios de Amazon GameLift Servers e implementar estrategias rentables en su solución de alojamiento de juegos es fundamental para optimizar el presupuesto de su infraestructura de juegos. Para obtener información detallada sobre el servicio, incluidos ejemplos de escenarios de precios y herramientas de precios, consulte [Precios de Amazon GameLift Servers](#).

AWS Los nuevos clientes pueden usarla Amazon GameLift Servers sin incurrir en cargos en virtud de la capa gratuita durante un máximo de 12 meses. Para evitar cargos, manténgase dentro de los límites de uso del nivel gratuito.

Modelos de precios

Amazon GameLift Servers ofrece varios modelos de precios diseñados para adaptarse a diferentes escenarios de uso y necesidades empresariales. Cada modelo tiene una base de costo diferente. Puede controlar los costos ajustando los factores de costo de cada modelo.

Alojamiento gestionado

Las flotas de contenedores gestionadas EC2 y gestionadas utilizan EC2 instancias de Amazon que Amazon GameLift Servers gestionan. Cuando configuras el alojamiento de juegos, eliges el tipo de EC2 instancia de Amazon que quieres usar.

- Costes de cómputo: paga por el uso de EC2 instancias por hora para organizar sesiones de juego. Los factores de costo por instancia incluyen:
 - AWS Región
 - Familia, tipo y tamaño de instancia
 - Uso de instancias de spot frente a instancias bajo demanda
 - Sistema operativo (las flotas de contenedores administradas se ejecutan solo en Linux)
- Transferencia de datos: pague por el tráfico entre los clientes de juego y los servidores de juegos alojados.

Alojamiento en cualquier lugar

El alojamiento con Anywhere se ejecuta en recursos de alojamiento de juegos que usted suministra y administra, por lo que no hay costos de computación. En su lugar, los costos se basan en los siguientes factores:

- Sesiones de juego: pague en función del número de sesiones de juego ubicadas en recursos de computación de Anywhere.
- Minutos de conexión: pague por minutos de conexión de los procesos del servidor.

Utilice la Calculadora de precios de Amazon GameLift Servers para hacer una estimación de los costos de las diferentes configuraciones de Amazon GameLift Servers. Utilice la calculadora al diseñar la solución de alojamiento de juegos o para prever los costos de los cambios de configuración. Consulte [Generación de estimaciones de precios de Amazon GameLift Servers](#).

Generación de estimaciones de precios de Amazon GameLift Servers

Con Calculadora de precios de AWS, puedes [crear una estimación de precios para Amazon GameLift Servers](#). No necesita un conocimiento profundo Cuenta de AWS o profundo AWS para usar la calculadora.

Calculadora de precios de AWS La calculadora te guía a través de las decisiones que afectan a los costes del servicio para darte una idea de cuánto Amazon GameLift Servers podría costar tu proyecto de juego. Si aún no está seguro de cómo piensa usar Amazon GameLift Servers, utilice los valores predeterminados para generar una estimación. Si planifica el uso en producción, la calculadora podrá ayudarle a probar posibles escenarios y a generar estimaciones más precisas.

Estimación del alojamiento administrado de Amazon GameLift Servers

Esta opción proporciona una estimación del costo del alojamiento de sus juegos en los servidores administrados de Amazon GameLift Servers, incluidos los costos de uso de la instancia de servidor y de transferencia de datos. Con el alojamiento administrado de Amazon GameLift Servers, no hay un costo adicional para el emparejamiento de FlexMatch.

Si alojas o planeas alojar servidores de juegos en más de una AWS región o en más de un tipo de instancia, crea una estimación para cada región y tipo de instancia.

Instancias de Amazon GameLift Servers

Esta sección te ayuda a estimar el tipo y la cantidad de recursos informáticos que necesitas para alojar las sesiones de juego de tus jugadores. Amazon GameLift Servers utiliza [instancias de Amazon Elastic Compute Cloud \(Amazon EC2\)](#) para administrar los servidores de juegos. En Amazon GameLift Servers, podrá implementar una flota de instancias con un tipo de instancia y un sistema operativo específicos. Si dispone de varias flotas o planea tenerlas, cree una estimación para cada una.

Para empezar, abre la [Amazon GameLift Servers página de configuración](#) de Calculadora de precios de AWS. Añada una Descripción, elija una Región y, a continuación, seleccione Estimación del alojamiento de Amazon GameLift Servers (instancia + transferencia de datos de salida). En Instancias de Amazon GameLift Servers, rellene los siguientes campos:

- Pico de jugadores simultáneos (CCU pico)

Este es el número máximo de jugadores que pueden conectarse a los servidores de juegos al mismo tiempo. Este campo indica la capacidad de alojamiento que necesita Amazon GameLift Servers para satisfacer la demanda máxima de los jugadores. Especifique el número máximo diario de jugadores que espera recibir mediante instancias en la región de AWS elegida.

Por ejemplo, si desea que se conecten 1000 jugadores a su juego a la vez, mantenga el valor predeterminado de **1000**.

- CCU medio por hora como porcentaje del CCU pico diario

Este es el número medio de jugadores simultáneos por hora durante un periodo de 24 horas. Utilizamos ese valor para realizar una estimación de la capacidad de alojamiento continua que necesita mantener Amazon GameLift Servers para sus jugadores. Si no sabe con seguridad el valor porcentual que debe utilizar, mantenga el valor predeterminado del **50** por ciento. Para los juegos con una demanda de jugadores estable, le recomendamos que especifique un valor del **70** por ciento.

Por ejemplo, si su juego tiene un valor de CCU medio por hora de 6000 y un valor CCU pico de 10 000, especifique el valor del **60** por ciento.

- Sesiones de juego por instancia

Es el número de sesiones de juego que cada una de las instancias del servidor de juegos puede alojar simultáneamente. Los factores que pueden afectar a ese número son los requisitos de los recursos del servidor de juegos, la cantidad de jugadores que alojar en cada sesión de juego y las

expectativas de rendimiento de los jugadores. Si para ese juego conoce el número de sesiones simultáneas, especifíquelo. También puede mantener el valor predeterminado de **20**.

- Jugadores por sesión de juego

Es el número medio de jugadores que se conectan a una sesión de juego, tal y como se define en el diseño del juego. Si dispone de modos de juego con un número diferente de jugadores, realice una estimación del número medio de jugadores por sesión de juego en todo el juego. El valor predeterminado es **8**.

- % de búfer de inactividad de instancia

Es el porcentaje de capacidad de alojamiento no utilizada que debe mantenerse como reserva para hacer frente a los picos repentinos de demanda de jugadores. El tamaño del búfer es un porcentaje de la cantidad total de instancias de una flota. El valor predeterminado es del **10** por ciento.

Por ejemplo, con un búfer de inactividad del 20 por ciento, una flota que admita a jugadores con 100 instancias activas mantiene 20 instancias inactivas.

- % de instancias de spot

Las flotas de Amazon GameLift Servers pueden usar una combinación de instancias bajo demanda e instancias de spot. Aunque las instancias bajo demanda ofrecen una disponibilidad más fiable, las instancias de spot ofrecen una alternativa muy rentable. Recomendamos utilizar una combinación de ellas para optimizar tanto el ahorro de costos como la disponibilidad. Para obtener más información sobre cómo Amazon GameLift Servers utiliza las instancias de spot, consulte [Instancias bajo demanda frente a instancias de spot](#).

En este campo, especifique el porcentaje de instancias de spot que mantener en una flota. Recomendamos un porcentaje de instancias de spot de entre el 50 y el 85 por ciento. El valor predeterminado es del **50** por ciento.

Por ejemplo, si implementa una flota con 100 instancias y especifica un **40** %, Amazon GameLift Servers trabajará para mantener 60 instancias bajo demanda y 40 instancias de spot.

- Tipo de instancia

Amazon GameLift Servers las flotas pueden utilizar una variedad de tipos de EC2 instancias de Amazon que varían en cuanto a la potencia de cálculo, la memoria, el almacenamiento y las capacidades de red. Al configurar una Amazon GameLift Servers flota, elige el tipo de instancia que mejor se adapte a las necesidades de tu juego. Para obtener información sobre cómo

seleccionar un tipo de instancia con Amazon GameLift Servers, consulte [Elección de los recursos de computación para una flota administrada](#).

Si conoce el tipo de instancia que está utilizando o que piensa utilizar en su flota de Amazon GameLift Servers, elija ese tipo. Si no sabe con seguridad qué tipo elegir, considere la posibilidad de elegir c5.large. Se trata de un tipo de alta disponibilidad con un tamaño y unas capacidades medios.

- Sistema operativo

En este campo se especifica el sistema operativo en el que se ejecutan los servidores de juegos, ya sea Linux o Windows. El valor predeterminado es Linux.

Transferencia de datos salientes (DTO)

En esta sección se le ayudará a realizar una estimación del costo del tráfico entre clientes de juegos y servidores de juegos. Las tarifas por transferencia de datos se aplican únicamente al tráfico saliente. La transferencia de datos entrantes no tiene ningún costo.

En la [Amazon GameLift Servers](#) [página Configurar](#) de Calculadora de precios de AWS, expanda Transferencia de datos (DTO) y, a continuación, complete los siguientes campos:

- Tipo de estimación de DTO

Puede realizar una estimación de DTO de cualquiera de las dos formas siguientes, en función de cómo realice el seguimiento de la transferencia de datos del juego.

- Por mes (en GB): elija este tipo si realiza un seguimiento del tráfico mensual de sus servidores de juegos.
- Por jugador: elija este tipo si realiza un seguimiento de la transferencia de datos por jugador. Este es el tipo predeterminado.

En el siguiente campo, se realizará una estimación de DTO por jugador en función del número de horas de juego calculado en la sección anterior.

- DTO al mes (en GB)

Si eligió el tipo de estimación de DTO Por mes (en GB), especifique el uso mensual estimado de DTO en GB en cada instancia y por región.

- DTO por jugador

Si ha elegido el tipo de estimación de DTO Por jugador, especifique el uso estimado de DTO por jugador en KB/s. El valor predeterminado es **4**.

Cuando haya terminado de configurar la estimación de precios de Amazon GameLift Servers, elija **Añadir a mi estimación**. Para obtener más información sobre cómo crear y administrar estimaciones en Calculadora de precios de AWS, consulte [Crear una estimación, configurar un servicio y agregar más servicios](#) en la Guía del Calculadora de precios de AWS usuario.

Estrategias de optimización de costos

Utilice estas estrategias con una solución de alojamiento administrada para contribuir a reducir los costos de alojamiento en la nube al tiempo que mantiene un alto rendimiento y una experiencia de juego de calidad.

Prácticas recomendadas

Siga estos consejos para incorporar prácticas de optimización de costos al alojamiento de sus juegos. Como mínimo, le recomendamos que programe revisiones periódicas de sus costos y patrones de uso de Amazon GameLift Servers. Si tiene preguntas relacionadas con los precios, póngase en contacto con su

Configuración de la flota

- Utilice la calculadora de precios: use esta herramienta con los datos de alojamiento del juego para explorar opciones y probar posibles escenarios de configuración para ahorrar costos. Consulte [Calculadora de precios de AWS para Amazon GameLift Servers](#).
- Ahorre en desarrollo: para el desarrollo y las pruebas, utilice tipos de instancias más pequeños y menos costosos, y ejecútelos solo cuando lo necesite.
- Haz coincidir el tipo de instancia con el juego: elige las familias, tipos y tamaños de EC2 instancias de Amazon para tus flotas que mejor se adapten a los requisitos del juego. Consulte la sección Tipos de instancia en [Elección de los recursos de computación para una flota administrada](#).
- Utilice tipos de instancia de Graviton: las instancias de Graviton funcionan con procesadores basados en ARM. Ofrecen una mejor relación precio-rendimiento, mayor eficiencia energética y costos más bajos que instancias comparables basadas en x86 con el modelo de precios bajo demanda.

- Utilice instancias de spot: las instancias de spot suelen costar menos que las instancias bajo demanda, si bien pueden no estar siempre disponibles. Combine flotas de spot y bajo demanda para lograr un equilibrio entre el bajo costo y la alta disponibilidad (se recomienda utilizar un 50-85 % de instancias de spot). Utilice instancias de spot para los modos de juego no críticos o durante las horas de menor actividad. Para supervisar la viabilidad de las instancias de spot, monitorice métricas como `InstanceInterruptions` y `GameServerInterruptions`. Consulte [Instancias bajo demanda frente a instancias de spot](#).
- Ejecute servidores en Linux: compile el tiempo de ejecución de su servidor de juegos para Linux. Las instancias que se implementan con Linux suelen ser más rentables que las que se implementan con Windows.
- Optimice el uso de los recursos: configure las flotas para obtener la máxima eficiencia ejecutando tantos procesos de servidores de juegos simultáneos como sea posible al tiempo que mantiene el rendimiento. Consulte [Optimización de la configuración del tiempo de ejecución del servidor de juegos en Amazon GameLift Servers administrado](#).
- Cree una estrategia multirregional: implementar el alojamiento de juegos en varias ubicaciones puede ofrecer a sus jugadores una latencia más baja y mejores opciones de respaldo. Equilibre la cobertura regional y la rentabilidad implementando primero el alojamiento en los mercados de mayor presencia y utilizando ubicaciones secundarias para aumentar la capacidad durante los picos de actividad. Consulte [Creación una cola con múltiples ubicaciones](#).

Escalado de la capacidad de la flota

- Escalado a cero: cuando las flotas no estén en uso, establezca manualmente la capacidad de la flota a cero para evitar cargos innecesarios. Consulte [Configurar manualmente la capacidad de una Amazon GameLift Servers flota](#).
- Añada escalado automático: evite el aprovisionamiento excesivo de recursos de alojamiento utilizando el escalado automático para ajustar la capacidad del alojamiento de juegos. Adapte la capacidad a las fluctuaciones de la demanda de los jugadores y a otras métricas clave. Consulte [Escalado automático de la capacidad de la flota con Amazon GameLift Servers](#).
- Mantenga un búfer: para gestionar los picos repentinos de demanda de los jugadores sin hacerlos esperar, utilice el seguimiento de objetivos para mantener un búfer de servidores de juegos inactivos. Personalice el búfer en función del tamaño y los patrones de uso de su base de jugadores. Consulte [Escalado automático basado en objetivos](#).

Ubicación de sesiones de juego

- Utilice colas de ubicación: las colas de Amazon GameLift Servers se basan en un algoritmo para determinar las “mejores ubicaciones de alojamiento posibles” para las sesiones de juego en función del costo de los recursos de alojamiento y de otros factores. Consulte [Configuración de la ubicación de las sesiones de juego](#).
- Personalice las prioridades de cola: puede cambiar la forma en que una cola prioriza los costos de alojamiento al ubicar las sesiones de juego. Consulte [Priorización de la ubicación de las sesiones de juego](#).
- Supervise las métricas de ubicación: haga un seguimiento de cómo las colas eligen las ubicaciones de las sesiones de juego para identificar oportunidades de optimización. Consulte [Métricas de Amazon GameLift Servers para colas](#).

Transferencia de datos

- Monitoree las transferencias de datos: supervise cómo se transfieren los datos entre los clientes de juego y los servidores de juegos, y adopte medidas para optimizar la actividad.
- Utilice la compresión de datos: considere la posibilidad de implementar técnicas de compresión de datos para las transferencias de datos entre los clientes y los servidores de juegos. La compresión desempeña un papel esencial a la hora de reducir el uso del ancho de banda y mejorar el rendimiento y la latencia del juego.

Herramientas de uso y costo de los recursos

Descubre cómo utilizar AWS las herramientas para supervisar y optimizar los costes de alojamiento de tus juegos con Amazon GameLift Servers. Para obtener información sobre herramientas adicionales, consulte [Administración de facturación y costos de AWS](#).

AWS herramientas de gestión de costes

- Consola de facturación: revise sus facturas y su uso de AWS . Consulte [Configuración de la facturación](#).
- Alertas de uso del nivel gratuito: configure notificaciones para recibirlas cuando se acerque a los límites del nivel gratuito. Consulte [Seguimiento del uso AWS de la capa gratuita](#).

- Alertas CloudWatch de facturación de Amazon: configura alertas cuando el uso alcance los umbrales personalizados. Consulte [Creación de una alarma de facturación para supervisar los cargos estimados de AWS](#).

Asignación y seguimiento de costos

- Etiquetas de asignación de costos: etiquete sus flotas y otros recursos para organizar y monitorizar los costos de alojamiento. Consulte [Organización y seguimiento de los costes mediante etiquetas de asignación de AWS costes](#).
- Informes de costos: cree informes categorizados por etiquetas asignadas. Consulte [Uso del informe de asignación de costos mensual](#).
- AWS Cost Explorer: analice costos, tendencias y previsiones con filtros y vistas personalizables. Consulte [Analizar los costes y el uso con AWS Cost Explorer](#).
- AWS Budgets: realice un seguimiento y tome medidas sobre los costos y el uso de AWS . Consulte [Administrar sus costos con AWS Budgets](#).

Supervisión del rendimiento con Amazon GameLift Servers Amazon CloudWatch

Supervise estas métricas clave para optimizar el uso de los recursos. Consulta las métricas en la Amazon GameLift Servers consola o usa los CloudWatch paneles de Amazon. Para obtener más información sobre todas las métricas disponibles de Amazon GameLift Servers, consulte [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#).

- Métricas de instancia: `ActiveInstances`, `IdleInstances` y `PercentIdleInstances`
- Métricas de procesos del servidor: `ActiveServerProcesses` y `HealthyServerProcesses`
- Métricas de sesiones de juego: `ActiveGameSessions` y `AvailableGameSessions`
- Métricas de sesiones de jugador: `CurrentPlayerSessions`
- Métricas de cola: `AverageWaitTime` y `QueueDepth`
- Métricas de emparejamiento: `CurrentTickets` y `MatchesPlaced`
- Rendimiento de hardware: `CPUUtilization`, `NetworkIn/NetworkOut` y `DiskReadBytes/DiskWriteBytes`
- Métricas de utilización de recursos
 - Utilice `PercentIdleInstances` para determinar el tamaño óptimo de la flota.

- Monitorice `PercentAvailableGameSessions` para garantizar una capacidad de alojamiento de juegos suficiente.
- Supervise `InstanceInterruptions` y `GameServerInterruptions` para determinar la viabilidad de las instancias de spot.

Guías de referencia de Amazon GameLift Servers

Esta sección contiene la documentación de referencia para usar Amazon GameLift Servers.

Temas

- [API de servicio de Amazon GameLift Servers](#)
- [SDK 5.x del servidor de Amazon GameLift Servers](#)
- [SDK del servidor de las versiones 4 y anteriores de Amazon GameLift Servers](#)
- [Eventos de ubicación de sesión de juego](#)
- [Versiones de AMI de Amazon GameLift Servers](#)
- [Ubicaciones de servicio de Amazon GameLift Servers](#)
- [Cuotas y puntos de conexión de Amazon GameLift Servers](#)
- [Límites de la API de Amazon GameLift Servers](#)
- [Límites de la API del SDK de Amazon GameLift Servers](#)
- [Señalizadores de pings de UDP de Amazon GameLift Servers](#)

API de servicio de Amazon GameLift Servers

Utilice esta lista basada en tareas para buscar las operaciones de API al crear las soluciones de alojamiento de juegos de Amazon GameLift Servers y otras características. El AWS SDK incluye estas operaciones en el espacio de `aws.gamelift` nombres. [Descarga el AWS SDK](#) o [consulta la documentación de referencia de la Amazon GameLift Servers API](#). También puedes usar la API con la interfaz de línea de AWS comandos (AWS CLI), tal y como se describe en la [referencia de AWS CLI comandos](#).

La API incluye dos conjuntos de operaciones para el alojamiento administrado de juegos:

- [Administración de los recursos de alojamiento de Amazon GameLift Servers](#)
- [Inicio de sesiones de juego y unión de los jugadores](#)

La API de servicio de Amazon GameLift Servers también contiene operaciones para su uso con otras herramientas y soluciones de Amazon GameLift Servers. Para obtener una lista FleetIQ APIs, consulta [las operaciones FleetIQ de la API](#). Para ver una lista de las FlexMatch APIs funciones de emparejamiento, consulta [las operaciones de la FlexMatch API](#).

Administración de los recursos de alojamiento de Amazon GameLift Servers

Llame a estas operaciones para configurar los recursos de alojamiento para sus servidores de juegos, escalar la capacidad para satisfacer la demanda de jugadores, obtener métricas de rendimiento y utilización, etc. Utilice estas operaciones de API para alojar servidores de juegos con Amazon GameLift Servers, incluidos Amazon GameLift Servers Realtime. También puede trabajar en la [consola de Amazon GameLift Servers](#) en la mayoría de las tareas de administración de recursos, o puede realizar llamadas con la herramienta AWS Command Line Interface (AWS CLI).

Preparación de servidores de juegos para la implementación

Cargue y configure el código del servidor de juegos del juego para prepararlo para su implementación y lanzamiento en los recursos de alojamiento.

Administración de compilaciones de servidores de juegos personalizados

- [upload-build](#): permite cargar archivos de compilación desde una ruta local y crear un nuevo recurso de compilación de Amazon GameLift Servers. Esta operación, disponible como AWS CLI comando, es la forma más común de cargar versiones de servidores de juegos.
- [CreateBuild](#)— Cree una nueva compilación con los archivos almacenados en un bucket de Amazon S3.
- [ListBuilds](#)— Obtenga una lista de todas las compilaciones subidas a una Amazon GameLift Servers región.
- [DescribeBuild](#)— Recuperar la información asociada a una compilación.
- [UpdateBuild](#)— Cambiar los metadatos de la compilación, incluidos el nombre y la versión de la compilación.
- [DeleteBuild](#)— Eliminar una compilación de Amazon GameLift Servers.

Administración de scripts de configuración de Amazon GameLift Servers Realtime

- [CreateScript](#)— Cargue JavaScript archivos y cree un nuevo recurso de Amazon GameLift Servers script.
- [ListScripts](#)— Obtenga una lista de todos los Realtime scripts subidos a una Amazon GameLift Servers región.
- [DescribeScript](#)— Recuperar la información asociada a un Realtime script.

- [UpdateScript](#)— Cambiar los metadatos del guion y cargar el contenido del guion revisado.
- [DeleteScript](#)— Eliminar un Realtime guion de Amazon GameLift Servers.

Configuración de los recursos informáticos para el alojamiento

Configure los recursos de alojamiento e impleméntelos con la compilación del servidor de juegos o el script de configuración de Realtime.

Creación y administración de flotas

- [CreateFleet](#)— Configura e implementa una nueva Amazon GameLift Servers flota de recursos informáticos para ejecutar tus servidores de juegos. Una vez implementados, los servidores de juegos se lanzan automáticamente según estén configurados y preparados para alojar sesiones de juego.
- [ListFleets](#)— Obtenga una lista de todas las flotas de una Amazon GameLift Servers región.
- [DeleteFleet](#)— Elimina una flota que ya no utilice servidores de juegos ni aloje a jugadores.
- Visualización/actualización de las ubicaciones de la flota
 - [CreateFleetLocations](#)— Agregue ubicaciones remotas a una flota existente que admita múltiples ubicaciones
 - [DescribeFleetLocationAttributes](#)— Obtenga una lista de todas las ubicaciones remotas de una flota y consulte el estado actual de cada ubicación.
 - [DeleteFleetLocations](#)— Elimine las ubicaciones remotas de una flota que admite múltiples ubicaciones.
- Consulte o actualice las configuraciones de la flota.
 - [DescribeFleetAttributes/UpdateFleetAttributes](#)— Consulta o cambia los metadatos y la configuración de una flota para proteger las sesiones de juego y limitar la creación de recursos.
 - [DescribeFleetPortSettings/UpdateFleetPortSettings](#)— Consulta o cambia los permisos de entrada (rangos de configuración de direcciones IP y puertos) permitidos para una flota.
 - [DescribeRuntimeConfiguration/UpdateRuntimeConfiguration](#)— Vea o cambie qué procesos del servidor (y cuántos) desea ejecutar en cada instancia de una flota.

Administración de la capacidad de la flota

- [Describe EC2 InstanceLimits](#): recupere el número máximo de instancias permitido para la AWS cuenta corriente y el nivel de uso actual.

- [DescribeFleetCapacity](#)— Recupera la configuración de capacidad actual de la región de origen de una flota.
- [DescribeFleetLocationCapacity](#)— Recupera la configuración de capacidad actual para cada ubicación de una flota con varias ubicaciones.
- [UpdateFleetCapacity](#)— Ajuste manualmente la configuración de capacidad de una flota.
- Configure :
 - [PutScalingPolicy](#)— Active el escalado automático basado en objetivos, cree una política de autoescalado personalizada o actualice una política existente.
 - [DescribeScalingPolicies](#)— Recuperar una política de escalado automático existente.
 - [DeleteScalingPolicy](#)— Elimine una política de escalado automático y evite que afecte a la capacidad de la flota.
 - [StartFleetActions](#)— Reiniciar las políticas de escalado automático de una flota.
 - [StopFleetActions](#)— Suspender las políticas de escalado de automóviles de una flota.

Monitoree la actividad de la flota.

- [DescribeFleetUtilization](#)— Recupera estadísticas sobre el número de procesos del servidor, las sesiones de juego y los jugadores que están activos actualmente en una flota.
- [DescribeFleetLocationUtilization](#)— Recupera las estadísticas de uso de cada ubicación de una flota con varias ubicaciones.
- [DescribeFleetEvents](#)— Ver los eventos registrados de una flota durante un período de tiempo específico.
- [DescribeGameSessions](#)— Recupera los metadatos de las sesiones de juego, incluidos el tiempo de ejecución de un juego y el número actual de jugadores.

Configuración de colas para la ubicación de sesiones de juego

Configure colas de varias flotas y regiones para colocar las sesiones de juego con los mejores recursos de alojamiento disponibles en cuanto a costo, latencia y resiliencia.

- [CreateGameSessionQueue](#)— Crea una cola para utilizarla al procesar las solicitudes de ubicación de las sesiones de juego.
- [DescribeGameSessionQueues](#)— Recupera las colas de sesiones de juego definidas en una región. Amazon GameLift Servers

- [UpdateGameSessionQueue](#)— Cambiar la configuración de una cola de sesiones de juego.
- [DeleteGameSessionQueue](#)— Eliminar una cola de sesiones de juego de la región.

Administrar alias

Utilice alias para representar sus flotas o crear un destino alternativo de terminal. Los alias son útiles cuando se pasa la actividad del juego de una flota a otra, como durante las actualizaciones de compilación del servidor de juegos.

- [CreateAlias](#)— Defina un nuevo alias y, si lo desea, asígnelo a una flota.
- [ListAliases](#)— Obtenga todos los alias de flota definidos en una Amazon GameLift Servers región.
- [DescribeAlias](#)— Recuperar información sobre un alias existente.
- [UpdateAlias](#)— Cambiar la configuración de un alias, por ejemplo, redirigirlo de una flota a otra.
- [DeleteAlias](#)— Eliminar un alias de la región.
- [ResolveAlias](#)— Obtiene el identificador de flota al que apunta un alias específico.

Conexión a instancias de alojamiento administrado

Consulte información sobre las distintas instancias de una flota o solicite el acceso remoto a la instancia de una flota especificada para solucionar problemas.

- [DescribeInstances](#)— Obtenga información sobre cada instancia de una flota, incluidos el ID de la instancia, la dirección IP, la ubicación y el estado.
- [GetInstanceAccess](#)— Solicite las credenciales de acceso necesarias para conectarse de forma remota a una instancia específica de una flota.

Configurar las interconexiones de VPC

Cree y administre conexiones de emparejamiento de VPC entre sus recursos de Amazon GameLift Servers alojamiento y otros recursos. AWS

- [CreateVpcPeeringAuthorization](#)— Autoriza una conexión de emparejamiento a uno de tus VPCs
- [DescribeVpcPeeringAuthorizations](#)— Recupera las autorizaciones de conexión entre pares válidas.
- [DeleteVpcPeeringAuthorization](#)— Eliminar una autorización de conexión entre pares.

- [CreateVpcPeeringConnection](#)— Establezca una conexión de emparejamiento entre la VPC de Amazon GameLift Servers una flota y una de las suyas. VPCs
- [DescribeVpcPeeringConnections](#)— Recupere información sobre las conexiones de emparejamiento de VPC activas o pendientes con una flota. Amazon GameLift Servers
- [DeleteVpcPeeringConnection](#)— Eliminar una conexión de emparejamiento de VPC con una flota. Amazon GameLift Servers

Inicio de sesiones de juego y unión de los jugadores

Llame a estas operaciones desde un servicio de backend para iniciar nuevas sesiones de juego, obtener información sobre las sesiones de juego existentes y unir a los jugadores a las sesiones de juego. Estas operaciones son para su uso con servidores de juegos personalizados alojados en Amazon GameLift Servers. Si está utilizando Amazon GameLift Servers Realtime, administre las sesiones de juego mediante la API de cliente de [Amazon GameLift Servers Realtime \(C#\)](#).

- Inicie nuevas sesiones de juego para uno o varios jugadores.
 - [StartGameSessionPlacement](#)— Pide Amazon GameLift Servers encontrar los mejores recursos de alojamiento disponibles e inicia una nueva sesión de juego. Este es el método preferido para crear nuevas sesiones de juego. Se basa en las colas de sesiones de juego para realizar un seguimiento de la disponibilidad del alojamiento en varias regiones y utilizar los algoritmos de FleetIQ para priorizar las ubicaciones en función de la latencia de los jugadores, el costo del alojamiento, la ubicación, etc.
 - [DescribeGameSessionPlacement](#)— Obtenga los detalles y el estado de una solicitud de colocación.
 - [StopGameSessionPlacement](#)— Cancelar una solicitud de colocación.
 - [CreateGameSession](#)— Inicia una nueva sesión de juego vacía en una ubicación específica de la flota. Esta operación le proporciona un mayor control sobre dónde iniciar la sesión de juego en lugar de utilizar FleetIQ para evaluar las opciones de ubicación. Debe añadir jugadores a la nueva sesión de juego en un paso aparte.
- Coloque a los jugadores en sesiones de juego existentes. Busque sesiones de juego en ejecución con ranuras de jugador disponibles y resérvelas para nuevos jugadores.
 - [CreatePlayerSession](#)— Reserva un espacio libre para que un jugador se una a una sesión de juego.
 - [CreatePlayerSessions](#)— Reserva espacios abiertos para que varios jugadores se unan a una sesión de juego.

- Trabaje con datos de sesiones de juego y de jugador. Administración de información sobre sesiones de juego y sesiones de jugador
 - [SearchGameSessions](#)— Solicita una lista de las sesiones de juego activas en función de un conjunto de criterios de búsqueda.
 - [DescribeGameSessions](#)— Recupera los metadatos de sesiones de juego específicas, incluido el tiempo de actividad y el número actual de jugadores.
 - [DescribeGameSessionDetails](#)— Recupera los metadatos, incluida la configuración de protección de la sesión de juego, de una o más sesiones de juego.
 - [DescribePlayerSessions](#)— Obtén detalles sobre la actividad de los jugadores, incluidos el estado, el tiempo de juego y los datos de los jugadores.
 - [UpdateGameSession](#)— Cambia la configuración de la sesión de juego, como el número máximo de jugadores y la política de registro.
 - [GetGameSessionLogUrl](#)— Obtener la ubicación de los registros guardados de una sesión de juego.

SDK 5.x del servidor de Amazon GameLift Servers

Esta sección contiene la documentación de referencia del SDK 5.x del servidor de Amazon GameLift Servers. El SDK del servidor proporciona la funcionalidad principal que el servidor de juegos utiliza para interactuar con el servicio de Amazon GameLift Servers. Por ejemplo, el servidor de juegos recibe peticiones del servicio para iniciar y detener las sesiones de juego, y le envía al servicio actualizaciones periódicas del estado de las sesiones de juego. Integre los servidores de juegos con el SDK del servidor antes de implementarlos para el alojamiento.

Utilice esta referencia del SDK del servidor con el fin de integrar servidores de juegos multijugador personalizados para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

La versión principal más reciente del SDK del servidor de Amazon GameLift Servers es la 5.x. Las siguientes características de alojamiento requieren el uso de la versión 5.x:

- Amazon GameLift Servers Anywhere
- Complemento de Amazon GameLift Servers para Unreal Engine y Unity

 Note

Si necesita utilizar la versión 4.x, o una versión anterior, del SDK del servidor, consulte [SDK del servidor de las versiones 4 y anteriores de Amazon GameLift Servers](#) para obtener documentación y la información de descarga.

Temas

- [Actualizaciones del SDK 5 del servidor de Amazon GameLift Servers](#)
- [Migración al SDK 5.x del servidor de Amazon GameLift Servers](#)
- [SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#)
- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)
- [SDK del servidor de Go para Amazon GameLift Servers: acciones](#)
- [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)

Actualizaciones del SDK 5 del servidor de Amazon GameLift Servers

Los servidores de juego alojados utilizan el SDK del servidor de Amazon GameLift Servers para comunicarse con el servicio de Amazon GameLift Servers a fin de iniciar y administrar las sesiones de juego para los jugadores. La última versión, el SDK del servidor 5 de Amazon GameLift Servers, ofrece mejoras y compatibilidad con las nuevas características de Amazon GameLift Servers. Si su compilación de servidor de juegos utiliza actualmente el SDK 4 del servidor de Amazon GameLift Servers (o una versión anterior), siga las instrucciones de este tema para actualizar los juegos.

La versión 5.0.0, y versiones posteriores, del SDK del servidor de Amazon GameLift Servers incluyen las siguientes actualizaciones:

- Más lenguajes: las bibliotecas están disponibles en C++, C# y Go; puede compilar las bibliotecas en C++ para usarlas con Unreal Engine.
- Compatibilidad con complementos de motor de juegos: los complementos independientes de Amazon GameLift Servers para Unreal Engine y Unity requieren las bibliotecas del SDK 5 del servidor de Amazon GameLift Servers. Estos complementos ofrecen flujos de trabajo guiados para integrar, probar e implementar juegos en Amazon GameLift Servers para el alojamiento. Consulte [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#) y la documentación [Complemento de Amazon GameLift Servers para Unreal Engine](#).

- **Compatibilidad con Amazon GameLift Servers Anywhere:** con flotas de Anywhere, puede configurar sus propios recursos de alojamiento para utilizar las características de Amazon GameLift Servers (incluido el emparejamiento). Añada el agente de Amazon GameLift Servers para automatizar la administración del ciclo de vida de las sesiones de juego. Utilice flotas de Anywhere para el alojamiento de producción con hardware en las instalaciones, o configure entornos de prueba para desarrollar juegos de un modo rápido e iterativo. Consulte [Alojamiento con Anywhere](#) y el [agente de Amazon GameLift Servers](#).
- **Herramientas de prueba actualizadas:** la característica de Amazon GameLift Servers Anywhere le permite configurar entornos de prueba locales o basados en la nube para los juegos. Configure las pruebas con o sin el agente de Amazon GameLift Servers. Estas herramientas sustituyen a la versión local de Amazon GameLift Servers. Consulte [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#).
- **Solución .NET consolidada para C#:** el SDK 5.1+ del servidor para C# es compatible con .NET Framework 4.6.2 (actualizado desde la versión 4.6.1) y .NET 6.0 en una sola solución. .NET Standard 2.1 está disponible con las bibliotecas creadas por Unity.
- **Nuevo recurso Compute:** este nuevo recurso combina diferentes tipos de recursos de alojamiento. Incluye recursos de alojamiento basados en la nube (flotas de contenedores o de EC2 administradas) y recursos de alojamiento controlados por el cliente (flotas de Anywhere). Incluye las siguientes actualizaciones:
 - Algunas de las nuevas llamadas a la API para el recurso Compute son [ListCompute\(\)](#), [DescribeCompute\(\)](#) y [GetComputeAccess\(\)](#). Estas acciones devuelven información sobre los recursos de alojamiento para cualquier tipo de flota de Amazon GameLift Servers. En términos generales, para las flotas con servidores de juegos que utilicen el SDK 5.x del servidor, utilice las acciones específicas del recurso de computación para reemplazar a las acciones específicas de la instancia. Además, las siguientes acciones se deben usar en flotas de Amazon GameLift Servers sin el agente de : [RegisterCompute\(\)](#), [DeregisterCompute\(\)](#) y [GetComputeAuthToken\(\)](#).
 - Nueva métrica `ActiveCompute` con dimensiones de `CloudWatch FleetId`, `Location` y `ComputeType`. Esta métrica reemplaza a la métrica anterior `ActiveInstances`.
- **Amazon EC2 Systems Manager (SSM) para acceso remoto:** para mayor seguridad, utilice SSM en lugar de SSH cuando se conecte a instancias en flotas administradas por Amazon GameLift Servers. Consulte [Conexión remota a instancias de flota de Amazon GameLift Servers](#).

Migración al SDK 5.x del servidor de Amazon GameLift Servers

Para actualizar un proyecto de juego para que utilice la versión 5.x del SDK del servidor, realice los siguientes cambios:

Note

Si necesita seguir usando la versión 4.x, o una versión anterior, del SDK del servidor, consulte [SDK del servidor de las versiones 4 y anteriores de Amazon GameLift Servers](#) para obtener documentación y la información de descarga. En concreto, para la versión 4.0.2 del SDK, puedes descargarla desde las [GitHub versiones oficiales](#).

1. Obtenga el paquete más reciente del SDK del servidor de Amazon GameLift Servers para su entorno de desarrollo [[sitio de descarga](#)]. Siga las instrucciones de instalación del archivo Readme correspondiente al paquete y la versión descargados. Consulta estas instrucciones para usar el servidor SDKs con tu proyecto de juego.
 - [Para entornos de desarrollo que utilizan C++, C# o Go](#)
 - [Para proyectos de Unreal Engine \(SDK del servidor de C++ solo para bibliotecas de Unreal\)](#)
 - [Para proyectos de Unity \(SDK del servidor de C# solo para bibliotecas de Unity\)](#)
 - [Para su uso con el complemento de Amazon GameLift Servers para Unreal Engine](#)
 - [Para su uso con el complemento de Amazon GameLift Servers para Unity](#)
2. Actualice el código del servidor de la siguiente manera:
 - Cambie la función de devolución de llamada del código del servidor, de `onCreateGameSession()` a `onStartGameSession()`.
 - Actualice las entradas `InitSDK()` como corresponda:
 - Si planea implementar la compilación del servidor de juegos en una flota de EC2 administrada por Amazon GameLift Servers o en una flota de Anywhere con el agente de Amazon GameLift Servers:

Llame a `InitSDK()` sin parámetros ([C++](#)) ([C#](#)) ([Unreal](#)). Esta llamada configura el entorno informático y una WebSocket conexión al Amazon GameLift Servers servicio.
 - Si planea implementar la compilación del servidor de juegos en una flota de Anywhere sin el agente de Amazon GameLift Servers:

Llame a `InitSDK()` con parámetros de servidor ([C++](#)) ([C#](#)) ([Unreal](#)). Un proceso del servidor de juegos utiliza estos parámetros para establecer una conexión con el servicio de Amazon GameLift Servers.

3. Si la compilación del servidor de juegos, u otras aplicaciones alojadas, se comunican con otros recursos de AWS mientras se están ejecutando, tendrá que cambiar cómo accede la aplicación a esos recursos. Sustituya el uso de `AssumeRoleCredentials` por la nueva acción `GetFleetRoleCredentials()` del SDK del servidor (para servidores de juegos) o utilice credenciales compartidas (para otras aplicaciones). Para obtener más información sobre cómo implementar este cambio, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).
4. Si el proyecto ha llamado a la acción `GetInstanceCertificate()` del SDK del servidor a fin de recuperar un certificado TLS, modifique el código para usar el nuevo `GetComputeCertificate()` ([C++](#)) ([C#](#)) ([Unreal](#)) en su lugar.
5. Cuando subas la versión del juego a Amazon GameLift Servers (por ejemplo, con [upload-build](#) o [CreateBuild\(\)](#)), establece el `ServerSdkVersion` parámetro en la versión 5.x que estés utilizando (este parámetro actualmente tiene el valor predeterminado 4.0.2). Este parámetro debe coincidir con las bibliotecas reales del SDK del servidor en la compilación del servidor del juego. Si especifica una versión incorrecta para una compilación de servidor de juegos cargada, todas las flotas que cree con esa compilación generarán errores. Consulte [Creación de una compilación de servidor de juegos para Amazon GameLift Servers](#).

En el siguiente ejemplo, se muestra cómo especificar la versión del SDK del servidor:

```
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 utiliza scripts para conectarse remotamente a flotas administradas, actualice los scripts para utilizar el nuevo proceso, tal y como se describe en [Conexión remota a instancias de flota de Amazon GameLift Servers](#).

SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones

Utilice la referencia del SDK 5.x del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

En este tema se describe la API de C++ de Amazon GameLift Servers que puede utilizar al realizar la compilación con la biblioteca estándar de C++ (std). En concreto, esta documentación se aplica al código que se compila con la opción -DDGAMELIFT_USE_STD=1.

SDK 5.x del servidor de C++ para Amazon GameLift Servers: tipos de datos

Utilice la referencia del SDK 5.x del servidor de C++ de Amazon GameLift Servers para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

En este tema se describe la API de C++ de Amazon GameLift Servers que puede utilizar al realizar la compilación con la biblioteca estándar de C++ (std). En concreto, esta documentación se aplica al código que se compila con la opción -DDGAMELIFT_USE_STD=1.

[SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#)

Tipos de datos

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)

- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Jugador](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [AttributeValue](#)
- [GetFleetRoleCredentialsRequest](#)
- [AwsLongOutcome](#)
- [AwsStringOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [GenericOutcome](#)
- [GenericOutcomeCallable](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [Iniciar SDKOutcome](#)
- [GameLiftError](#)
- [Enums](#)

LogParameters

Un objeto que identifica los archivos generados durante una sesión de juego que quiere que Amazon GameLift Servers cargue y almacene una vez finalizada la sesión de juego. El servidor de juegos proporciona LogParameters a Amazon GameLift Servers como parte de un objeto ProcessParameters en una llamada a [ProcessReady\(\)](#).

Propiedades	Descripción
LogPaths	Lista de las rutas de directorio a archivos de registro del servidor de juegos que desea que Amazon GameLift Servers almacene para futuros accesos. El proceso del servidor genera esos archivos durante una sesión de juego. Defina los nombres y las rutas de los archivos en el servidor de juegos y almacénelos en el directorio raíz de compilación del juego. Las rutas del registro deben ser absolutas. Por ejemplo, si la compilación del juego almacena los registros de sesión de juego en una ruta del tipo <code>MyGame\sessionLogs\</code>, la ruta sería <code>c:\game\MyGame\sessionLogs</code> en una instancia de Windows. Tipo: <code>std::vector<std::string></code> Obligatorio: no

MetricsParameters

Utilice este tipo de datos para configurar la recopilación de métricas y los informes de bloqueo del servidor del juego. El servidor de juegos comunica `MetricsParameters` a Amazon GameLift Servers en una llamada a [InitMetrics\(\)](#).

Propiedades	Descripción
StatsdHost	Nombre de host o dirección IP del servidor de StatsD para la recopilación de métricas. Tipo: <code>std::string</code> Obligatorio: no

StatsdPort	Número de puerto del servidor de StatsD para la recopilación de métricas. Tipo: <code>int</code> Obligatorio: no
CrashReporterHost	Nombre de host o dirección IP del servidor que notifica el bloqueo. Tipo: <code>std::string</code> Obligatorio: no
CrashReporterPort	Número de puerto del servidor que notifica el bloqueo. Tipo: <code>int</code> Obligatorio: no
FlushIntervalMs	Intervalo en milisegundos para vaciar los datos de las métricas al servidor. Tipo: <code>int</code> Obligatorio: no
MaxPacketSize	Tamaño máximo en bytes de los paquetes de métricas enviados al servidor. Tipo: <code>int</code> Obligatorio: no

ProcessParameters

Este tipo de datos contiene el conjunto de parámetros enviado a Amazon GameLift Servers en una [ProcessReady\(\)](#).

Propiedades	Descripción
LogParameters	Un objeto con rutas de directorio a los archivos que se generan durante una sesión de juego. Amazon GameLift Servers copia y almacena los archivos para accesos futuros. Tipo: <code>Aws::GameLift::Server:: LogParameters</code> Obligatorio: no
OnHealthCheck	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos y espera otros 60 segundos para obtener una respuesta. El proceso del servidor devuelve TRUE si está en buen estado y FALSE si no. Si no se devuelve ninguna respuesta, Amazon GameLift Servers registra el proceso del servidor como defectuoso. Tipo: <code>std::function<bool()> onHealthCheck</code> Obligatorio: no
OnProcessTerminate	La función de devolución de llamada que invoca Amazon GameLift Servers para forzar el cierre del proceso de servidor. Después de llamar a esta función, Amazon GameLift Servers espera 5 minutos hasta que el proceso del servidor se cierre y responde con una llamada a ProcessEnding() antes de cerrar el proceso del servidor.

	Tipo: <code>std::function<void()></code> <code>onProcessTerminate</code> Obligatorio: sí
OnStartGameSession	La función de devolución de llamada que se Amazon GameLift Servers invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a una solicitud de un cliente. CreateGameSession La función de devolución de llamada pasa un objeto GameSession. Tipo: <code>const std::function<void (Aws::GameLift::Model::GameSession)></code> <code>onStartGameSession</code> Obligatorio: sí
OnUpdateGameSession	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de reposición de emparejamiento para proporcionar datos actualizados del emparejador. Pasa un objeto GameSession, una actualización de estado (<code>updateReason</code>) y el ID del ticket de reposición de emparejamiento. Tipo: <code>std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)></code> <code>onUpdateGameSession</code> Obligatorio: no

Puerto

El número de puerto en el que escucha el proceso del servidor para recibir conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: Integer

Obligatorio: sí

UpdateGameSession

Este tipo de datos se actualiza a un objeto de sesión de juego, que incluye el motivo por el que se actualizó la sesión de juego y el ID del ticket de reposición correspondiente si la reposición se utiliza para reponer las sesiones de los jugadores en la sesión de juego.

Propiedades	Descripción
GameSession	Un objeto GameSession. El objeto <code>GameSession</code> contiene propiedades que describen una sesión de juego. Tipo: <code>Aws::GameLift::Server::GameSession</code> Obligatorio: sí
UpdateReason	El motivo por el que se actualiza la sesión de juego. Tipo: <code>Aws::GameLift::Server::UpdateReason</code> Obligatorio: sí

Propiedades	Descripción
BackfillTicketId	El ID de ticket de reposición que intenta actualizar la sesión de juego. Tipo: <code>std::string</code> Obligatorio: no

GameSession

Este tipo de datos proporciona detalles de una sesión de juego.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juego. El ARN de una sesión de juego tiene el siguiente formato: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Tipo: <code>std::string</code> Obligatorio: no
Name	Una etiqueta descriptiva de la sesión de juego. Tipo: <code>std::string</code> Obligatorio: no
FleetId	Un identificador único para la flota en la que se ejecuta la sesión de juego. Tipo: <code>std::string</code> Obligatorio: no

Propiedades	Descripción
MaximumPlayerSessionCount	El número máximo de conexiones de jugadores a la sesión de juego. Tipo: <code>int</code> Obligatorio: no
Puerto	El número de puerto de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: <code>int</code> Obligatorio: no
IpAddress	La dirección IP del servidor de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: <code>std::string</code> Obligatorio: no
GameSessionData	Un conjunto de propiedades de sesión de juego personalizadas, formateadas como un valor de una sola cadena. Tipo: <code>std::string</code> Obligatorio: no

Propiedades	Descripción
MatchmakerData	Información sobre el proceso de emparejamiento que se utilizó para crear la sesión de juego, en sintaxis JSON, con formato como cadena. Además de la configuración de emparejamiento utilizada, contiene datos sobre todos los jugadores asignados al emparejamiento, incluidos los atributos de los jugadores y las asignaciones de los equipos. Tipo: <code>std::string</code> Obligatorio: no
GameProperties	Un conjunto de propiedades personalizadas de una sesión de juego, con formato como pares clave-valor. Estas propiedades se pasan a una solicitud de iniciar una nueva sesión de juego. Tipo: <code>std::vector<GameProperty></code> Obligatorio: no

Propiedades	Descripción
DnsName	El identificador de DNS asignado a la instancia que ejecuta la sesión de juego. Los valores tienen formato siguiente: • Flotas habilitadas para TLS: <unique identifier>.<region identifier>.amazongamelift.com . • Non-TLS-enabled flotas: ec2-<unique identifier>.compute.amazonaws.com . Cuando se conecte a una sesión de juego que se ejecute en una flota habilitada de TLS, debe utilizar el nombre de DNS, no la dirección IP. Tipo: <code>std::string</code> Obligatorio: no

ServerParameters

Información utilizada por el proceso del servidor de juegos para establecer una conexión con el servicio de Amazon GameLift Servers. Incluya estos parámetros al llamar a [InitSDK\(\)](#) solo si la compilación del servidor de juegos se va a implementar en una flota de Anywhere o en una flota de contenedores sin el agente de Amazon GameLift Servers. Para todos los demás escenarios de implementación, llame a [InitSDK\(\)](#) sin parámetros.

Propiedades	Descripción
websocketUrl	El <code>GameLiftServerSdkEndpoint</code> que devuelve Amazon GameLift Servers cuando se invoca RegisterCompute para un recurso de computación de Amazon GameLift Servers Anywhere.

Propiedades	Descripción
	Tipo: <code>std::string</code> Obligatorio: sí
<code>processId</code>	Un identificador único registrado en el proceso de servidor que aloja el juego. Tipo: <code>std::string</code> Obligatorio: sí
<code>hostId</code>	El <code>HostID</code> es el <code>ComputeName</code> utilizado cuando registró el recurso informático. Para obtener más información, consulte, RegisterCompute . Tipo: <code>std::string</code> Obligatorio: sí
<code>fleetId</code>	El identificador único de la flota en la que está registrado el recurso informático. Para obtener más información, consulte, RegisterCompute . Tipo: <code>std::string</code> Obligatorio: sí
<code>authToken</code>	El token de autenticación generado por Amazon GameLift Servers que autentica el servidor en Amazon GameLift Servers. Para obtener más información, consulte, GetComputeAuthToken . Tipo: <code>std::string</code> Obligatorio: sí

StartMatchBackfillRequest

Información utilizada para crear una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información a Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	Un identificador único de la sesión de juegos. El GetGameSessionId de la operación de la API devuelve el identificador en formato de ARN. Tipo: <code>std::string</code> Obligatorio: sí
MatchmakingConfigurationArn	Un identificador único, con formato de ARN, que el emparejador utiliza para esta solicitud. El ARN del emparejador para la sesión de juego original se encuentra en el objeto de sesión de juego en la propiedad de datos del emparejador. Puede obtener más información sobre los datos del emparejador en Trabajo con datos del emparejador. Tipo: <code>std::string</code> Obligatorio: sí
Players	Un conjunto de datos que representa a todos los jugadores que están en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Tipo: <code>std::vector<Player></code> Obligatorio: sí

Propiedades	Descripción
TicketId	Un identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no proporciona un valor, Amazon GameLift Servers generará uno. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario. Tipo: <code>std::string</code> Obligatorio: no

Jugador

Este tipo de datos representa a un jugador en el sistema de emparejamiento. Al iniciar una solicitud de emparejamiento, un jugador tiene un ID de jugador, atributos y, posiblemente, datos de latencia. Amazon GameLift Servers añade la información de los equipos tras realizar un emparejamiento.

Propiedades	Descripción
LatencyInMS	Un conjunto de valores expresados en milisegundos que indican la cantidad de latencia que experimenta un jugador cuando se conecta a una ubicación. Si se utiliza esta propiedad, el jugador solo se empareja con las ubicaciones que aparecen. Si un creador de emparejamientos tiene una regla que evalúa la latencia de los jugadores, estos deben informar de la latencia para ser emparejados. Tipo: <code>Dictionary<string,int></code> Obligatorio: no

Propiedades	Descripción
PlayerAttributes	Una colección de pares de clave-valor que contienen información del jugador para su uso en el emparejamiento. Las claves de atributos del jugador deben coincidir con las PlayerAttributes utilizadas en un conjunto de reglas de emparejamiento. Para obtener más información sobre los atributos de los jugadores, consulte Attribute Value. Tipo: <code>std::map<std::string, AttributeValue></code> Obligatorio: no
PlayerId	Un identificador único de un jugador. Tipo: <code>std::string</code> Obligatorio: no
Equipo	El nombre del equipo al que está asignado el jugador en un emparejamiento. Defina el nombre del equipo se define en el conjunto de reglas de emparejamiento. Tipo: <code>std::string</code> Obligatorio: no

DescribePlayerSessionsRequest

Un objeto que especifica las sesiones de jugador que recuperar. El proceso del servidor proporciona esta información con una llamada [DescribePlayerSessions\(\)](#) a Amazon GameLift Servers.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juegos. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de sesión de juego es <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code>. El <code>GameSessionID</code> es una cadena de ID personalizada o una Tipo: <code>std::string</code> Obligatorio: no
PlayerSessionId	El identificador único de una sesión de jugador. Utilice este parámetro para solicitar una sesión de jugador específica. Tipo: <code>std::string</code> Obligatorio: no
PlayerId	El identificador único de un jugador. Utilice este parámetro para solicitar todas las sesiones de jugador para un jugador específico. Consulte Generación de ID de jugador. Tipo: <code>std::string</code> Obligatorio: no
PlayerSessionStatusFilter	El estado de la sesión de jugador para filtrar los resultados. Los posibles estados de sesión de jugador son los siguientes: • RESERVADO: se ha recibido la solicitud de sesión de jugador, pero el jugador aún no se

Propiedades	Descripción
	ha conectado al proceso del servidor o aún no se ha validado. • ACTIVO: el proceso del servidor ha validado el jugador y está conectado. • COMPLETO: la conexión del jugador se ha caído. • TIEMPO DE ESPERA AGOTADO: se ha recibido una solicitud de sesión de jugador, pero el jugador no se ha conectado y/o no se ha validado en el plazo de tiempo de espera (60 segundos). Tipo: <code>std::string</code> Obligatorio: no
NextToken	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: <code>std::string</code> Obligatorio: no
Límite	El número máximo de resultados que devolver. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: <code>int</code> Obligatorio: no

StopMatchBackfillRequest

Información utilizada para cancelar una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	Un identificador único de sesión de juego de la solicitud que se va a cancelar. Tipo: <code>char[]</code> Obligatorio: no
MatchmakingConfigurationArn	Un identificador único del emparejador al que se envió esta solicitud. Tipo: <code>char[]</code> Obligatorio: no
TicketId	Un identificador único del ticket de solicitud de reposición que se va a cancelar. Tipo: <code>char[]</code> Obligatorio: no

AttributeValue

Utilice estos valores en pares de clave-valor de atributo [Jugador](#). Este objeto le permite especificar un valor de atributo mediante cualquiera de los tipos de datos válidos: cadena, número, matriz de cadenas o mapa de datos. Cada objeto `AttributeValue` debe utilizar exactamente una de las propiedades disponibles: S, N, SL o SDM.

Propiedades	Description (Descripción)
AttrType	Especifica el tipo de valor del atributo. Entre los posibles tipos de valores de atributo se incluyen:

Propiedades	Description (Descripción)
	• NONE • STRING • DOUBLE • STRING_LIST • STRING_DOUBLE_MAP Obligatorio: no
S	Representa un valor de atributo de cadena. Tipo: <code>std::string</code> Obligatorio: no
N	Representa un valor de atributo numérico. Tipo: <code>double</code> Obligatorio: no
SL	Representa una matriz de valores de atributos de cadena. Tipo: <code>std::vector<std::string></code> Obligatorio: no
SDM	Representa un diccionario de claves de cadena y valores dobles. Tipo: <code>std::map<std::string, double></code> Obligatorio: no

GetFleetRoleCredentialsRequest

Este tipo de datos proporciona al servidor del juego un acceso limitado al resto de tus AWS recursos. Para obtener más información, consulte, [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Propiedades	Descripción
RoleArn	El nombre de recurso de Amazon (ARN) de la función de servicio que amplía el acceso limitado a sus AWS recursos. Tipo: <code>std::string</code> Obligatorio: no
RoleSessionName	El nombre de sesión del rol que puede usar para identificar de forma exclusiva una AWS Security Token Service AssumeRole sesión. Este nombre aparece en los registros de auditoría, como los de CloudTrail. Tipo: <code>std::string</code> Obligatorio: no

AwsLongOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: <code>long</code> Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor <code>r</code>, para que el código de llamada pueda tomar posesión del objeto.

Propiedades	Description (Descripción)
	Tipo: long&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

AwsStringOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: std::string Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: long&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool

Propiedades	Description (Descripción)
	Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

DescribePlayerSessionsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “DescribePlayerSessionsResult” Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: <code>Aws::GameLift::Server::Model::DescribePlayerSessionsResult&&</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError”

Propiedades	Description (Descripción)
	Obligatorio: no

DescribePlayerSessionsResult

Una colección de objetos que contiene propiedades para cada sesión de jugador que se empareja con la solicitud.

Propiedades	Description (Descripción)
NextToken	Un token que indica el inicio de la siguiente página de resultado s secuencial. Utilice el token devuelto con una llamada anterior a esta operación. Para especificar el inicio del conjunto de resultados, no especifique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: <code>std::string</code> Obligatorio: sí
PlayerSessions	Tipo: <code>IList<the section called "PlayerSession"></code> Obligatorio:
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: <code>std::string&&</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente.

Propiedades	Description (Descripción)
	Tipo: the section called “GameLiftError”
	Obligatorio: no

GenericOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GenericOutcomeCallable

Este tipo de datos es un resultado genérico asíncrono. Tiene las siguientes propiedades:

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente.

Propiedades	Description (Descripción)
	Tipo: the section called “GameLiftError”
	Obligatorio: no

PlayerSession

Este tipo de datos representa la sesión de un jugador que Amazon GameLift Servers transfiere al servidor de juegos. Para obtener más información, consulte [PlayerSession](#).

Propiedades	Description (Descripción)
CreationTime	Tipo: long Obligatorio: no
FleetId	Tipo: std::string Obligatorio: no
GameSessionId	Tipo: std::string Obligatorio: no
IpAddress	Tipo: std::string Obligatorio: no
PlayerData	Tipo: std::string Obligatorio: no
PlayerId	Tipo: std::string Obligatorio: no
PlayerSessionId	Tipo: std::string Obligatorio: no

Propiedades	Description (Descripción)
Puerto	Tipo: <code>int</code> Obligatorio: no
Status	Estado de la sesión de juego para filtrar los resultados. Cuando <code>PlayerId</code> se proporciona una <code>PlayerSessionId</code> o, no <code>PlayerSessionStatusFilter</code> tiene ningún efecto en la respuesta. Tipo: una enumeración de <code>PlayerSessionStatus</code> . Entre los valores posibles se incluyen: • ACTIVO • COMPLETED • NOT_SET • RESERVED • TIEMPO DE ESPERA AGOTADO Obligatorio: no
TerminationTime	Tipo: <code>long</code> Obligatorio: no
DnsName	Tipo: <code>std::string</code> Obligatorio: no

StartMatchBackfillOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “StartMatchBackfillResult”

Propiedades	Description (Descripción)
	Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: <code>StartMatchBackfillResult</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

StartMatchBackfillResult

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
TicketId	Un identificador único de un ticket de emparejamiento Si no se especifica ningún ID de ticket aquí, Amazon GameLift Servers generará uno en forma de UUID. Utilice este identificador para realizar un seguimiento del estado del ticket de relleno del emparejamiento y recuperar los resultados del emparejamiento. Tipo: <code>std::string</code> Obligatorio: no

GetComputeCertificateOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “GetComputeCertificateResult” Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: <code>Aws::GameLift::Server::Model::GetComputeCertificateResult&&</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GetComputeCertificateResult

La ruta al certificado TLS de su recurso informático y el nombre de host del equipo.

Propiedades	Description (Descripción)
CertificatePath	La ruta al certificado TLS de su recurso informático. Cuando se utiliza una flota administrada por Amazon GameLift Servers, la ruta contiene los siguientes elementos: • <code>certificate.pem</code> : el certificado del usuario final. La cadena de certificados completa es la combinación del <code>certificateChain.pem</code> adjunto a este certificado. • <code>certificateChain.pem</code> : la cadena de certificados que contiene el certificado raíz y los certificados intermedios. • <code>rootCertificate.pem</code> : el certificado raíz. • <code>privateKey.pem</code> : la clave privada del certificado del usuario final. Tipo: <code>std::string</code> Obligatorio: no
ComputeName	El nombre del recurso informático. Tipo: <code>std::string</code> Obligatorio: no

GetFleetRoleCredentialsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “GetFleetRoleCredentialsResult” Obligatorio: no

Propiedades	Description (Descripción)
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: <code>Aws::GameLift::Server::Model::GetFleetRoleCredentialsResult</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GetFleetRoleCredentialsResult

Propiedades	Description (Descripción)
AccessKeyId	El ID de la clave de acceso para autenticar y proporcionar acceso a los recursos de AWS . Tipo: <code>string</code> Obligatorio: no
AssumedRoleId	El ID del usuario al que pertenece el rol de servicio. Tipo: <code>string</code> Obligatorio: no

Propiedades	Description (Descripción)
AssumedRoleUserArn	El nombre de recurso de Amazon (ARN) del usuario al que pertenece el rol de servicio. Tipo: <code>string</code> Obligatorio: no
Expiration	El tiempo que queda hasta que caduquen las credenciales de la sesión. Tipo: <code>DateTime</code> Obligatorio: no
SecretAccessKey	El ID de clave de acceso secreta para la autenticación. Tipo: <code>string</code> Obligatorio: no
SessionToken	Un token para identificar la sesión activa actual que interactúa con tus AWS recursos. Tipo: <code>string</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

Iniciar SDKOutcome

Note

`InitSDKOutcome` se devuelve solo cuando compila el SDK con la marca `std`. Si realiza la compilación con la marca `nostd`, se devuelve [the section called “GenericOutcome”](#) en su lugar.

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GameLiftError

Propiedades	Description (Descripción)
ErrorType	Tipo de error. Tipo: una enumeración de <code>GameLiftErrorType</code> . Obligatorio: no
ErrorMessage	El nombre del error. Tipo: <code>std::string</code> Obligatorio: no

Propiedades	Description (Descripción)
ErrorMessage	Mensaje de error. Tipo: <code>std::string</code> Obligatorio: no

Enums

Las enumeraciones definidas para el SDK del servidor para Amazon GameLift Servers (C++) se definen de la siguiente manera:

GameLiftErrorType

Valor de cadena que indica el tipo de error. Los valores válidos son:

- `BAD_REQUEST_EXCEPTION`
- `GAMESESSION_ID_NOT_SET`: no se ha establecido el ID de sesión de juego.
- `INTERNAL_SERVICE_EXCEPTION`
- `LOCAL_CONNECTION_FAILED`: se ha producido un error en la conexión local a Amazon GameLift Servers.
- `NETWORK_NOT_INITIALIZED`: la red no se ha inicializado.
- `SERVICE_CALL_FAILED`: se produjo un error en una llamada a un servicio. AWS
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_INVALID_URL`
- `WEBSOCKET_CONNECT_FAILURE_TIMEOUT`
- `ALREADY_INITIALIZED`: el servidor o cliente de Amazon GameLift Servers ya se inicializó con `Initialize()`.
- `FLEET_MISMATCH`: la flota de destino no coincide con la flota de `gameSession` o `playerSession`.
- `GAMELIFT_CLIENT_NOT_INITIALIZED`: el cliente de Amazon GameLift Servers no se ha inicializado.
- `GAMELIFT_SERVER_NOT_INITIALIZED`: el servidor de Amazon GameLift Servers no se ha inicializado.

- `GAME_SESSION_ENDED_FAILED`: el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego había finalizado.
- `GAME_SESSION_NOT_READY`: no se activó la sesión de juego del servidor de Amazon GameLift Servers.
- `GAME_SESSION_READY_FAILED`: el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego estaba lista.
- `INITIALIZATION_MISMATCH`: se llamó a un método de cliente después de `Server::Initialize()` o viceversa.
- `NOT_INITIALIZED`: el servidor o el cliente de Amazon GameLift Servers no se han iniciado con `Initialize()`.
- `NO_TARGET_ALIASID_SET`: no se ha establecido un `aliasId` de destino.
- `NO_TARGET_FLEET_SET`: no se ha establecido una flota de destino.
- `PROCESS_ENDING_FAILED`: el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está finalizando.
- `PROCESS_NOT_ACTIVE` — El proceso del servidor aún no está activo, no está vinculado a un y no puede aceptarlo ni procesarlo. `GameSession PlayerSessions`
- `PROCESS_NOT_READY`: el proceso de servidor aún no está listo para activarse.
- `PROCESS_READY_FAILED`: el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está listo.
- `SDK_VERSION_DETECTION_FAILED`: no se pudo detectar la versión del SDK.
- `STX_CALL_FAILED`: no se pudo realizar una llamada al componente de backend del servidor. `XStx`
- `STX_INITIALIZATION_FAILED`: el componente de backend del servidor no se pudo inicializar. `XStx`
- `UNEXPECTED_PLAYER_SESSION`: el servidor ha detectado una sesión de jugador no registrada.
- `WEBSOCKET_CONNECT_FAILURE`
- `WEBSOCKET_CONNECT_FAILURE_FORBIDDEN`
- `WEBSOCKET_CONNECT_FAILURE_INVALID_URL`
- `WEBSOCKET_CONNECT_FAILURE_TIMEOUT`
- `WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE`: error recuperable al enviar un mensaje al servicio. `GameLift WebSocket`

- `WEBSOCKET_SEND_MESSAGE_FAILURE`: error al enviar un mensaje al GameLift servicio. WebSocket
- `MATCH_BACKFILL_REQUEST_VALIDATION`: se ha producido un error en la validación de la solicitud.
- `PLAYER_SESSION_REQUEST_VALIDATION`: se ha producido un error en la validación de la solicitud.

PlayerSessionCreationPolicy

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos. Los valores válidos son:

- `ACCEPT_ALL`: se aceptan todas las sesiones de jugador nuevas.
- `DENY_ALL`: se rechazan todas las sesiones de jugador nuevas.
- `NOT_SET`: la sesión de juego no está configurada para aceptar o denegar sesiones de nuevos jugadores.

SDK 5.x del servidor de C++ para Amazon GameLift Servers: tipos de datos

Temas

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)

- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
Aws::GameLift::AwsStringOutcome Server::GetSdkVersion();
```

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto [the section called “AwsStringOutcome”](#). El objeto devuelto incluye el número de versión (por ejemplo, 5.0.0). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitMetrics()

Inicializa el sistema de métricas para recopilar los datos de rendimiento del servidor e informar sobre ellos. Para obtener unos resultados óptimos, llame a este método antes que [InitSDK\(\)](#) para habilitar la recopilación de métricas durante la inicialización del SDK.

Sintaxis

```
Aws::GameLift::GenericOutcome InitMetrics();  
Aws::GameLift::GenericOutcome InitMetrics(const  
    Aws::GameLift::Server::MetricsParameters &metricsParameters);
```


Parameters

[MetricsParameters](#) (opcional)

Parámetros de configuración de la recopilación de métricas. Si no se proporciona, utiliza valores predeterminados que pueden ser anulados por variables de entorno.

Valor devuelto

Si se ejecuta correctamente, devuelve un objeto [the section called “GenericOutcome”](#) que indica un resultado correcto. Si no funciona, devuelve un mensaje de error.

Ejemplo

```
// Initialize with default parameters (uses environment variables if available)
// Defaults: localhost:8125 for StatsD, localhost:8126 for crash reporter
// FlushInterval: 10000ms, MaxPacketSize: 512 bytes
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::InitMetrics();
if (outcome.IsSuccess()) {
    // Metrics system initialized successfully
}

// Initialize with custom parameters
Aws::GameLift::Server::MetricsParameters metricsParams("localhost", 8125, "crash-host",
8126, 5000, 1024);
Aws::GameLift::GenericOutcome customOutcome =
    Aws::GameLift::Server::InitMetrics(metricsParams);
if (customOutcome.IsSuccess()) {
    // Metrics system initialized with custom parameters
}
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers. Llame a este método en el lanzamiento, antes de cualquier otro paso de inicialización relacionado con Amazon GameLift Servers. Esta acción lee los parámetros del servidor desde el entorno de host para configurar la comunicación entre el proceso del servidor de juegos y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Si la compilación del servidor de juegos se va a implementar sin el agente de Amazon GameLift Servers en una flota de Amazon GameLift Servers Anywhere o flota de contenedores, llame a [InitSDK\(\)](#) y especifique un conjunto de parámetros de servidor.

Sintaxis

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK();
```

Valor devuelto

Devuelve un objeto [the section called “Iniciar SDKOutcome”](#) que indica si el proceso del servidor está listo para llamar a [ProcessReady\(\)](#).

Ejemplo

```
//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()

Inicializa el SDK de Amazon GameLift Servers. Llame a este método en el lanzamiento, antes de cualquier otro paso de inicialización relacionado con Amazon GameLift Servers. Para esta acción, es necesario que un conjunto de parámetros del servidor establezca la comunicación entre el proceso del servidor de juegos y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Si la compilación del servidor de juegos se va a implementar en una flota de EC2 administrada por Amazon GameLift Servers o en una flota de Amazon GameLift Servers Anywhere o flota de contenedores con el agente de Amazon GameLift Servers, llame a [InitSDK\(\)](#) sin especificar los parámetros del servidor.

Sintaxis

```
Server::InitSDKOutcome Server::initSdkOutcome = InitSDK(serverParameters);
```

Parameters

ServerParameters

Para inicializar un servidor de juegos en una flota de Amazon GameLift Servers Anywhere, cree un objeto `ServerParameters` con la siguiente información:

- La URL WebSocket utilizada para conectarse a tu servidor de juegos.
- El ID del proceso utilizado para alojar su servidor de juegos.
- El ID del proceso utilizado para alojar los procesos del servidor de juegos.
- El ID de la flota de Amazon GameLift Servers que contiene su recurso de computación de Amazon GameLift Servers Anywhere.
- El token de autorización generado por la operación de Amazon GameLift Servers.

Valor devuelto

Devuelve un objeto [the section called “Iniciar SDKOutcome”](#) que indica si el proceso del servidor está listo para llamar a [ProcessReady\(\)](#).

Note

Si las llamadas a `InitSDK()` no funcionan en las compilaciones de juegos implementadas en las flotas de Anywhere, compruebe el parámetro `ServerSdkVersion` que se utiliza al crear el recurso de compilación. Debe establecer este valor de forma explícita en la versión del SDK del servidor en uso. El valor predeterminado de este parámetro es 4.x, que no es compatible. Para resolver este problema, cree una compilación nueva e impleméntela en una flota nueva.

Ejemplo

Ejemplo de Amazon GameLift Servers Anywhere

```
//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()

Notifica a Amazon GameLift Servers que el proceso del servidor de juegos está listo para alojar sesiones de juego. Llame a este método después de invocar [InitSDK\(\)](#). Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters  
&processParameters);
```

Parameters

processParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra las implementaciones tanto de la función de llamada [ProcessReady\(\)](#) como de la función de delegación.

```

// 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()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Este método debe llamarse después de que el proceso del servidor esté listo para alojar una sesión de juego. Los parámetros especifican los nombres de funciones de devolución de llamada para que Amazon GameLift Servers llame en determinadas circunstancias. El código de servidor de juegos debe implementar estas funciones.

La llamada es asíncrona. Para realizar una llamada síncrona, utilice [ProcessReady\(\)](#). Consulte [Inicialización del proceso del servidor](#) para obtener más detalles.

Sintaxis

```
GenericOutcomeCallable ProcessReadyAsync(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Informa a Amazon GameLift Servers de que el proceso del servidor está finalizando. Llame a este método después de realizar todas las demás tareas de limpieza (lo que incluye el cierre de la sesión de juegos activa) y antes de finalizar el proceso. Según el resultado de `ProcessEnding()`, el proceso finaliza con éxito (0) o error (-1) y genera un evento de flota. Si el proceso finaliza con un error, el evento de flota generado será `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Sintaxis

```
Aws::GameLift::GenericOutcome processEndingOutcome =  
    Aws::GameLift::Server::ProcessEnding();
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

En este ejemplo, se llama a `ProcessEnding()` y `Destroy()` antes de finalizar el proceso del servidor con un código de salida correcto o erróneo.

```
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()

Notifica a Amazon GameLift Servers que el proceso del servidor ha iniciado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de la inicialización de todas las sesiones de juego.

Sintaxis

```
Aws::GameLift::GenericOutcome activateGameSessionOutcome =  
    Aws::GameLift::Server::ActivateGameSession();
```


Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de delegación `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();
}
```

UpdatePlayerSessionCreationPolicy()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores.

Sintaxis

```
GenericOutcome
UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCreationPolicy
newPlayerSessionPolicy);
```

Parameters

playerCreationSessionPolítica

Tipo: valor de `PlayerSessionCreationPolicy` [enumeración](#).

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
Aws::GameLift::GenericOutcome outcome =  
  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

GetGameSessionId()

Recupera el ID de la sesión de juego alojada por el proceso del servidor.

En el caso de los procesos inactivos que no se activan con una sesión de juego, la llamada devuelve [the section called “GameLiftError”](#).

Sintaxis

```
AwsStringOutcome GetGameSessionId()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto [the section called “AwsStringOutcome”](#). Si no funciona, devuelve un mensaje de error.

En el caso de los procesos inactivos que no se activan con una sesión de juego, la llamada devuelve Success=True y GameSessionId="".

Ejemplo

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetTerminationTime()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción después de recibir una

devolución de llamada `onProcessTerminate()` desde Amazon GameLift Servers. Amazon GameLift Servers llama a `onProcessTerminate()` por los motivos siguientes:

- Cuando el proceso del servidor informa de un mal estado o no responde a Amazon GameLift Servers.
- Al finalizar la instancia durante un evento de reducción vertical.
- [Cuando se finaliza una instancia debido a la interrupción de una instancia de spot.](#)

Sintaxis

```
AwsDateTimeOutcome GetTerminationTime()
```

Valor devuelto

Si el proceso se realiza correctamente, devuelve la hora de terminación como un objeto `AwsDateTimeOutcome`. El valor es la hora de terminación expresado en ciclos transcurridos desde `0001 00:00:00`. Por ejemplo, el valor de la fecha y hora `2020-09-13 12:26:40 -000Z` es igual a `637355968000000000` ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Si el proceso no ha recibido un `ProcessParameters`. `OnProcessTerminate()` devolución de llamada, se devuelve un mensaje de error. Para obtener más información sobre cómo cerrar un proceso de servidor, consulte [Respuesta a una notificación de cierre del proceso del servidor](#).

Ejemplo

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

AcceptPlayerSession()

Notifica a Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión de jugador es válido. Una vez validada la sesión del jugador, Amazon GameLift Servers cambia el estado del espacio de jugador de `RESERVADO` a `ACTIVO`.

Sintaxis

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo gestiona una solicitud de conexión que incluye la validación y el rechazo de una sesión de jugador no válida. 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()

Informa a Amazon GameLift Servers de que un jugador se ha desconectado del proceso del servidor. Como respuesta, Amazon GameLift Servers cambia el espacio del jugador para que esté disponible.

Sintaxis

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice este método para obtener información sobre los siguientes elementos:

- Una sesión para un jugador
- Todas las sesiones del jugador en una sesión de juego
- Todas las sesiones de jugador están asociadas a un único ID de jugador

Sintaxis

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
    describePlayerSessionsRequest)
```

Parameters

[DescribePlayerSessionsRequest](#)

Un objeto [the section called “DescribePlayerSessionsRequest”](#) que describe las sesiones de jugador que recuperar.

Valor devuelto

Si funciona correctamente, devuelve un objeto [the section called “DescribePlayerSessionsOutcome”](#) que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud.

Ejemplo

En este ejemplo se solicitan todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Al omitir `NextToken` establecer el valor límite en 10, Amazon GameLift Servers devuelve los registros de las sesiones de los primeros 10 jugadores que coincidan con la solicitud.

```
// 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()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan nuevos jugadores, Amazon GameLift Servers ofrece datos actualizados del emparejador mediante la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest
startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

StartMatchBackfillRequest Objeto que comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers generará uno.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se encuentra en los datos del emparejador de la sesión de juego.
- El ID de la sesión de juego que se va a reponer.
- Datos del emparejador disponibles para los jugadores actuales de la sesión de juego.

Valor devuelto

Devuelve un objeto [the section called “StartMatchBackfillOutcome”](#) con el ID del ticket de reposición de emparejamiento o un error con un mensaje de error.

Ejemplo

```
// 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()

Cancela una solicitud de reposición de emparejamiento activa. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Sintaxis

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

[StopMatchBackfillRequest](#)

Un StopMatchBackfillRequest objeto que identifique el billete de emparejamiento que se va a cancelar:

- ID del ticket que se asignará a la solicitud de reposición
- El emparejador al que se envió la solicitud de reposición
- La sesión de juego asociada a la solicitud de reposición.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Recupera la ruta al certificado TLS utilizado para cifrar la conexión de red entre el recurso de computación de Amazon GameLift Servers Anywhere y Amazon GameLift Servers. Puede utilizar la ruta del certificado al registrar el dispositivo informático en una flota de Amazon GameLift Servers Anywhere. Para obtener más información, consulte, [RegisterCompute](#).

Sintaxis

```
GetComputeCertificateOutcome Server::GetComputeCertificate()
```

Valor devuelto

Devuelve [the section called “GetComputeCertificateOutcome”](#).

Ejemplo

```
Aws::GameLift::GetComputeCertificateOutcome certificate =  
    Aws::GameLift::Server::GetComputeCertificate();
```

GetFleetRoleCredentials()

Recupera las credenciales del rol de IAM que autorizan a Amazon GameLift Servers a interactuar con otros Servicios de AWS. Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Sintaxis

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest  
    request);
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Valor devuelto

Devuelve un objeto [the section called “GetFleetRoleCredentialsOutcome”](#).

Ejemplo

```
// 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);
```

En este ejemplo, se muestra el uso del valor `RoleSessionName` opcional para asignar un nombre a la sesión de credenciales con fines de auditoría. Si no proporciona un nombre de sesión de rol, se utilizará el valor predeterminado `[host-id] "[fleet-id]-»`.

```
// 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);
```

Destroy()

Libera de la memoria el SDK del servidor de juegos de Amazon GameLift Servers. Como práctica recomendada, llame a este método después de `ProcessEnding()` y antes de finalizar el proceso. Si utiliza una flota de Anywhere y no va a finalizar los procesos del servidor después de cada sesión de juego, llame a `Destroy()` y, a continuación, a `InitSDK()` para reinicializar antes de informar a Amazon GameLift Servers de que el proceso está listo para organizar una sesión de juego con `ProcessReady()`.

Sintaxis

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameters

No hay parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
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 del servidor de C# para Amazon GameLift Servers: acciones

Utilice la referencia del SDK 5.x del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#). Si utiliza el complemento de Amazon GameLift Servers para Unity, consulte también [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

SDK 5.x del servidor de C# para Amazon GameLift Servers: tipos de datos

Utilice la referencia del SDK 5.x del servidor de C# de Amazon GameLift Servers para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#). Si utiliza el complemento de Amazon GameLift Servers para Unity, consulte también [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#).

[SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)

Tipos de datos

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)

- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Jugador](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)
- [AttributeValue](#)
- [AwsStringOutcome](#)
- [GenericOutcome](#)
- [MetricsManagerOutcome](#)
- [DescribePlayerSessionsOutcome](#)
- [DescribePlayerSessionsResult](#)
- [PlayerSession](#)
- [StartMatchBackfillOutcome](#)
- [StartMatchBackfillResult](#)
- [GetComputeCertificateOutcome](#)
- [GetComputeCertificateResult](#)
- [GetFleetRoleCredentialsOutcome](#)
- [GetFleetRoleCredentialsResult](#)
- [AwsDateTimeOutcome](#)
- [GameLiftError](#)
- [Enums](#)

LogParameters

Utilice este tipo de datos para identificar los archivos generados durante una sesión de juegos que desee cargar en Amazon GameLift Servers cuando finalice la sesión de juego. El servidor de juegos comunica `LogParameters` to a Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Propiedades	Descripción

LogPaths	Lista de las rutas de directorio a archivos de registro del servidor de juegos que desea que Amazon GameLift Servers almacene para futuros accesos. El proceso del servidor genera esos archivos durante una sesión de juego. Defina los nombres y las rutas de los archivos en el servidor de juegos y almacénelos en el directorio raíz de compilación del juego. Las rutas del registro deben ser absolutas. Por ejemplo, si la compilación del juego almacena los registros de sesión de juego en una ruta del tipo <code>MyGame\sessionLogs\</code> , la ruta sería <code>c:\game\MyGame\sessionLogs</code> en una instancia de Windows. Tipo: <code>List<String></code> Obligatorio: no

MetricsParameters

Utilice este tipo de datos para configurar la recopilación de métricas y los informes de bloqueo del servidor del juego. El servidor de juegos comunica `MetricsParameters` a Amazon GameLift Servers en una llamada a [InitMetrics\(\)](#).

Propiedades	Descripción
StatsdHost	Nombre de host o dirección IP del servidor de StatsD para la recopilación de métricas. Tipo: <code>String</code> Obligatorio: no
StatsdPort	Número de puerto del servidor de StatsD para la recopilación de métricas.

	Tipo: Integer Obligatorio: no
CrashReporterHost	Nombre de host o dirección IP del servidor que notifica el bloqueo. Tipo: String Obligatorio: no
CrashReporterPort	Número de puerto del servidor que notifica el bloqueo. Tipo: Integer Obligatorio: no
FlushIntervalMs	Intervalo en milisegundos para vaciar los datos de las métricas al servidor. Tipo: Integer Obligatorio: no
MaxPacketSize	Tamaño máximo en bytes de los paquetes de métricas enviados al servidor. Tipo: Integer Obligatorio: no

ProcessParameters

Este tipo de datos contiene el conjunto de parámetros enviado a Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Propiedades	Descripción
LogParameters	El objeto con una lista de rutas de directorio a archivos de registro de la sesión de juego.

	Tipo: <code>Aws::GameLift::Server::</code> LogParameters Obligatorio: sí
OnHealthCheck	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos. Después de llamar a esta función, Amazon GameLift Servers espera una respuesta durante 60 segundos y si no recibe ninguna, Amazon GameLift Servers registra el proceso del servidor como en mal estado. Tipo: <code>void OnHealthCheckDelegate()</code> Obligatorio: sí
OnProcessTerminate	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para forzar el cierre del proceso de servidor. Después de llamar a esta función, Amazon GameLift Servers espera cinco minutos hasta que el proceso del servidor se cierre y responde con una llamada a ProcessEnding() antes de cerrar el proceso del servidor. Tipo: <code>void OnProcessTerminate Delegate()</code> Obligatorio: sí

OnStartGameSession

El nombre de la función de devolución de llamada que se Amazon GameLift Servers invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a la solicitud del cliente. [CreateGameSession](#) La función de devolución de llamada obtiene un objeto [GameSession](#).

Tipo: void OnStartGameSession
Delegate([GameSession](#))

Obligatorio: sí

OnUpdateGameSession

Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de reposición de emparejamiento para proporcionar datos actualizados del emparejador. Pasa un objeto [GameSession](#), una actualización de estado (updateReason) y el ID del ticket de reposición de emparejamiento.

Tipo: void OnUpdateGameSessionDelegate
([UpdateGameSession](#))

Obligatorio: no

Puerto

El número de puerto en el que escucha el proceso del servidor para recibir conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: Integer

Obligatorio: sí

UpdateGameSession

La información actualizada de un objeto de sesión de juego incluido el motivo por el que se actualizó la sesión de juego. Si la actualización está relacionada con una acción de reposición del emparejamiento, este tipo de datos incluye el ID de ticket de reposición.

Propiedades	Descripción
GameSession	Un objeto GameSession. El objeto <code>GameSession</code> contiene propiedades que describen una sesión de juego. Tipo: <code>GameSession</code> <code>GameSession()</code> Obligatorio: sí
UpdateReason	El motivo por el que se actualiza la sesión de juego. Tipo: <code>UpdateReason</code> <code>UpdateReason()</code> Obligatorio: sí
BackfillTicketId	El ID de ticket de reposición que intenta actualizar la sesión de juego.

Propiedades	Descripción
	Tipo: <code>String</code>
	Obligatorio: sí

GameSession

Detalles de una sesión de juego.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juego. El ARN de una sesión de juego tiene el siguiente formato: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Tipo: <code>String</code> Obligatorio: no
Name	Una etiqueta descriptiva de la sesión de juego. Tipo: <code>String</code> Obligatorio: no
FleetId	Un identificador único para la flota en la que se ejecuta la sesión de juego. Tipo: <code>String</code> Obligatorio: no
MaximumPlayerSessionCount	El número máximo de conexiones de jugadores a la sesión de juego. Tipo: <code>Integer</code>

Propiedades	Descripción
	Obligatorio: no
Puerto	El número de puerto de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: Integer Obligatorio: no
IpAddress	La dirección IP del servidor de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: String Obligatorio: no
GameSessionData	Un conjunto de propiedades de sesión de juego personalizadas, formateadas como un valor de una sola cadena. Tipo: String Obligatorio: no

Propiedades	Descripción
MatchmakerData	La información sobre el proceso de emparejamiento que se utilizó para crear la sesión de juego, en sintaxis JSON, con formato como cadena. Además de la configuración de emparejamiento utilizada, contiene datos sobre todos los jugadores asignados al emparejamiento, incluidos los atributos de los jugadores y las asignaciones de los equipos. Tipo: <code>String</code> Obligatorio: no
GameProperties	Un conjunto de propiedades personalizadas de una sesión de juego, con formato como pares clave-valor. Estas propiedades se pasan a una solicitud de iniciar una nueva sesión de juego. Tipo: <code>Dictionary<string, string></code> Obligatorio: no

Propiedades	Descripción
DnsName	El identificador de DNS asignado a la instancia que ejecuta la sesión de juego. Los valores tienen formato siguiente: • Flotas habilitadas para TLS: <unique identifier>.<region identifier>.amazongamelift.com . • Non-TLS-enabled flotas:ec2-<unique identifier>.compute.amazonaws.com . Cuando se conecte a una sesión de juego que se ejecute en una flota habilitada de TLS, debe utilizar el nombre de DNS, no la dirección IP. Tipo: String Obligatorio: no

ServerParameters

La información utilizada para mantener la conexión entre un servidor de Amazon GameLift Servers Anywhere y el servicio Amazon GameLift Servers. Esta información se utiliza al inicializar nuevos procesos de servidor con [InitSDK\(\)](#). Para los servidores alojados en instancias de EC2 administradas por Amazon GameLift Servers, utilice un objeto vacío.

Propiedades	Descripción
WebSocketUrl	El <code>GameLiftServerSdkEndpoint</code> devuelto cuando se llama a <code>RegisterCompute</code> como parte de Amazon GameLift Servers Anywhere. Tipo: String

Propiedades	Descripción
	Obligatorio: sí
ProcessId	Un identificador único registrado en el proceso del servidor que aloja el juego. Tipo: <code>String</code> Obligatorio: sí
HostId	Un identificador único para el alojamiento con el servidor que procesa el juego. El HostID es el que ComputeName se utilizó al registrar el ordenador. Para obtener más información, consulte: RegisterCompute Tipo: <code>String</code> Obligatorio: sí
FleetId	El ID de la flota de la flota en la que está registrado el recurso informático. Para obtener más información, consulte, RegisterCompute. Tipo: <code>String</code> Obligatorio: sí
AuthToken	El token de autenticación generado por Amazon GameLift Servers que autentica el servidor en Amazon GameLift Servers. Para obtener más información, consulte, GetComputeAuthToken. Tipo: <code>String</code> Obligatorio: sí

StartMatchBackfillRequest

Información utilizada para crear una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información a Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	El identificador único de la sesión de juego. El GetGameSessionId de la operación de la API devuelve el identificador en formato de ARN. Tipo: <code>String</code> Obligatorio: sí
MatchmakingConfigurationArn	El identificador único, con formato de ARN, que el emparejador utiliza para esta solicitud. El ARN del emparejador para la sesión de juego original se encuentra en el objeto de sesión de juego en la propiedad de datos del emparejador. Puede obtener más información sobre los datos del emparejador en Trabajo con datos del emparejador. Tipo: <code>String</code> Obligatorio: sí
Players	Un conjunto de datos que representa a todos los jugadores que están actualmente en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Tipo: <code>List<Player></code> Obligatorio: sí

Propiedades	Descripción
TicketId	El identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no proporciona un valor, Amazon GameLift Servers generará uno. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario. Tipo: <code>String</code> Obligatorio: no

Jugador

Representa a un jugador en el emparejamiento. Al iniciar una solicitud de emparejamiento, un jugador tiene un ID de jugador, atributos y, posiblemente, datos de latencia. Amazon GameLift Servers añade la información de los equipos tras realizar un emparejamiento.

Propiedades	Descripción
LatencyInMS	Un conjunto de valores expresados en milisegundos que indican la cantidad de latencia que experimenta un jugador cuando se conecta a una ubicación. Si se utiliza esta propiedad, el jugador solo se empareja con las ubicaciones que aparecen. Si un creador de emparejamientos tiene una regla que evalúa la latencia de los jugadores, estos deben informar de la latencia para ser emparejados. Tipo: <code>Dictionary<string, int></code> Obligatorio: no

Propiedades	Descripción
PlayerAttributes	Una colección de pares de clave-valor que contienen información del jugador para su uso en el emparejamiento. Las claves de atributos del jugador deben coincidir con las PlayerAttributes utilizadas en un conjunto de reglas de emparejamiento. Para obtener más información sobre los atributos de los jugadores, consulte Attribute Value. Tipo: Dictionary<string, Attribute Value Obligatorio: no
PlayerId	Un identificador único de un jugador. Tipo: String Obligatorio: no
Equipo	El nombre del equipo al que está asignado el jugador en un emparejamiento. Defina el nombre del equipo se define en el conjunto de reglas de emparejamiento. Tipo: String Obligatorio: no

DescribePlayerSessionsRequest

Este tipo de datos se utiliza para especificar qué sesión o sesiones de jugador recuperar. Se puede usar de varias maneras: (1) proporcionar una PlayerSessionId para solicitar una sesión de jugador específica; (2) proporcionar una GameSessionId para solicitar todas las sesiones de jugador de la sesión de juego especificada; o (3) proporcionar una PlayerId para solicitar todas las sesiones de

jugador para el jugador especificado. Para grandes recopilaciones de sesiones de jugador, utilice los parámetros de paginación para recuperar resultados como páginas secuenciales.

Propiedades	Descripción
GameSessionId	El identificador único de la sesión de juego. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de la sesión de juego es el siguiente: <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> . El valor de la <code><ID string></code> es una cadena de ID personalizada (si se especificó una cuando se creó la sesión de juego) o una cadena generada. Tipo: <code>String</code> Obligatorio: no
PlayerSessionId	El identificador único de una sesión de jugador. Tipo: <code>String</code> Obligatorio: no
PlayerId	El identificador único de un jugador. Consulte Generación de ID de jugador. Tipo: <code>String</code> Obligatorio: no
PlayerSessionStatusFilter	El estado de la sesión de jugador para filtrar los resultados. Los posibles estados de sesión de jugador son: • RESERVADO: se ha recibido la solicitud de sesión del jugador, pero el jugador aún no

Propiedades	Descripción
	se ha conectado al servidor y and/or se ha validado el proceso. • ACTIVE: el proceso del servidor ha validado el jugador y actualmente está conectado. • COMPLETED: ha caído la conexión del jugador. • TIEMPO DE ESPERA: se recibió una solicitud de sesión de un jugador, pero el jugador que no and/or se conectó no se validó dentro del límite de tiempo de espera (60 segundos). Tipo: String Obligatorio: no
NextToken	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: String Obligatorio: no
Límite	El número máximo de resultados que devolver. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: int Obligatorio: no

StopMatchBackfillRequest

Información utilizada para cancelar una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	El identificador único de sesión de juego de la solicitud que se va a cancelar. Tipo: <code>string</code> Obligatorio: sí
MatchmakingConfigurationArn	El identificador único del emparejador al que se envió esta solicitud. Tipo: <code>string</code> Obligatorio: sí
TicketId	El identificador único del ticket de solicitud de reposición que se va a cancelar. Tipo: <code>string</code> Obligatorio: sí

GetFleetRoleCredentialsRequest

Este tipo de datos proporciona al servidor del juego un acceso limitado al resto de tus recursos. AWS Para obtener más información, consulte, [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Propiedades	Descripción
RoleArn	El nombre de recurso de Amazon (ARN) de la función de servicio que amplía el acceso limitado a sus AWS recursos.

Propiedades	Descripción
	Tipo: <code>string</code>
	Obligatorio: sí
RoleSessionName	El nombre de la sesión que describe el uso de las credenciales del rol.
	Tipo: <code>string</code>
	Obligatorio: no

AttributeValue

Utilice estos valores en pares de clave-valor de atributo [Jugador](#). Este objeto le permite especificar un valor de atributo mediante cualquiera de los tipos de datos válidos: cadena, número, matriz de cadenas o mapa de datos. Cada objeto `AttributeValue` puede utilizar solo una de las propiedades disponibles.

Propiedades	Description (Descripción)
attrType	Especifica el tipo de valor del atributo.
	Tipo: un valor de enum de <code>AttrType</code> .
	Obligatorio: no
S	Representa un valor de atributo de cadena.
	Tipo: <code>string</code>
	Obligatorio: sí
N	Representa un valor de atributo numérico.
	Tipo: <code>double</code>
	Obligatorio: sí
SL	Representa una matriz de valores de atributos de cadena.

Propiedades	Description (Descripción)
	Tipo: <code>string[]</code> Obligatorio: sí
SDM	Representa un diccionario de claves de cadena y valores dobles. Tipo: <code>Dictionary<string, double></code> Obligatorio: sí

AwsStringOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: <code>string</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GenericOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

MetricsManagerOutcome

Representa el resultado de una llamada [InitMetrics\(\)](#). Contiene una MetricsManager instancia de éxito o información de error en caso de error.

Resultado	La MetricsManager instancia para recopilar métricas e informar sobre ellas. Tipo: <code>MetricsManager</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError”

Obligatorio: no

DescribePlayerSessionsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “DescribePlayerSessionsResult” Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

DescribePlayerSessionsResult

Propiedades	Description (Descripción)
NextToken	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: string Obligatorio: sí

Propiedades	Description (Descripción)
PlayerSessions	Una colección de objetos que contiene propiedades para cada sesión de jugador que se empareja con la solicitud. Tipo: <code>ICollection<the section called "PlayerSession"></code> Obligatorio:
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called "GameLiftError" Obligatorio: no

PlayerSession

Propiedades	Description (Descripción)
CreationTime	Tipo: <code>long</code> Obligatorio: sí
FleetId	Tipo: <code>string</code> Obligatorio: sí
GameSessionId	Tipo: <code>string</code> Obligatorio: sí
IpAddress	Tipo: <code>string</code>

Propiedades	Description (Descripción)
	Obligatorio: sí
PlayerData	Tipo: <code>string</code> Obligatorio: sí
PlayerId	Tipo: <code>string</code> Obligatorio: sí
PlayerSessionId	Tipo: <code>string</code> Obligatorio: sí
Puerto	Tipo: <code>int</code> Obligatorio: sí
Status	Tipo: una enumeración de <code>PlayerSessionStatus</code> . Obligatorio: sí
TerminationTime	Tipo: <code>long</code> Obligatorio: sí
DnsName	Tipo: <code>string</code> Obligatorio: sí

StartMatchBackfillOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called "StartMatchBackfillResult"

Propiedades	Description (Descripción)
	Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called "GameLiftError" Obligatorio: no

StartMatchBackfillResult

Propiedades	Description (Descripción)
TicketId	Tipo: <code>string</code> Obligatorio: sí

GetComputeCertificateOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called "GetComputeCertificateResult" Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code>

Propiedades	Description (Descripción)
	Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GetComputeCertificateResult

La ruta al certificado TLS de su recurso informático y el nombre de host del equipo.

Propiedades	Description (Descripción)
CertificatePath	Tipo: string Obligatorio: sí
ComputeName	Tipo: string Obligatorio: sí

GetFleetRoleCredentialsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “GetFleetRoleCredentialsResult” Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool

Propiedades	Description (Descripción)
	Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GetFleetRoleCredentialsResult

Propiedades	Description (Descripción)
AccessKeyId	El ID de la clave de acceso para autenticar y proporcionar acceso a los recursos de AWS . Tipo: <code>string</code> Obligatorio: no
AssumedRoleId	El ID del usuario al que pertenece el rol de servicio. Tipo: <code>string</code> Obligatorio: no
AssumedRoleUserArn	El nombre de recurso de Amazon (ARN) del usuario al que pertenece el rol de servicio. Tipo: <code>string</code> Obligatorio: no
Expiration	El tiempo que queda hasta que caduquen las credenciales de la sesión. Tipo: <code>DateTime</code>

Propiedades	Description (Descripción)
	Obligatorio: no
SecretAccessKey	El ID de clave de acceso secreta para la autenticación. Tipo: <code>string</code> Obligatorio: no
SessionToken	Un token para identificar la sesión activa actual que interactúa con tus AWS recursos. Tipo: <code>string</code> Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

AwsDateTimeOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: <code>DateTime</code> Obligatorio: no

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

GameLiftError

Propiedades	Description (Descripción)
ErrorType	Tipo de error. Tipo: una enumeración de <code>GameLiftErrorType</code> . Obligatorio: no
ErrorMessage	El nombre del error. Tipo: <code>string</code> Obligatorio: no
ErrorName	Mensaje de error. Tipo: <code>string</code> Obligatorio: no

Enums

Las enumeraciones definidas para el SDK del servidor para Amazon GameLift Servers (C#) se definen de la siguiente manera:

AttrType

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

Valor de cadena que indica el tipo de error. Los valores válidos son:

- SERVICE_CALL_FAILED: se produjo un error en una llamada a un AWS servicio.
- LOCAL_CONNECTION_FAILED: se ha producido un error en la conexión local a Amazon GameLift Servers.
- NETWORK_NOT_INITIALIZED: la red no se ha inicializado.
- GAMESESSION_ID_NOT_SET: no se ha establecido el ID de sesión de juego.
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION
- ALREADY_INITIALIZED: el servidor o cliente de Amazon GameLift Servers ya se inicializó con Initialize().
- FLEET_MISMATCH: la flota de destino no coincide con la flota de gameSession o playerSession.
- GAMELIFT_CLIENT_NOT_INITIALIZED: el cliente de Amazon GameLift Servers no se ha inicializado.
- GAMELIFT_SERVER_NOT_INITIALIZED: el servidor de Amazon GameLift Servers no se ha inicializado.
- GAME_SESSION_ENDED_FAILED: el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego había finalizado.
- GAME_SESSION_NOT_READY: no se activó la sesión de juego del servidor de Amazon GameLift Servers.

- **GAME_SESSION_READY_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego estaba lista.
- **INITIALIZATION_MISMATCH:** se llamó a un método de cliente después de `Server::Initialize()` o viceversa.
- **NOT_INITIALIZED:** el servidor o el cliente de Amazon GameLift Servers no se han iniciado con `Initialize()`.
- **NO_TARGET_ALIASID_SET:** no se ha establecido un `aliasId` de destino.
- **NO_TARGET_FLEET_SET:** no se ha establecido una flota de destino.
- **PROCESS_ENDING_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está finalizando.
- **PROCESS_NOT_ACTIVE** — El proceso del servidor aún no está activo, no está vinculado a un y no puede aceptarlo ni procesarlo. `GameSession PlayerSessions`
- **PROCESS_NOT_READY:** el proceso de servidor aún no está listo para activarse.
- **PROCESS_READY_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está listo.
- **SDK_VERSION_DETECTION_FAILED:** no se pudo detectar la versión del SDK.
- **STX_CALL_FAILED:** no se pudo realizar una llamada al componente de backend del servidor. `XStx`
- **STX_INITIALIZATION_FAILED:** el componente de backend del servidor no se pudo inicializar. `XStx`
- **UNEXPECTED_PLAYER_SESSION:** el servidor ha detectado una sesión de jugador no registrada.
- **WEBSOCKET_CONNECT_FAILURE**
- **WEBSOCKET_CONNECT_FAILURE_FORBIDDEN**
- **WEBSOCKET_CONNECT_FAILURE_INVALID_URL**
- **WEBSOCKET_CONNECT_FAILURE_TIMEOUT**
- **WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE:** error recuperable al enviar un mensaje al servicio. `GameLift WebSocket`
- **WEBSOCKET_SEND_MESSAGE_FAILURE:** error al enviar un mensaje al GameLift servicio. `WebSocket`
- **MATCH_BACKFILL_REQUEST_VALIDATION:** se ha producido un error en la validación de la solicitud.

- `PLAYER_SESSION_REQUEST_VALIDATION`: se ha producido un error en la validación de la solicitud.

PlayerSessionCreationPolicy

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos. Los valores válidos son:

- `ACCEPT_ALL`: se aceptan todas las sesiones de jugador nuevas.
- `DENY_ALL`: se rechazan todas las sesiones de jugador nuevas.
- `NOT_SET`: la sesión de juego no está configurada para aceptar o denegar sesiones de nuevos jugadores.

PlayerSessionStatus

- `ACTIVE`
- `COMPLETED`
- `NOT_SET`
- `RESERVED`
- `TIMEDOUT`

SDK 5.x del servidor de C# para Amazon GameLift Servers: tipos de datos

Temas

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)

- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
AwsStringOutcome GetSdkVersion();
```

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto [the section called “AwsStringOutcome”](#). La cadena devuelta solo incluye el número de versión (por ejemplo, 5.0.0). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

InitMetrics()

Inicializa el sistema de métricas para recopilar los datos de rendimiento del servidor e informar sobre ellos. Para obtener unos resultados óptimos, llame a este método antes que [InitSDK\(\)](#) para habilitar la recopilación de métricas durante la inicialización del SDK.

Sintaxis

```
MetricsManagerOutcome InitMetrics();  
MetricsManagerOutcome InitMetrics(MetricsParameters metricsParameters);
```

Parameters

[MetricsParameters](#) (opcional)

Parámetros de configuración de la recopilación de métricas. Si no se proporciona, utiliza valores predeterminados que pueden ser anulados por variables de entorno.

Valor devuelto

Si tiene éxito, devuelve un [the section called “MetricsManagerOutcome”](#) objeto que contiene la MetricsManager instancia. Si no funciona, devuelve un mensaje de error.

Ejemplo

```
// Initialize with default parameters (uses environment variables if available)
var outcome = GameLiftServerAPI.InitMetrics();
if (outcome.Success) {
    var metricsManager = outcome.Result;
} else {
    Console.WriteLine("Failed to initialize metrics: " + outcome.Error.ErrorMessage);
}

// Initialize with custom parameters
var metricsParams = new MetricsParameters("localhost", 8125, "crash-host", 9999, 1000,
    1024);
var customOutcome = GameLiftServerAPI.InitMetrics(metricsParams);
if (customOutcome.Success) {
    var metricsManager = customOutcome.Result;
}
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers para una flota de EC2 administrada. Llame a este método en el lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers. Este método lee parámetros de servidor del entorno de host para configurar la comunicación entre el servidor y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Sintaxis

```
GenericOutcome InitSDK();
```

Valor devuelto

Si tiene éxito, devuelve un `InitSdkOutcome` objeto para indicar que el proceso del servidor está listo para la llamada [ProcessReady\(\)](#).

Ejemplo

```
//Call InitSDK to establish a local connection with the GameLift agent to enable
further communication.
GenericOutcome initSDKOutcome = GameLiftServerAPI.InitSDK();
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers para una flota de Anywhere. Llame a este método en el lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers. Este método requiere que los parámetros del servidor explícito configure la comunicación entre el servidor y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Sintaxis

```
GenericOutcome InitSDK(ServerParameters serverParameters);
```

Parameters

[ServerParameters](#)

Para inicializar un servidor de juegos en una flota de Amazon GameLift Servers Anywhere, cree un objeto `ServerParameters` con la siguiente información:

- La URL WebSocket utilizada para conectarse al servidor del juego.
- El ID del proceso utilizado para alojar su servidor de juegos.
- El ID del proceso utilizado para alojar los procesos del servidor de juegos.
- El ID de la flota de Amazon GameLift Servers que contiene su recurso de computación de Amazon GameLift Servers Anywhere.
- El token de autorización generado por la operación de Amazon GameLift Servers.

Valor devuelto

Si tiene éxito, devuelve un `InitSdkOutcome` objeto que indica que el proceso del servidor está listo para la llamada [ProcessReady\(\)](#).

Note

Si las llamadas a `InitSDK()` no funcionan en las compilaciones de juegos implementadas en las flotas de Anywhere, compruebe el parámetro `ServerSdkVersion` que se utiliza al crear el recurso de compilación. Debe establecer este valor de forma explícita en la versión del SDK del servidor en uso. El valor predeterminado de este parámetro es 4.x, que no es compatible. Para resolver este problema, cree una compilación nueva e impleméntela en una flota nueva.

Ejemplo

```
//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()

Notifica a Amazon GameLift Servers que el proceso del servidor de juegos está listo para alojar sesiones de juego. Llame a este método después de invocar [InitSDK\(\)](#). Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameters

[ProcessParameters](#)

Un objeto `ProcessParameters` contiene información sobre el proceso del servidor.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra las implementaciones tanto de la función de método como de la función de delegación.

```
// 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()

Informa a Amazon GameLift Servers de que el proceso del servidor está finalizando. Llame a este método después de realizar todas las demás tareas de limpieza (lo que incluye el cierre de la sesión de juegos activa) y antes de finalizar el proceso. Según el resultado de `ProcessEnding()`, el proceso finaliza con éxito (0) o error (-1) y genera un evento de flota. Si el proceso finaliza con un error, el evento de flota generado será `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Sintaxis

```
GenericOutcome ProcessEnding()
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

En este ejemplo, se llama a `ProcessEnding()` y `Destroy()` antes de finalizar el proceso del servidor con un código de salida correcto o erróneo.

```
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()

Notifica a Amazon GameLift Servers que el proceso del servidor ha iniciado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de la inicialización de todas las sesiones de juego.

Sintaxis

```
GenericOutcome ActivateGameSession()
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de delegación `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()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores.

Sintaxis

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy
playerSessionPolicy)
```

Parameters

playerSessionPolicy

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos.

Los valores válidos son:

- `ACCEPT_ALL`: se aceptan todas las sesiones de jugador nuevas.
- `DENY_ALL`: se rechazan todas las sesiones de jugador nuevas.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
GenericOutcome updatePlayerSessionPolicyOutcome =  
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

GetGameSessionId()

Recupera el ID de la sesión de juego alojada por el proceso del servidor.

En el caso de los procesos inactivos que no se activan con una sesión de juego, la llamada devuelve [the section called “GameLiftError”](#).

Sintaxis

```
AwsStringOutcome GetGameSessionId()
```

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto [the section called “AwsStringOutcome”](#). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
AwsStringOutcome getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetTerminationTime()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción después de recibir una devolución de llamada `onProcessTerminate()` desde Amazon GameLift Servers. Amazon GameLift Servers llama a `onProcessTerminate()` por los motivos siguientes:

- Cuando el proceso del servidor informa de un mal estado o no responde a Amazon GameLift Servers.
- Al finalizar la instancia durante un evento de reducción vertical.
- [Cuando se finaliza una instancia debido a la interrupción de una instancia de spot.](#)

Sintaxis

```
AwsDateTimeOutcome GetTerminationTime()
```

Valor devuelto

Si el proceso se realiza correctamente, devuelve la hora de terminación como un objeto [the section called “AwsDateTimeOutcome”](#). El valor es la hora de terminación expresado en ciclos transcurridos desde 0001 00:00:00. Por ejemplo, el valor de la fecha y hora 2020-09-13 12:26:40 -000Z es igual a 6373559680000000000 ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Ejemplo

```
AwsDateTimeOutcome getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```

AcceptPlayerSession()

Notifica a Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión de jugador es válido. Una vez validada la sesión del jugador, Amazon GameLift Servers cambia el estado del espacio de jugador de RESERVADO a ACTIVO.

Sintaxis

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

ID único que se emite GameLift cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra una función para gestionar una solicitud de conexión, incluida la validación y el rechazo de una sesión de jugador no válida. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId)
{
```

```
GenericOutcome acceptPlayerSessionOutcome =
GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
if(acceptPlayerSessionOutcome.Success)
{
    connectionToSessionMap.emplace(connection, playerSessionId);
    connection.Accept();
}
else
{
    connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);
}
}
```

RemovePlayerSession()

Informa a Amazon GameLift Servers de que un jugador se ha desconectado del proceso del servidor. Como respuesta, Amazon GameLift Servers cambia el espacio del jugador para que esté disponible.

Sintaxis

```
GenericOutcome RemovePlayerSession(String playerSessionId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
GenericOutcome removePlayerSessionOutcome =
GameLiftServerAPI.RemovePlayerSession(playerSessionId);
```

DescribePlayerSessions()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice esta acción para obtener información para una única sesión de jugador,

para todas las sesiones de jugador de una sesión de juego o para todas las sesiones de jugador asociadas a un solo ID de jugador.

Sintaxis

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest describePlayerSessionsRequest)
```

Parameters

[DescribePlayerSessionsRequest](#)

Un objeto [the section called “DescribePlayerSessionsRequest”](#) que describe las sesiones de jugador que recuperar.

Valor devuelto

Si funciona correctamente, devuelve un objeto [the section called “DescribePlayerSessionsOutcome”](#) que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud.

Ejemplo

Este ejemplo muestra una solicitud de todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Si se omite NextTokeny se establece el valor límite en 10, Amazon GameLift Servers se devolverán los registros de las sesiones de los primeros 10 jugadores que coincidan con la solicitud.

```
// 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()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan nuevos jugadores, Amazon GameLift Servers ofrece datos actualizados del emparejador mediante la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
startBackfillRequest);
```

Parameters

[StartMatchBackfillRequest](#)

Un objeto `StartMatchBackfillRequest` contiene información sobre la solicitud de reposición.

Valor devuelto

Devuelve un objeto [the section called “StartMatchBackfillOutcome”](#) con el ID del ticket de reposición de emparejamiento o un error con un mensaje de error.

Ejemplo

```
// 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()

Cancela una solicitud de reposición de emparejamiento activa. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Sintaxis

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

[StopMatchBackfillRequest](#)

Un objeto `StopMatchBackfillRequest` que proporciona detalles sobre el ticket de emparejamiento que va a detener.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Recupera la ruta al certificado TLS utilizado para cifrar la conexión de red entre el servidor de juegos y el cliente de juego. Puede utilizar la ruta del certificado al registrar el dispositivo informático en una flota de Amazon GameLift Servers Anywhere. Para obtener más información, consulte, [RegisterCompute](#).

Sintaxis

```
GetComputeCertificateOutcome GetComputeCertificate();
```

Valor devuelto

Devuelve un `GetComputeCertificateResponse` objeto que contiene lo siguiente:

- `CertificatePath`: La ruta al certificado TLS de su recurso informático. Cuando se utiliza una flota administrada por Amazon GameLift Servers, la ruta contiene los siguientes elementos:
 - `certificate.pem`: el certificado del usuario final. La cadena de certificados completa es la combinación del `certificateChain.pem` adjunto a este certificado.
 - `certificateChain.pem`: la cadena de certificados que contiene el certificado raíz y los certificados intermedios.
 - `rootCertificate.pem`: el certificado raíz.
 - `privateKey.pem`: la clave privada del certificado del usuario final.
- `ComputeName`: el nombre del recurso informático.

Ejemplo

```
GetComputeCertificateOutcome getComputeCertificateOutcome =  
    GameLiftServerAPI.GetComputeCertificate();
```

GetFleetRoleCredentials()

Recupera las credenciales del rol de IAM que autorizan a Amazon GameLift Servers a interactuar con otros Servicios de AWS. Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Sintaxis

```
GetFleetRoleCredentialsOutcome GetFleetRoleCredentials(GetFleetRoleCredentialsRequest  
    request);
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Credenciales de rol que amplían el acceso limitado a tus AWS recursos al servidor del juego.

Valor devuelto

Devuelve un objeto [the section called “GetFleetRoleCredentialsOutcome”](#).

Ejemplo

```
// form the fleet credentials request  
GetFleetRoleCredentialsRequest getFleetRoleCredentialsRequest = new  
    GetFleetRoleCredentialsRequest(){  
    RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"  
};  
GetFleetRoleCredentialsOutcome GetFleetRoleCredentialsOutcome credentials =  
    GetFleetRoleCredentials(getFleetRoleCredentialsRequest);
```

Destroy()

Libera de la memoria el SDK del servidor de juegos de Amazon GameLift Servers. Como práctica recomendada, llame a este método después de `ProcessEnding()` y antes de finalizar el proceso.

Si utiliza una flota de Anywhere y no va a finalizar los procesos del servidor después de cada sesión de juego, llame a `Destroy()` y, a continuación, a `InitSDK()` para reinicializar antes de informar a Amazon GameLift Servers de que el proceso está listo para organizar una sesión de juego con `ProcessReady()`.

Sintaxis

```
GenericOutcome Destroy()
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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 del servidor de Go para Amazon GameLift Servers: acciones

Utilice la referencia del SDK 5.x del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

`GameLiftServerAPI.go` define las acciones del SDK del servidor Go.

SDK del servidor de Go para Amazon GameLift Servers: tipos de datos

Utilice la referencia del SDK del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

GameLiftServerAPI.go define las acciones del SDK del servidor Go.

[SDK del servidor de Go para Amazon GameLift Servers: acciones](#)

Tipos de datos

- [LogParameters](#)
- [MetricsParameters](#)
- [ProcessParameters](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [ServerParameters](#)
- [StartMatchBackfillRequest](#)
- [Jugador](#)
- [DescribePlayerSessionsRequest](#)
- [StopMatchBackfillRequest](#)
- [GetFleetRoleCredentialsRequest](#)

LogParameters

Un objeto que identifica los archivos generados durante una sesión de juego que quiere que Amazon GameLift Servers cargue y almacene una vez finalizada la sesión de juego. El servidor de juegos proporciona LogParameters a Amazon GameLift Servers como parte de un objeto ProcessParameters en una llamada a [ProcessReady\(\)](#).

Propiedades	Descripción
LogPaths	Lista de las rutas de directorio a archivos de registro del servidor de juegos que desea que Amazon GameLift Servers almacene para futuros accesos. El proceso del servidor genera esos archivos durante una sesión de juego.

	Defina los nombres y las rutas de los archivos en el servidor de juegos y almacénelos en el directorio raíz de compilación del juego. Las rutas del registro deben ser absolutas. Por ejemplo, si la compilación del juego almacena los registros de sesión de juego en una ruta del tipo <code>MyGame\sessionLogs\</code>, la ruta sería <code>c:\game\MyGame\sessionLogs</code> en una instancia de Windows. Tipo: <code>[]string</code> Obligatorio: no

MetricsParameters

Objeto que comunica los parámetros de configuración para inicializar el sistema de métricas. Esta configuración se utiliza para configurar los informes de StatsD, los informes de bloqueo y el comportamiento de procesamiento de las métricas. El servidor de juegos proporciona `MetricsParameters` a Amazon GameLift Servers como parte de una llamada a [InitMetrics\(\)](#).

Propiedades	Descripción
StatsdHost	Host del servidor de StatsD para los informes de métricas (por ejemplo, "localhost"). Tipo: <code>string</code> Obligatorio: sí
StatsdPort	Puerto del servidor de StatsD para los informes de métricas (por ejemplo, 8125). Tipo: <code>int</code> Obligatorio: sí
CrashReporterHost	Host del servidor que notifica el bloqueo con fines de seguimiento y supervisión de procesos. Tipo: <code>string</code>

	Obligatorio: sí
CrashReporterPort	Puerto que notifica el bloqueo con fines de seguimiento y supervisión de procesos. Tipo: <code>int</code> Obligatorio: sí
FlushIntervalMs	Intervalo de vaciado de métricas en milisegundos. Controla la frecuencia con la que se envían las métricas a StatsD. Tipo: <code>int</code> Obligatorio: sí
MaxPacketSize	Tamaño máximo de paquete para las métricas, en bytes. Limita el tamaño de los paquetes UDP enviados a StatsD. Tipo: <code>int</code> Obligatorio: sí

ProcessParameters

Un objeto que describe la comunicación entre un proceso del servidor y Amazon GameLift Servers. El proceso del servidor proporciona esta información a Amazon GameLift Servers con una llamada a [ProcessReady\(\)](#).

Propiedades	Descripción
LogParameters	Un objeto con rutas de directorio a los archivos que se generan durante una sesión de juego. Amazon GameLift Servers copia y almacena los archivos para accesos futuros. Tipo: LogParameters Obligatorio: no

OnHealthCheck	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos y espera otros 60 segundos para obtener una respuesta. El proceso del servidor devuelve TRUE si está en buen estado y FALSE si no. Si no se devuelve ninguna respuesta, Amazon GameLift Servers registra el proceso del servidor como defectuoso. Tipo: <code>OnHealthCheck func() bool</code> Obligatorio: no
OnProcessTerminate	La función de devolución de llamada que invoca Amazon GameLift Servers para forzar el cierre del proceso de servidor. Después de llamar a esta función, Amazon GameLift Servers espera 5 minutos hasta que el proceso del servidor se cierre y responde con una llamada a ProcessEnding() antes de cerrar el proceso del servidor. Tipo: <code>OnProcessTerminate func()</code> Obligatorio: sí
OnStartGameSession	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de reposición de emparejamiento para proporcionar datos actualizados del emparejador. Pasa un objeto GameSession, una actualización de estado (<code>updateReason</code>) y el ID del ticket de reposición de emparejamiento. Tipo: <code>OnStartGameSession func (model.GameSession)</code> Obligatorio: sí

OnUpdateGameSession	Función de devolución de llamada que invoca Amazon GameLift Servers para pasar la información de sesión de juego actualizada al proceso del servidor. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de reposición de emparejamiento para proporcionar datos actualizados del emparejador. Tipo: <code>OnUpdateGameSession func (model.UpdateGameSession)</code> Obligatorio: no
Port	El número de puerto en el que escucha el proceso del servidor para recibir conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor. Tipo: <code>int</code> Obligatorio: sí

UpdateGameSession

Las actualizaciones en un objeto de sesión de juego, que incluye el motivo por el que se actualizó la sesión de juego y el ID del ticket de reposición correspondiente si la reposición se utiliza para reponer las sesiones de los jugadores en la sesión de juego.

Propiedades	Descripción
GameSession	Un objeto GameSession. El objeto <code>GameSession</code> contiene propiedades que describen una sesión de juego. Tipo: <code>GameSession GameSession()</code> Obligatorio: sí
UpdateReason	El motivo por el que se actualiza la sesión de juego.

Propiedades	Descripción
	Tipo: UpdateReason UpdateReason() Obligatorio: sí
BackfillTicketId	El ID de ticket de reposición que intenta actualizar la sesión de juego. Tipo: String Obligatorio: no

GameSession

Los detalles de una sesión de juego.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juego. Un nombre de recurso de Amazon (ARN) de sesión de juego tiene el siguiente formato: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Tipo: String Obligatorio: no
Name	Una etiqueta descriptiva de la sesión de juego. Tipo: String Obligatorio: no
FleetId	Un identificador único para la flota en la que se ejecuta la sesión de juego. Tipo: String Obligatorio: no

Propiedades	Descripción
MaximumPlayerSessionCount	El número máximo de conexiones de jugadores a la sesión de juego. Tipo: Integer Obligatorio: no
Puerto	El número de puerto de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, una aplicación necesita tanto la dirección IP como el número de puerto. Tipo: Integer Obligatorio: no
IpAddress	La dirección IP del servidor de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: String Obligatorio: no
GameSessionData	Un conjunto de propiedades de sesión de juego personalizadas, formateadas como un valor de una sola cadena. Tipo: String Obligatorio: no
MatchmakerData	La información sobre el proceso de emparejamiento que se utilizó para crear la sesión de juego, en sintaxis JSON, con formato como cadena. Además de la configuración de emparejamiento utilizada, contiene datos sobre todos los jugadores asignados al emparejamiento, incluidos los atributos de los jugadores y las asignaciones de los equipos. Tipo: String Obligatorio: no

Propiedades	Descripción
GameProperties	Un conjunto de propiedades personalizadas de una sesión de juego, con formato como pares clave-valor. Estas propiedades se pasan a una solicitud de iniciar una nueva sesión de juego. Tipo: <code>map[string] string</code> Obligatorio: no
DnsName	El identificador de DNS asignado a la instancia que ejecuta la sesión de juego. Los valores tienen formato siguiente: • Flotas habilitadas para TLS: <code><unique identifier>.<region identifier>.amazongamelift.com</code> . • Non-TLS-enabled flotas: <code>ec2-<unique identifier>.compute.amazonaws.com</code> . Cuando se conecte a una sesión de juego que se ejecute en una flota habilitada de TLS, debe utilizar el nombre de DNS, no la dirección IP. Tipo: <code>String</code> Obligatorio: no

ServerParameters

La información utilizada para mantener la conexión entre un servidor de Amazon GameLift Servers Anywhere y el servicio Amazon GameLift Servers. Esta información se utiliza al inicializar nuevos procesos de servidor con [InitSDK\(\)](#). Para los servidores alojados en instancias de EC2 administradas por Amazon GameLift Servers, utilice un objeto vacío.

Propiedades	Descripción
WebSocket URL	<code>GameLiftServerSdkEndpoint</code> que devuelve Amazon GameLift Servers cuando se invoca RegisterCompute para un recurso de computación de Amazon GameLift Servers Anywhere.

Propiedades	Descripción
	Tipo: <code>string</code> Obligatorio: sí
ProcessID	Un identificador único registrado en el proceso del servidor que aloja el juego. Tipo: <code>string</code> Obligatorio: sí
HostID	El identificador único del recurso informático que aloja el nuevo proceso del servidor. El HostID es el ComputeName utilizado cuando registró el recurso informático. Para obtener más información, consulte RegisterCompute . Tipo: <code>string</code> Obligatorio: sí
FleetID	El identificador único de la flota en la que está registrado el recurso informático. Para obtener más información, consulte RegisterCompute . Tipo: <code>string</code> Obligatorio: sí
AuthToken	El token de autenticación generado por Amazon GameLift Servers que autentica el servidor en Amazon GameLift Servers. Para obtener más información, consulte GetComputeAuthToken . Tipo: <code>string</code> Obligatorio: sí

StartMatchBackfillRequest

Información utilizada para crear una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información a Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	El identificador único de la sesión de juego. El GetGameSessionId de la operación de la API devuelve el identificador en formato de ARN. Tipo: String Obligatorio: sí
MatchmakingConfigurationArn	El identificador único, en forma de ARN, que el emparejador utiliza para esta solicitud. El ARN del emparejador para la sesión de juego original se encuentra en el objeto de sesión de juego en la propiedad de datos del emparejador. Para obtener más información sobre los datos del emparejador, consulte Trabajo con datos del emparejador. Tipo: String Obligatorio: sí
Players	Un conjunto de datos que representa a todos los jugadores que están actualmente en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Tipo: []model.Player Obligatorio: sí
TicketId	El identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no proporciona un valor, Amazon GameLift Servers generará uno. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario. Tipo: String Obligatorio: no

Jugador

El objeto que representa a un jugador en el emparejamiento. Al iniciar una solicitud de emparejamiento, un jugador tiene un ID de jugador, atributos y, posiblemente, datos de latencia. Amazon GameLift Servers añade la información de los equipos tras realizar un emparejamiento.

Propiedades	Descripción
LatencyInMS	Un conjunto de valores expresados en milisegundos que indican la cantidad de latencia que experimenta un jugador cuando se conecta a una ubicación. Si se utiliza esta propiedad, el jugador solo se empareja con las ubicaciones que aparecen. Si un creador de emparejamientos tiene una regla que evalúa la latencia de los jugadores, estos deben informar de la latencia para ser emparejados. Tipo: <code>map[string] int</code> Obligatorio: no
PlayerAttributes	Una colección de pares de clave-valor que contienen información del jugador para su uso en el emparejamiento. Las claves de atributos del jugador deben coincidir con las PlayerAttributes utilizadas en un conjunto de reglas de emparejamiento. Para obtener más información sobre los atributos de los jugadores, consulte AttributeValue. Tipo: <code>map[string] AttributeValue</code> Obligatorio: no
PlayerId	Un identificador único de un jugador. Tipo: <code>String</code> Obligatorio: no
Equipo	El nombre del equipo al que está asignado el jugador en un emparejamiento. Defina el nombre del equipo se define en el conjunto de reglas de emparejamiento.

Propiedades	Descripción
	Tipo: String
	Obligatorio: no

DescribePlayerSessionsRequest

Un objeto que especifica las sesiones de jugador que recuperar. El proceso del servidor proporciona esta información con una llamada [DescribePlayerSessions\(\)](#) a Amazon GameLift Servers.

Propiedades	Descripción
GameSessionID	Un identificador único de la sesión de juegos. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de sesión de juego es <code>arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string></code> . El GameSessionID es una cadena de ID personalizada o una cadena generada. Tipo: String Obligatorio: no
PlayerSessionID	El identificador único de una sesión de jugador. Utilice este parámetro para solicitar una sesión de jugador específica. Tipo: String Obligatorio: no
PlayerID	El identificador único de un jugador. Utilice este parámetro para solicitar todas las sesiones de jugador para un jugador específico. Consulte Generación de ID de jugador. Tipo: String Obligatorio: no

Propiedades	Descripción
<code>PlayerSessionStatusFilter</code>	El estado de la sesión de jugador para filtrar los resultados. Los posibles estados de sesión de jugador son los siguientes: • RESERVADO: se ha recibido la solicitud de sesión de jugador, pero el jugador aún no se ha conectado al proceso del servidor o aún no se ha validado. • ACTIVO: el proceso del servidor ha validado el jugador y está conectado. • COMPLETO: la conexión del jugador se ha caído. • TIEMPO DE ESPERA AGOTADO: se ha recibido una solicitud de sesión de jugador, pero el jugador no se ha conectado y/o no se ha validado en el plazo de tiempo de espera (60 segundos). Tipo: <code>String</code> Obligatorio: no
<code>NextToken</code>	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: <code>String</code> Obligatorio: no
<code>Limit</code>	El número máximo de resultados que devolver. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: <code>int</code> Obligatorio: no

StopMatchBackfillRequest

Información utilizada para cancelar una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	El identificador único de sesión de juego de la solicitud que se va a cancelar. Tipo: <code>string</code> Obligatorio: no
MatchmakingConfigurationArn	El identificador único del emparejador al que se envió esta solicitud. Tipo: <code>string</code> Obligatorio: no
TicketId	El identificador único del ticket de solicitud de reposición que se va a cancelar. Tipo: <code>string</code> Obligatorio: no

GetFleetRoleCredentialsRequest

Las credenciales de rol que amplían el acceso limitado a tus AWS recursos al servidor del juego. Para obtener más información, consulte, [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Propiedades	Descripción
RoleArn	El ARN del rol de servicio que amplía el acceso limitado a sus recursos de AWS . Tipo: <code>string</code> Obligatorio: sí
RoleSessionName	El nombre de la sesión que describe el uso de las credenciales del rol. Tipo: <code>string</code>

Propiedades	Descripción
	Obligatorio: sí

[SDK del servidor de Go para Amazon GameLift Servers: tipos de datos](#)

Acciones

- [GetSdkVersion\(\)](#)
- [InitMetrics\(\)](#)
- [InitMetricsFromEnvironment\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
func GetSdkVersion() (string, error)
```

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como una cadena. La cadena devuelta solo incluye el número de versión (por ejemplo, 5.0.0). Si no funciona, devuelve un mensaje de error, como `common.SdkVersionDetectionFailed`.

Ejemplo

```
version, err := server.GetSdkVersion()
```

InitMetrics()

Inicializa la recopilación de métricas para el SDK de Amazon GameLift Servers. Este método configura los informes de métricas para ayudar a supervisar el rendimiento y el estado de los servidores. Llame a este método después de [InitSDK\(\)](#), pero antes de [ProcessReady\(\)](#).

Sintaxis

```
func InitMetrics() error  
func InitMetrics(metricsParameters MetricsParameters) error
```

Parameters

MetricsParameters (opcional)

Objeto `MetricsParameters` que configura la recopilación de métricas. Si no se proporciona, se utiliza la configuración de métricas predeterminada. La `MetricsParameters` estructura contiene los siguientes campos:

- `StatsdHost`: el nombre de host o dirección IP del servidor de StatsD.
- `StatsdPort`: número de puerto del servidor de StatsD.
- `CrashReporterHost`: nombre de host o dirección IP del servidor que notifica el bloqueo.
- `CrashReporterPort`: número de puerto del servidor que notifica el bloqueo.
- `FlushIntervalMs`: intervalo en milisegundos para vaciar los datos de las métricas al servidor.
- `MaxPacketSize`: tamaño máximo de paquete para las métricas, en bytes.

Para obtener más información sobre la `MetricsParameters` estructura, consulte [Server SDK 5.x para los tipos de datos de C#](#).

Valor devuelto

Si se realiza correctamente, devuelve un error `nil` para indicar que la recopilación de métricas se ha inicializado correctamente.

Ejemplo

Inicialización de las métricas con la configuración predeterminada:

```
err := server.InitMetrics()
```

Inicialización de las métricas con una configuración personalizada:

```
metricsParams := MetricsParameters{
    StatsdHost:      "localhost",
    StatsdPort:      8125,
    CrashReporterHost: "localhost",
    CrashReporterPort: 9125,
    FlushIntervalMs:  5000,
    MaxPacketSize:    1024,
}

err := server.InitMetrics(metricsParams)
```

InitMetricsFromEnvironment()

Inicializa la recopilación de métricas para el SDK de Amazon GameLift Servers utilizando la configuración de las variables de entorno. Este método configura los informes de métricas utilizando la configuración predeterminada derivada del entorno de tiempo de ejecución.

Llame a este método después de [InitSDK\(\)](#), pero antes de [ProcessReady\(\)](#).

Sintaxis

```
func InitMetricsFromEnvironment() error
```

Valor devuelto

Si se realiza correctamente, devuelve un error `nil` para indicar que la recopilación de métricas se ha inicializado correctamente con la configuración del entorno.

Ejemplo

```
err := server.InitMetricsFromEnvironment()
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers. Llame a este método en el lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers. Este método establece la comunicación entre el servidor y el servicio Amazon GameLift Servers.

Sintaxis

```
func InitSDK(params ServerParameters) error
```

Parameters

ServerParameters

Para inicializar un servidor de juegos en una flota de Amazon GameLift Servers Anywhere, cree un objeto `ServerParameters` con la siguiente información:

- La URL WebSocket utilizada para conectarse al servidor de juegos.
- El ID del proceso utilizado para alojar su servidor de juegos.
- El ID del proceso utilizado para alojar los procesos del servidor de juegos.
- El ID de la flota de Amazon GameLift Servers que contiene su recurso de computación de Amazon GameLift Servers Anywhere.
- El token de autorización generado por la operación de Amazon GameLift Servers.

Para inicializar un servidor de juegos en una flota de EC2 administrada de Amazon GameLift Servers, cree un objeto `ServerParameters` sin parámetros. Con esta llamada, el agente de Amazon GameLift Servers configura el entorno informático y se conecta automáticamente al servicio Amazon GameLift Servers por usted.

Valor devuelto

Si funciona correctamente, devuelve el error `nil` para indicar que el proceso del servidor está listo para llamar a [ProcessReady\(\)](#).

 Note

Si las llamadas a `InitSDK()` no funcionan en las compilaciones de juegos implementadas en las flotas de Anywhere, compruebe el parámetro `ServerSdkVersion` que se utiliza al crear el recurso de compilación. Debe establecer este valor de forma explícita en la versión del SDK del servidor en uso. El valor predeterminado de este parámetro es 4.x, que no es compatible. Para resolver este problema, cree una compilación nueva e impleméntela en una flota nueva.

Ejemplo

Ejemplo de Amazon GameLift Servers Anywhere

```
//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)
```

Ejemplo de EC2 administrado por Amazon GameLift Servers

```
//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()

Notifica a Amazon GameLift Servers que el proceso del servidor de juegos está listo para alojar sesiones de juego. Llame a este método después de invocar [InitSDK\(\)](#). Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
func ProcessReady(param ProcessParameters) error
```

Parameters

ProcessParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información sobre el proceso del servidor:

- Nombres de los métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- El número de puerto de escucha del servidor de proceso.
- El tipo de datos [LogParameters](#) que contiene la ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Valor devuelto

Devuelve un error con un mensaje de error si el método falla. Devuelve `nil` si el método se realiza correctamente.

Ejemplo

Este ejemplo ilustra las implementaciones tanto de la función de llamada [ProcessReady\(\)](#) como de la función de delegación.

```
// 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()

Informa a Amazon GameLift Servers de que el proceso del servidor está finalizando. Llame a este método después de realizar todas las demás tareas de limpieza (lo que incluye el cierre de la sesión de juegos activa) y antes de finalizar el proceso. Según el resultado de `ProcessEnding()`, el proceso finaliza con éxito (0) o error (-1) y genera un evento de flota. Si el proceso finaliza con un error, el evento de flota generado será `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Sintaxis

```
func ProcessEnding() error
```

Valor devuelto

Devuelve un código de error 0 o un código de error definido.

Ejemplo

```
// 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()

Notifica a Amazon GameLift Servers que el proceso del servidor ha iniciado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Se llama a esta acción como parte de la

función de devolución de llamada `onStartGameSession()`, después de la inicialización de todas las sesiones de juego.

Sintaxis

```
func ActivateGameSession() error
```

Valor devuelto

Devuelve un error con un mensaje de error si el método falla.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de delegación `onStartGameSession()`.

```
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()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores.

Sintaxis

```
func UpdatePlayerSessionCreationPolicy(policy model.PlayerSessionCreationPolicy) error
```

Parameters

playerSessionCreationPolítica

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos.

Los valores válidos son:

- **model.AcceptAll**: se aceptan todas las sesiones de jugador nuevas.
- **model.DenyAll**: se rechazan todas las sesiones de jugador nuevas.

Valor devuelto

Devuelve un error con un mensaje de error si se produce un error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
err := server.UpdatePlayerSessionCreationPolicy(model.AcceptAll)
```

GetGameSessionId()

Recupera el ID de la sesión de juego alojada por el proceso del servidor.

Sintaxis

```
func GetGameSessionID() (string, error)
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego y el error nil. En el caso de los procesos inactivos que no se han activado aún con una sesión de juego, la llamada devuelve una cadena vacía y el error nil.

Ejemplo

```
gameSessionID, err := server.GetGameSessionID()
```

GetTerminationTime()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción después de recibir una devolución de llamada `onProcessTerminate()` desde Amazon GameLift Servers. Amazon GameLift Servers llama a `onProcessTerminate()` por los motivos siguientes:

- Cuando el proceso del servidor ha informado de un mal estado o no ha respondido a Amazon GameLift Servers.
- Al finalizar la instancia durante un evento de reducción vertical.
- [Cuando se finaliza una instancia debido a la interrupción de una instancia de spot.](#)

Sintaxis

```
func GetTerminationTime() (int64, error)
```

Valor devuelto

Si se ejecuta correctamente, devuelve la marca temporal en segundos en la que está previsto que el proceso del servidor se cierre y finalice el error `nil`. El valor es la hora de terminación expresado en ciclos transcurridos desde `0001 00:00:00`. Por ejemplo, el valor de la fecha y hora `2020-09-13 12:26:40 -0000Z` es igual a `637355968000000000` ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Ejemplo

```
terminationTime, err := server.GetTerminationTime()
```

AcceptPlayerSession()

Notifica a Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión de jugador es válido. Una vez validada la sesión del jugador, Amazon GameLift Servers cambia el estado del espacio de jugador de `RESERVED` a `ACTIVE`.

Sintaxis

```
func AcceptPlayerSession(playerSessionID string) error
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo gestiona una solicitud de conexión que incluye la validación y el rechazo de una sesión de jugador no válida. IDs

```
func ReceiveConnectingPlayerSessionID(conn Connection, playerSessionID string) {  
    err := server.AcceptPlayerSession(playerSessionID)  
    if err != nil {  
        connection.Accept()  
    } else {  
        connection.Reject(err.Error())  
    }  
}
```

RemovePlayerSession()

Informa a Amazon GameLift Servers de que un jugador se ha desconectado del proceso del servidor. Como respuesta, Amazon GameLift Servers cambia el espacio del jugador para que esté disponible.

Sintaxis

```
func RemovePlayerSession(playerSessionID string) error
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
err := server.RemovePlayerSession(playerSessionID)
```

DescribePlayerSessions()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice este método para obtener información sobre los siguientes elementos:

- Una sesión para un jugador
- Todas las sesiones del jugador en una sesión de juego
- Todas las sesiones de jugador están asociadas a un único ID de jugador

Sintaxis

```
func DescribePlayerSessions(req request.DescribePlayerSessionsRequest)
(result.DescribePlayerSessionsResult, error) {
    return srv.describePlayerSessions(&req)
}
```

Parameters

[DescribePlayerSessionsRequest](#)

Es un objeto `DescribePlayerSessionsRequest` que describe las sesiones de jugador a recuperar.

Valor devuelto

Si funciona correctamente, devuelve un objeto `DescribePlayerSessionsResult` que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud.

Ejemplo

En este ejemplo se solicitan todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Al omitir `NextToken` establecer el valor límite en 10, Amazon GameLift Servers devuelve los registros de las sesiones de los primeros 10 jugadores que coincidan con la solicitud.

```
// 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()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan nuevos jugadores, Amazon GameLift Servers ofrece datos actualizados del emparejador mediante la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
func StartMatchBackfill(req request.StartMatchBackfillRequest)
(result.StartMatchBackfillResult, error)
```

Parameters

[StartMatchBackfillRequest](#)

Un `StartMatchBackfillRequest` objeto comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers genera uno.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se encuentra en los datos del emparejador de la sesión de juego.
- El ID de la sesión de juego que se va a reponer.
- Datos del emparejador disponibles para los jugadores actuales de la sesión de juego.

Valor devuelto

Devuelve un objeto `StartMatchBackfillResult` con el ID del ticket de reposición de emparejamiento o un error con un mensaje de error.

Ejemplo

```
// 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()

Cancela una solicitud de reposición de emparejamiento activa. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Sintaxis

```
func StopMatchBackfill(req request.StopMatchBackfillRequest) error
```

Parameters

[StopMatchBackfillRequest](#)

Un `StopMatchBackfillRequest` objeto que identifica el billete de emparejamiento que se va a cancelar:

- ID del ticket que se asignará a la solicitud de reposición
- El emparejador al que se envió la solicitud de reposición
- La sesión de juego asociada a la solicitud de reposición.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
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()

Recupera la ruta al certificado TLS utilizado para cifrar la conexión de red entre el servidor de juegos y el cliente de juego. Puede utilizar la ruta del certificado al registrar el dispositivo informático en una flota de Amazon GameLift Servers Anywhere. Para obtener más información, consulte [RegisterCompute](#).

Sintaxis

```
func GetComputeCertificate() (result.GetComputeCertificateResult, error)
```

Valor devuelto

Devuelve un objeto `GetComputeCertificateResult` que contiene los siguientes elementos:

- `CertificatePath`: la ruta al certificado TLS de su recurso informático. Cuando se utiliza una flota administrada por Amazon GameLift Servers, la ruta contiene los siguientes elementos:
 - `certificate.pem`: el certificado del usuario final. La cadena de certificados completa es la combinación del `certificateChain.pem` adjunto a este certificado.

- `certificateChain.pem`: la cadena de certificados que contiene el certificado raíz y los certificados intermedios.
- `rootCertificate.pem`: el certificado raíz.
- `privateKey.pem`: la clave privada del certificado del usuario final.
- `ComputeName`: el nombre del recurso informático.

Ejemplo

```
tlsCertificate, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

GetFleetRoleCredentials()

Recupera las credenciales de la función de servicio que has creado para extender los permisos a la otra Servicios de AWS . Amazon GameLift Servers Estas credenciales permiten que su servidor de juegos utilice sus recursos de AWS . Para obtener más información, consulte [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Sintaxis

```
func GetFleetRoleCredentials(  
    req request.GetFleetRoleCredentialsRequest,  
) (result.GetFleetRoleCredentialsResult, error) {  
    return srv.getFleetRoleCredentials(&req)  
}
```

Parameters

[GetFleetRoleCredentialsRequest](#)

Credenciales de rol que amplían el acceso limitado a tus AWS recursos al servidor del juego.

Valor devuelto

Devuelve un objeto `GetFleetRoleCredentialsResult` que contiene los siguientes elementos:

- `AssumedRoleUserArn` - El nombre de recurso de Amazon (ARN) del usuario al que pertenece la función de servicio.
- `AssumedRoleId` - El ID del usuario al que pertenece el rol de servicio.

- **AccessKeyId** - El ID de la clave de acceso para autenticar y proporcionar acceso a sus AWS recursos.
- **SecretAccessKey** - El identificador de la clave de acceso secreta para la autenticación.
- **SessionToken** - Un token para identificar la sesión activa actual que interactúa con tus AWS recursos.
- **Vencimiento**: el tiempo que queda hasta que caduquen las credenciales de la sesión.

Ejemplo

```
// form the customer credentials request
getFleetRoleCredentialsRequest := request.NewGetFleetRoleCredentials()
getFleetRoleCredentialsRequest.RoleArn = "arn:aws:iam::123456789012:role/service-role/exampleGameLiftAction"

credentials, err := server.GetFleetRoleCredentials(getFleetRoleCredentialsRequest)
```

Destroy()

Libera de la memoria el SDK del servidor de juegos de Amazon GameLift Servers. Como práctica recomendada, llame a este método después de `ProcessEnding()` y antes de finalizar el proceso. Si utiliza una flota de Anywhere y no va a finalizar los procesos del servidor después de cada sesión de juego, llame a `Destroy()` y, a continuación, a `InitSDK()` para reinicializar antes de informar a Amazon GameLift Servers de que el proceso está listo para organizar una sesión de juego con `ProcessReady()`.

Sintaxis

```
func Destroy() error {
    return srv.destroy()
}
```

Valor devuelto

Devuelve un error con un mensaje de error si el método falla.

Ejemplo

```
// 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 del servidor de C++ (Unreal) para Amazon GameLift Servers: acciones

Utilice esta referencia del SDK 5.x del servidor de Unreal de Amazon GameLift Servers como ayuda para preparar un juego multijugador para el uso con Amazon GameLift Servers. Para obtener más información sobre el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#). Si utiliza el complemento de Amazon GameLift Servers para Unreal, consulte también [Complemento de Amazon GameLift Servers para Unreal Engine](#).

Note

En este tema se describe la API de C++ de Amazon GameLift Servers que puede utilizar al realizar la compilación para Unreal Engine. En concreto, esta documentación se aplica al código que se compila con la opción `-DBUILD_FOR_UNREAL=1`.

SDK del servidor de C++ (Unreal) 5.x para Amazon GameLift Servers: tipos de datos

Utilice esta referencia del SDK 5.x del servidor de Unreal de Amazon GameLift Servers como ayuda para preparar un juego multijugador para el uso con Amazon GameLift Servers. Para obtener más información sobre el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#). Si utiliza el complemento de Amazon GameLift Servers para Unreal, consulte también [Complemento de Amazon GameLift Servers para Unreal Engine](#).

Note

En este tema se describe la API de C++ de Amazon GameLift Servers que puede utilizar al realizar la compilación para Unreal Engine. En concreto, esta documentación se aplica al código que se compila con la opción `-DBUILD_FOR_UNREAL=1`.

[SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)

Tipos de datos

- [FProcessParámetros](#)
- [UpdateGameSession](#)
- [GameSession](#)
- [FServerParámetros](#)
- [FStartMatchBackfillRequest](#)
- [FPlayer](#)
- [FGameLiftDescribePlayerSessionsRequest](#)
- [FStopMatchBackfillRequest](#)
- [FAttributeValor](#)
- [FGameLiftGetFleetRoleCredentialsRequest](#)
- [FGameLiftLongOutcome](#)
- [FGameLiftStringOutcome](#)
- [FGameLiftDescribePlayerSessionsOutcome](#)
- [FGameLiftDescribePlayerSessionsResult](#)
- [FGenericResultado](#)
- [FGameLiftPlayerSession](#)
- [FGameLiftGetComputeCertificateOutcome](#)
- [FGameLiftGetComputeCertificateResult](#)
- [FGameLiftGetFleetRoleCredentialsOutcome](#)
- [FGetFleetRoleCredentialsResult](#)
- [FGameLiftError](#)
- [Enums](#)

FProcessParámetros

Este tipo de datos contiene el conjunto de parámetros enviado a Amazon GameLift Servers en una [ProcessReady\(\)](#).

Propiedades	Descripción

LogParameters	Un objeto con rutas de directorio a los archivos que se generan durante una sesión de juego. Amazon GameLift Servers copia y almacena los archivos para accesos futuros. Tipo: TArray<FString> Obligatorio: no
OnHealthCheck	Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos y espera otros 60 segundos para obtener una respuesta. El proceso del servidor devuelve TRUE si está en buen estado y FALSE si no. Si no se devuelve ninguna respuesta, Amazon GameLift Servers registra el proceso del servidor como defectuoso. Esta propiedad es una función de delegación definida como DECLARE_DELEGATE_RetVal(bool, FOnHealthCheck) ; Tipo: FOnHealthCheck Obligatorio: no

OnProcessTerminate	La función de devolución de llamada que invoca Amazon GameLift Servers para forzar el cierre del proceso de servidor. Después de llamar a esta función, Amazon GameLift Servers espera 5 minutos hasta que el proceso del servidor se cierre y responde con una llamada a ProcessEnding() antes de cerrar el proceso del servidor. Tipo: FSimpleDelegate Obligatorio: sí
OnStartGameSession	La función de devolución de llamada que se Amazon GameLift Servers invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a una solicitud de un cliente. CreateGameSession La función de devolución de llamada pasa un objeto GameSession. Esta propiedad es una función de delegación definida como DECLARE_DELEGATE_OneParam(FOnStartGameSession, Aws::GameLift::Server::Model::GameSession); Tipo: FOnStartGameSession Obligatorio: sí

OnUpdateGameSession

Nombre de la función de devolución de llamada que invoca Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de reposición de emparejamiento para proporcionar datos actualizados del emparejador. Pasa un objeto [GameSession](#), una actualización de estado (`updateReason`) y el ID del ticket de reposición de emparejamiento.

Esta propiedad es una función de delegación definida como `DECLARE_DELEGATE_OneParam(FOnUpdateGameSession, Aws::GameLift::Server::Model::UpdateGameSession);` ;

Tipo: `FOnUpdateGameSession`

Obligatorio: no

Puerto

El número de puerto en el que escucha el proceso del servidor para recibir conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: `int`

Obligatorio: sí

UpdateGameSession

Este tipo de datos se actualiza a un objeto de sesión de juego, que incluye el motivo por el que se actualizó la sesión de juego y el ID del ticket de reposición correspondiente si la reposición se utiliza para reponer las sesiones de los jugadores en la sesión de juego.

Propiedades	Descripción
GameSession	Un objeto GameSession. El objeto <code>GameSession</code> contiene propiedades que describen una sesión de juego. Tipo: <code>Aws::GameLift::Server::GameSession</code> Obligatorio: no
UpdateReason	El motivo por el que se actualiza la sesión de juego. Tipo: <code>enum class UpdateReason</code> • <code>MATCHMAKING_DATA_UPDATED</code> • <code>BACKFILL_FAILED</code> • <code>BACKFILL_TIMED_OUT</code> • <code>BACKFILL_CANCELLED</code> Obligatorio: no
BackfillTicketId	El ID de ticket de reposición que intenta actualizar la sesión de juego. Tipo: <code>char[]</code> Obligatorio: no

GameSession

Este tipo de datos proporciona detalles de una sesión de juego.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juego. El ARN de una sesión de juego tiene el siguiente formato: <code>arn:aws:gamelift:<region>::gamesession/<fleet ID>/<custom ID string or idempotency token></code> . Tipo: <code>char[]</code> Obligatorio: no
Name	Una etiqueta descriptiva de la sesión de juego. Tipo: <code>char[]</code> Obligatorio: no
FleetId	Un identificador único para la flota en la que se ejecuta la sesión de juego. Tipo: <code>char[]</code> Obligatorio: no
MaximumPlayerSessionCount	El número máximo de conexiones de jugadores a la sesión de juego. Tipo: <code>int</code> Obligatorio: no
Puerto	El número de puerto de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: <code>int</code>

Propiedades	Descripción
	Obligatorio: no
IpAddress	La dirección IP del servidor de la sesión de juego. Para conectarse a un servidor de Amazon GameLift Servers juegos, la aplicación necesita tanto la dirección IP como el número de puerto. Tipo: <code>char[]</code> Obligatorio: no
GameSessionData	Un conjunto de propiedades de sesión de juego personalizadas, formateadas como un valor de una sola cadena. Tipo: <code>char[]</code> Obligatorio: no
MatchmakerData	Información sobre el proceso de emparejamiento que se utilizó para crear la sesión de juego, en sintaxis JSON, con formato como cadena. Además de la configuración de emparejamiento utilizada, contiene datos sobre todos los jugadores asignados al emparejamiento, incluidos los atributos de los jugadores y las asignaciones de los equipos. Tipo: <code>char[]</code> Obligatorio: no

Propiedades	Descripción
GameProperties	Un conjunto de propiedades personalizadas de una sesión de juego, con formato como pares clave-valor. Estas propiedades se pasan a una solicitud de iniciar una nueva sesión de juego. Tipo: <code>GameProperty[]</code> Obligatorio: no
DnsName	El identificador de DNS asignado a la instancia que ejecuta la sesión de juego. Los valores tienen formato siguiente: - Flotas habilitadas para TLS: <code><unique identifier>.<region identifier>.amazongamelift.com</code> . - Non-TLS-enabled flotas: <code>ec2-<unique identifier>.compute.amazonaws.com</code> . Cuando se conecte a una sesión de juego que se ejecute en una flota habilitada de TLS, debe utilizar el nombre de DNS, no la dirección IP. Tipo: <code>char[]</code> Obligatorio: no

FServerParámetros

La información utilizada para mantener la conexión entre un servidor de Amazon GameLift Servers Anywhere y el servicio Amazon GameLift Servers. Esta información se utiliza al inicializar nuevos procesos de servidor con [InitSDK\(\)](#). Para los servidores alojados en instancias de EC2 administradas por Amazon GameLift Servers, utilice un objeto vacío.

Propiedades	Descripción
webSocketUrl	El <code>GameLiftServerSdkEndpoint</code> que devuelve Amazon GameLift Servers cuando se invoca RegisterCompute para un recurso de computación de Amazon GameLift Servers Anywhere. Tipo: <code>char[]</code> Obligatorio: sí
processId	Un identificador único registrado en el proceso de servidor que aloja el juego. Tipo: <code>char[]</code> Obligatorio: sí
hostId	El <code>HostID</code> es el <code>ComputeName</code> utilizado cuando registró el recurso informático. Para obtener más información, consulte, RegisterCompute. Tipo: <code>char[]</code> Obligatorio: sí
fleetId	El identificador único de la flota en la que está registrado el recurso informático. Para obtener más información, consulte, RegisterCompute. Tipo: <code>char[]</code> Obligatorio: sí
authToken	El token de autenticación generado por Amazon GameLift Servers que autentica el servidor en Amazon GameLift Servers. Para

Propiedades	Descripción
	obtener más información, consulte, GetComputeAuthToken .
	Tipo: char[]
	Obligatorio: sí

FStartMatchBackfillRequest

Información utilizada para crear una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información a Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	Un identificador único de la sesión de juegos. El GetGameSessionId de la operación de la API devuelve el identificador en formato de ARN. Tipo: char[] Obligatorio: sí
MatchmakingConfigurationArn	Un identificador único, con formato de ARN, que el emparejador utiliza para esta solicitud. El ARN del emparejador para la sesión de juego original se encuentra en el objeto de sesión de juego en la propiedad de datos del emparejador. Puede obtener más información sobre los datos del emparejador en Trabajo con datos del emparejador . Tipo: char[] Obligatorio: sí

Propiedades	Descripción
Players	Un conjunto de datos que representa a todos los jugadores que están en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Tipo: TArray<FPlayer> Obligatorio: sí
TicketId	Un identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no proporciona un valor, Amazon GameLift Servers generará uno. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario. Tipo: char[] Obligatorio: no

FPlayer

Este tipo de datos representa a un jugador en el sistema de emparejamiento. Al iniciar una solicitud de emparejamiento, un jugador tiene un ID de jugador, atributos y, posiblemente, datos de latencia. Amazon GameLift Servers añade la información de los equipos tras realizar un emparejamiento.

Propiedades	Descripción
LatencyInMS	Un conjunto de valores expresados en milisegundos que indican la cantidad de latencia que experimenta un jugador cuando se conecta a una ubicación.

Propiedades	Descripción
	Si se utiliza esta propiedad, el jugador solo se empareja con las ubicaciones que aparecen. Si un creador de emparejamientos tiene una regla que evalúa la latencia de los jugadores , estos deben informar de la latencia para ser emparejados. Tipo: <code>TMap>FString, int32<</code> Obligatorio: no
PlayerAttributes	Una colección de pares de clave-valor que contienen información del jugador para su uso en el emparejamiento. Las claves de atributos del jugador deben coincidir con las PlayerAttributes utilizadas en un conjunto de reglas de emparejamiento. Para obtener más información sobre los atributos de los jugadores, consulte Attribute Value. Tipo: <code>TMap>FString, FAttributeValue<</code> Obligatorio: no
PlayerId	Un identificador único de un jugador. Tipo: <code>std::string</code> Obligatorio: no

Propiedades	Descripción
Equipo	El nombre del equipo al que está asignado el jugador en un emparejamiento. Defina el nombre del equipo se define en el conjunto de reglas de emparejamiento. Tipo: FString Obligatorio: no

FGameLiftDescribePlayerSessionsRequest

Un objeto que especifica las sesiones de jugador que recuperar. El proceso del servidor proporciona esta información con una llamada [DescribePlayerSessions\(\)](#) a Amazon GameLift Servers.

Propiedades	Descripción
GameSessionId	Un identificador único de la sesión de juegos. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de sesión de juego es FString. El GameSessionID es una cadena de ID personalizada o una Tipo: std::string Obligatorio: no
PlayerSessionId	El identificador único de una sesión de jugador. Utilice este parámetro para solicitar una sesión de jugador específica. Tipo: FString Obligatorio: no

Propiedades	Descripción
PlayerId	El identificador único de un jugador. Utilice este parámetro para solicitar todas las sesiones de jugador para un jugador específico. Consulte Generación de ID de jugador. Tipo: FString Obligatorio: no
PlayerSessionStatusFilter	El estado de la sesión de jugador para filtrar los resultados. Los posibles estados de sesión de jugador son los siguientes: • RESERVADO: se ha recibido la solicitud de sesión de jugador, pero el jugador aún no se ha conectado al proceso del servidor o aún no se ha validado. • ACTIVO: el proceso del servidor ha validado el jugador y está conectado. • COMPLETO: la conexión del jugador se ha caído. • TIEMPO DE ESPERA AGOTADO: se ha recibido una solicitud de sesión de jugador, pero el jugador no se ha conectado y/o no se ha validado en el plazo de tiempo de espera (60 segundos). Tipo: FString Obligatorio: no

Propiedades	Descripción
NextToken	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: FString Obligatorio: no
Límite	El número máximo de resultados que devolver. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: int Obligatorio: no

FStopMatchBackfillRequest

Información utilizada para cancelar una solicitud de reposición de emparejamiento. El servidor de juegos comunica esa información al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Propiedades	Descripción
GameSessionArn	Un identificador único de sesión de juego de la solicitud que se va a cancelar. Tipo: FString Obligatorio: sí
MatchmakingConfigurationArn	Un identificador único del emparejador al que se envió esta solicitud. Tipo: FString

Propiedades	Descripción
	Obligatorio: sí
TicketId	Un identificador único del ticket de solicitud de reposición que se va a cancelar. Tipo: FString Obligatorio: sí

FAttributeValor

Utilice estos valores en pares de clave-valor de atributo [FPlayer](#). Este objeto le permite especificar un valor de atributo mediante cualquiera de los tipos de datos válidos: cadena, número, matriz de cadenas o mapa de datos. Cada objeto `AttributeValue` puede utilizar solo una de las propiedades disponibles.

Propiedades	Description (Descripción)
attrType	Especifica el tipo de valor del atributo. Tipo: un valor de enum de <code>FAttributeType</code> . Obligatorio: no
S	Representa un valor de atributo de cadena. Tipo: FString Obligatorio: no
N	Representa un valor de atributo numérico. Tipo: double Obligatorio: no
SL	Representa una matriz de valores de atributos de cadena. Tipo: TArray<FString>

Propiedades	Description (Descripción)
	Obligatorio: no
SDM	Representa un diccionario de claves de cadena y valores dobles. Tipo: TMap<FString, double> Obligatorio: no

FGameLiftGetFleetRoleCredentialsRequest

Este tipo de datos proporciona credenciales de rol que amplían el acceso limitado a sus recursos de AWS al servidor de juegos. Para obtener más información, consulte, [Configuración de un rol de servicio de IAM para Amazon GameLift Servers](#).

Propiedades	Descripción
RoleArn	El nombre de recurso de Amazon (ARN) de la función de servicio que amplía el acceso limitado a sus AWS recursos. Tipo: FString Obligatorio: no
RoleSessionName	El nombre de la sesión que describe el uso de las credenciales del rol. Tipo: FString Obligatorio: no

FGameLiftLongOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: long Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: long&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGameLiftStringOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: FString Obligatorio: no

Propiedades	Description (Descripción)
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: FString&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGameLiftDescribePlayerSessionsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “FGameLiftDescribePlayerSessionsResult” Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: FGameLiftDescribePlayerSessionsResult&&

Propiedades	Description (Descripción)
	Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGameLiftDescribePlayerSessionsResult

Propiedades	Description (Descripción)
PlayerSessions	Tipo: TArray<FGameLiftPlayerSession> Obligatorio: sí
NextToken	El token que indica el inicio de la siguiente página de resultados. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora. Tipo: FString Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí

Propiedades	Description (Descripción)
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGenericResultado

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGameLiftPlayerSession

Propiedades	Description (Descripción)
CreationTime	Tipo: long Obligatorio: sí
FleetId	Tipo: FString Obligatorio: sí

Propiedades	Description (Descripción)
GameSessionId	Tipo: FString Obligatorio: sí
IpAddress	Tipo: FString Obligatorio: sí
PlayerData	Tipo: FString Obligatorio: sí
PlayerId	Tipo: FString Obligatorio: sí
PlayerSessionId	Tipo: FString Obligatorio: sí
Puerto	Tipo: int Obligatorio: sí
Status	Tipo: una enumeración de PlayerSessionStatus . Obligatorio: sí
TerminationTime	Tipo: long Obligatorio: sí
DnsName	Tipo: FString Obligatorio: sí

FGameLiftGetComputeCertificateOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “FGameLiftGetComputeCertificateResult” Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: FGameLiftGetComputeCertificateResult&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGameLiftGetComputeCertificateResult

La ruta al certificado TLS de su recurso informático y el nombre de host del equipo.

Propiedades	Description (Descripción)
CertificatePath	Tipo: FString Obligatorio: sí
ComputeName	Tipo: FString

Propiedades	Description (Descripción)
	Obligatorio: sí

FGameLiftGetFleetRoleCredentialsOutcome

Este tipo de datos es el resultado de una acción y produce un objeto con las siguientes propiedades:

Propiedades	Description (Descripción)
Resultado	El resultado de la acción. Tipo: the section called “FGetFleetRoleCredentialsResult” Obligatorio: no
ResultWithOwnership	El resultado de la acción, expresado como un valor r, para que el código de llamada pueda tomar posesión del objeto. Tipo: FGameLiftGetFleetRoleCredentialsResult&& Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: bool Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “FGameLiftError” Obligatorio: no

FGetFleetRoleCredentialsResult

Propiedades	Description (Descripción)
AccessKeyId	El ID de la clave de acceso para autenticar y proporcionar acceso a los recursos de AWS . Tipo: FString Obligatorio: no
AssumedRoleId	El ID del usuario al que pertenece el rol de servicio. Tipo: FString Obligatorio: no
AssumedRoleUserArn	El nombre de recurso de Amazon (ARN) del usuario al que pertenece el rol de servicio. Tipo: FString Obligatorio: no
Expiration	El tiempo que queda hasta que caduquen las credenciales de la sesión. Tipo: FDateTime Obligatorio: no
SecretAccessKey	El ID de clave de acceso secreta para la autenticación. Tipo: FString Obligatorio: no
SessionToken	Un token para identificar la sesión activa actual que interactúa con tus AWS recursos. Tipo: FString

Propiedades	Description (Descripción)
	Obligatorio: no
Success	Si la acción se realizó correctamente o no. Tipo: <code>bool</code> Obligatorio: sí
Error	El error que se genera si la acción no se ha realizado correctamente. Tipo: the section called “GameLiftError” Obligatorio: no

FGameLiftError

Propiedades	Description (Descripción)
ErrorType	Tipo de error. Tipo: una enumeración de <code>GameLiftErrorType</code> . Obligatorio: no
ErrorMessage	El nombre del error. Tipo: <code>std::string</code> Obligatorio: no
ErrorName	Mensaje de error. Tipo: <code>std::string</code> Obligatorio: no

Enums

Las enumeraciones definidas para el SDK del servidor de Amazon GameLift Servers (Unreal) se definen de la siguiente manera:

FAttributeEscriba

- NONE
- STRING
- DOUBLE
- STRING_LIST
- STRING_DOUBLE_MAP

GameLiftErrorType

Valor de cadena que indica el tipo de error. Los valores válidos son:

- SERVICE_CALL_FAILED: no se pudo realizar una llamada a un AWS servicio.
- LOCAL_CONNECTION_FAILED: se ha producido un error en la conexión local a Amazon GameLift Servers.
- NETWORK_NOT_INITIALIZED: la red no se ha inicializado.
- GAMESESSION_ID_NOT_SET: no se ha establecido el ID de sesión de juego.
- BAD_REQUEST_EXCEPTION
- INTERNAL_SERVICE_EXCEPTION
- ALREADY_INITIALIZED: el servidor o cliente de Amazon GameLift Servers ya se inicializó con Initialize().
- FLEET_MISMATCH: la flota de destino no coincide con la flota de gameSession o playerSession.
- GAMESLIFT_CLIENT_NOT_INITIALIZED: el cliente de Amazon GameLift Servers no se ha inicializado.
- GAMESLIFT_SERVER_NOT_INITIALIZED: el servidor de Amazon GameLift Servers no se ha inicializado.
- GAME_SESSION_ENDED_FAILED: el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego había finalizado.
- GAME_SESSION_NOT_READY: no se activó la sesión de juego del servidor de Amazon GameLift Servers.

- **GAME_SESSION_READY_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo ponerse en contacto con el servicio para informar de que la sesión de juego estaba lista.
- **INITIALIZATION_MISMATCH:** se llamó a un método de cliente después de `Server::Initialize()` o viceversa.
- **NOT_INITIALIZED:** el servidor o el cliente de Amazon GameLift Servers no se han iniciado con `Initialize()`.
- **NO_TARGET_ALIASID_SET:** no se ha establecido un `aliasId` de destino.
- **NO_TARGET_FLEET_SET:** no se ha establecido una flota de destino.
- **PROCESS_ENDING_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está finalizando.
- **PROCESS_NOT_ACTIVE** — El proceso del servidor aún no está activo, no está vinculado a un y no puede aceptarlo ni procesarlo. `GameSession PlayerSessions`
- **PROCESS_NOT_READY:** el proceso de servidor aún no está listo para activarse.
- **PROCESS_READY_FAILED:** el SDK del servidor de Amazon GameLift Servers no pudo contactar con el servicio para informar de que el proceso está listo.
- **SDK_VERSION_DETECTION_FAILED:** no se pudo detectar la versión del SDK.
- **STX_CALL_FAILED:** no se pudo realizar una llamada al componente de backend del servidor. `XStx`
- **STX_INITIALIZATION_FAILED:** el componente de backend del servidor no se pudo inicializar. `XStx`
- **UNEXPECTED_PLAYER_SESSION:** el servidor ha detectado una sesión de jugador no registrada.
- **WEBSOCKET_CONNECT_FAILURE**
- **WEBSOCKET_CONNECT_FAILURE_FORBIDDEN**
- **WEBSOCKET_CONNECT_FAILURE_INVALID_URL**
- **WEBSOCKET_CONNECT_FAILURE_TIMEOUT**
- **WEBSOCKET_RETRIABLE_SEND_MESSAGE_FAILURE:** error recuperable al enviar un mensaje al servicio. `GameLift WebSocket`
- **WEBSOCKET_SEND_MESSAGE_FAILURE:** error al enviar un mensaje al GameLift servicio. `WebSocket`
- **MATCH_BACKFILL_REQUEST_VALIDATION:** se ha producido un error en la validación de la solicitud.

- `PLAYER_SESSION_REQUEST_VALIDATION`: se ha producido un error en la validación de la solicitud.

EPlayerSessionCreationPolicy

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos. Los valores válidos son:

- `ACCEPT_ALL`: se aceptan todas las sesiones de jugador nuevas.
- `DENY_ALL`: se rechazan todas las sesiones de jugador nuevas.
- `NOT_SET`: la sesión de juego no está configurada para aceptar o denegar sesiones de nuevos jugadores.

EPlayerSessionStatus

- `ACTIVE`
- `COMPLETED`
- `NOT_SET`
- `RESERVED`
- `TIMEDOUT`

[SDK del servidor de C++ \(Unreal\) 5.x para Amazon GameLift Servers: tipos de datos](#)

Temas

- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [InitSDK\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessEnding\(\)](#)
- [ActivateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetTerminationTime\(\)](#)
- [AcceptPlayerSession\(\)](#)
- [RemovePlayerSession\(\)](#)

- [DescribePlayerSessions\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [GetComputeCertificate\(\)](#)
- [GetFleetRoleCredentials\(\)](#)
- [Destroy\(\)](#)

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
FGameLiftStringOutcome GetSdkVersion();
```

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto [the section called "FGameLiftStringOutcome"](#). El objeto devuelto incluye el número de versión (por ejemplo, 5.0.0). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers para una flota de EC2 administrada. Llame a este método en el lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers. Este método lee parámetros de servidor del entorno de host para configurar la comunicación entre el servidor y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Sintaxis

```
FGameLiftGenericOutcome InitSDK()
```


Valor devuelto

Si funciona correctamente, devuelve un objeto `InitSdkOutcome` e indica que el proceso del servidor está listo para llamar a [ProcessReady\(\)](#).

Ejemplo

```
//Call InitSDK to establish a local connection with the Amazon GameLift Servers Agent
to enable further communication.
FGameLiftGenericOutcome initSdkOutcome = GameLiftSdkModule->InitSDK();
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers para una flota de Anywhere o una flota de contenedores administrada. Llame a este método en el lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers. Este método requiere que los parámetros del servidor explícito configure la comunicación entre el servidor y el servicio Amazon GameLift Servers. Utiliza un token de idempotencia, por lo que puede volver a intentar esta llamada de forma segura cuando falle.

Sintaxis

```
FGameLiftGenericOutcome InitSDK(serverParameters)
```

Parameters

[FServerParámetros](#)

Para inicializar un servidor de juegos en una flota de Amazon GameLift Servers Anywhere, cree un objeto `ServerParameters` con la siguiente información:

- La URL WebSocket utilizada para conectarse a tu servidor de juegos.
- El ID del proceso utilizado para alojar su servidor de juegos.
- El ID del proceso utilizado para alojar los procesos del servidor de juegos.
- El ID de la flota de Amazon GameLift Servers que contiene su recurso de computación de Amazon GameLift Servers Anywhere.
- El token de autorización generado por la operación de Amazon GameLift Servers.

Valor devuelto

Si funciona correctamente, devuelve un objeto `InitSdkOutcome` e indica que el proceso del servidor está listo para llamar a [ProcessReady\(\)](#).

Note

Si las llamadas a `InitSDK()` no funcionan en las compilaciones de juegos implementadas en las flotas de Anywhere, compruebe el parámetro `ServerSdkVersion` que se utiliza al crear el recurso de compilación. Debe establecer este valor de forma explícita en la versión del SDK del servidor en uso. El valor predeterminado de este parámetro es 4.x, que no es compatible. Para resolver este problema, cree una compilación nueva e impleméntela en una flota nueva.

Ejemplo

```
//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()

Notifica a Amazon GameLift Servers que el proceso del servidor de juegos está listo para alojar sesiones de juego. Llame a este método después de invocar [InitSDK\(\)](#). Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
GenericOutcome ProcessReady(const Aws::GameLift::Server::ProcessParameters
&processParameters);
```

Parameters

processParameters

Un objeto [FProcessParámetros](#) que comunica la siguiente información sobre el proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra las implementaciones tanto de la función de llamada [ProcessReady\(\)](#) como de la función de delegación.

```
//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()

Informa a Amazon GameLift Servers de que el proceso del servidor está finalizando. Llame a este método después de realizar todas las demás tareas de limpieza (lo que incluye el cierre de la sesión de juegos activa) y antes de finalizar el proceso. Según el resultado de `ProcessEnding()`, el proceso finaliza con éxito (0) o error (-1) y genera un evento de flota. Si el proceso termina con un error, el evento de flota generado es `SERVER_PROCESS_TERMINATED_UNHEALTHY`.

Sintaxis

```
FGameLiftGenericOutcome ProcessEnding()
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
//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()

Notifica a Amazon GameLift Servers que el proceso del servidor ha iniciado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de la inicialización de todas las sesiones de juego.

Sintaxis

```
FGameLiftGenericOutcome ActivateGameSession()
```

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de delegación `onStartGameSession()`.

```
//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();
};
```

UpdatePlayerSessionCreationPolicy()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores.

Sintaxis

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy
policy)
```

Parameters

playerCreationSessionPolítica

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos.

Los valores válidos son:

- ACCEPT_ALL: se aceptan todas las sesiones de jugador nuevas.
- DENY_ALL: se rechazan todas las sesiones de jugador nuevas.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
FGameLiftGenericOutcome outcome = GameLiftSdkModule-
>UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::EPlayerSessionCreationPolicy::ACCEPT_A
```

GetGameSessionId()

Recupera el ID de la sesión de juego alojada por el proceso del servidor.

En el caso de los procesos inactivos que no se activan con una sesión de juego, la llamada devuelve [the section called "FGameLiftError"](#).

Sintaxis

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto [the section called "FGameLiftStringOutcome"](#). Si no funciona, devuelve un mensaje de error.

En el caso de los procesos inactivos que no se activan con una sesión de juego, la llamada devuelve `Success=True` y `GameSessionId=""`.

Ejemplo

```
//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()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción después de recibir una

devolución de llamada `onProcessTerminate()` desde Amazon GameLift Servers. Amazon GameLift Servers llama a `onProcessTerminate()` por los motivos siguientes:

- Cuando el proceso del servidor informa de un mal estado o no responde a Amazon GameLift Servers.
- Al finalizar la instancia durante un evento de reducción vertical.
- [Cuando se finaliza una instancia debido a la interrupción de una instancia de spot.](#)

Sintaxis

```
AwsDateTimeOutcome GetTerminationTime()
```

Valor devuelto

Si el proceso se realiza correctamente, devuelve la hora de terminación como un objeto `AwsDateTimeOutcome`. El valor es la hora de terminación expresado en ciclos transcurridos desde 0001 00:00:00. Por ejemplo, el valor de la fecha y hora 2020-09-13 12:26:40 -000Z es igual a 6373559680000000000 ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Si el proceso no ha recibido ninguna devolución de llamada `ProcessParameters.OnProcessTerminate()`, se devuelve un mensaje de error. Para obtener más información sobre cómo cerrar un proceso de servidor, consulte [Respuesta a una notificación de cierre del proceso del servidor](#).

Ejemplo

```
AwsDateTimeOutcome TermTimeOutcome = GameLiftSdkModule->GetTerminationTime();
```

AcceptPlayerSession()

Notifica a Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión de jugador es válido. Una vez validada la sesión del jugador, Amazon GameLift Servers cambia el estado del espacio de jugador de RESERVADO a ACTIVO.

Sintaxis

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo gestiona una solicitud de conexión que incluye la validación y el rechazo de una sesión de jugador no válida. IDs

```
bool GameLiftManager::AcceptPlayerSession(const FString& playerSessionId, 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()

Informa a Amazon GameLift Servers de que un jugador se ha desconectado del proceso del servidor. Como respuesta, Amazon GameLift Servers cambia el espacio del jugador para que esté disponible.

Sintaxis

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
bool GameLiftManager::RemovePlayerSession(const FString& playerId)
{
    #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()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice este método para obtener información sobre los siguientes elementos:

- Una sesión para un jugador
- Todas las sesiones del jugador en una sesión de juego
- Todas las sesiones de jugador están asociadas a un único ID de jugador

Sintaxis

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameters

[FGameLiftDescribePlayerSessionsRequest](#)

Un objeto [the section called “FGameLiftDescribePlayerSessionsRequest”](#) que describe las sesiones de jugador que recuperar.

Valor devuelto

Si funciona correctamente, devuelve un objeto [the section called “FGameLiftDescribePlayerSessionsOutcome”](#) que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud.

Ejemplo

En este ejemplo se solicitan todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Al omitir NextTokeny establecer el valor límite en 10, Amazon GameLift Servers devuelve los registros de las sesiones de los primeros 10 jugadores que coincidan con la solicitud.

```

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()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan nuevos jugadores, Amazon GameLift Servers ofrece datos actualizados del emparejador mediante la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```

FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
&startBackfillRequest);

```

Parameters

FStartMatchBackfillRequest

StartMatchBackfillRequest Objeto que comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers generará uno.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se encuentra en los datos del emparejador de la sesión de juego.
- El ID de la sesión de juego que se va a reponer.
- Datos del emparejador disponibles para los jugadores actuales de la sesión de juego.

Valor devuelto

Devuelve un objeto StartMatchBackfillOutcome con el ID del ticket de reposición de emparejamiento o un error con un mensaje de error.

Ejemplo

```
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);
        };
        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()

Cancela una solicitud de reposición de emparejamiento activa. Para obtener más información, consulte [Característica de reposición de FlexMatch](#).

Sintaxis

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

Parameters

[FStopMatchBackfillRequest](#)

Un `StopMatchBackfillRequest` objeto que identifique el billete de emparejamiento que se va a cancelar:

- ID del ticket que se asignará a la solicitud de reposición
- El emparejador al que se envió la solicitud de reposición
- La sesión de juego asociada a la solicitud de reposición.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
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()

Recupera la ruta al certificado TLS utilizado para cifrar la conexión de red entre el recurso de computación de Amazon GameLift Servers Anywhere y Amazon GameLift Servers. Puede utilizar la ruta del certificado al registrar el dispositivo informático en una flota de Amazon GameLift Servers Anywhere. Para obtener más información, consulte, [RegisterCompute](#).

Sintaxis

```
FGameLiftGetComputeCertificateOutcome FGameLiftServerSDKModule::GetComputeCertificate()
```

Valor devuelto

Devuelve un objeto `GetComputeCertificateResponse` que contiene los siguientes elementos:

- **CertificatePath:** La ruta al certificado TLS de su recurso informático.
- **HostName:** el nombre de host de su recurso informático.

Ejemplo

```
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()

Recupera las credenciales del rol de IAM que autorizan a Amazon GameLift Servers a interactuar con otros Servicios de AWS. Para obtener más información, consulte [Conecta tu servidor de juegos Amazon GameLift Servers alojado a otros AWS recursos](#).

Sintaxis

```
FGameLiftGetFleetRoleCredentialsOutcome  
FGameLiftServerSDKModule::GetFleetRoleCredentials(const  
FGameLiftGetFleetRoleCredentialsRequest &request)
```

Parameters

[FGameLiftGetFleetRoleCredentialsRequest](#)

Valor devuelto

Devuelve un objeto [the section called “FGameLiftGetFleetRoleCredentialsOutcome”](#).

Ejemplo

```
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  
}
```

Destroy()

Libera de la memoria el SDK del servidor de juegos de Amazon GameLift Servers. Como práctica recomendada, llame a este método después de `ProcessEnding()` y antes de finalizar el proceso. Si utiliza una flota de Anywhere y no va a finalizar los procesos del servidor después de cada sesión de juego, llame a `Destroy()` y, a continuación, a `InitSDK()` para reinicializar antes de informar a Amazon GameLift Servers de que el proceso está listo para organizar una sesión de juego con `ProcessReady()`.

Sintaxis

```
FGameLiftGenericOutcome Destroy();
```

Parameters

No hay parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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 del servidor de las versiones 4 y anteriores de Amazon GameLift Servers

Esta referencia contiene la documentación relativa al SDK del servidor de las versiones 4 y anteriores de Amazon GameLift Servers. El SDK del servidor proporciona la funcionalidad principal que utiliza el servidor de juegos para comunicarse con el servicio de Amazon GameLift Servers. Por ejemplo, el servidor de juegos recibe peticiones del servicio para iniciar y detener las sesiones de juego, y le envía al servicio actualizaciones periódicas del estado de las sesiones de juego. Integre los servidores de juegos con el SDK del servidor antes de implementarlos para el alojamiento.

Utilice esta referencia del SDK del servidor con el fin de integrar servidores de juegos multijugador personalizados para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Para la versión 4.0.2 del SDK, puedes descargarla desde las [GitHub versiones oficiales](#). El paquete completo del SDK, que incluye el GameLiftLocal archivo.jar, está disponible en el artefacto GameLift-CSharp-ServerSDK -4.0.2.zip.

La versión principal más reciente del SDK del servidor de Amazon GameLift Servers es la 5.x. Las siguientes características de alojamiento requieren actualizaciones a la versión 5.x:

- Amazon GameLift Servers Anywhere

- Contenedores administrados de Amazon GameLift Servers
- Complemento de Amazon GameLift Servers para Unreal Engine y Unity

Temas

- [SDK del servidor de C++ para Amazon GameLift Servers 4.x: acciones](#)
- [SDK del servidor de C# para Amazon GameLift Servers 4.x: acciones](#)
- [SDK del servidor \(Unreal\) para Amazon GameLift Servers: acciones](#)

SDK del servidor de C++ para Amazon GameLift Servers 4.x: acciones

Utilice la referencia del SDK del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#).

SDK del servidor de C++ para Amazon GameLift Servers 4.x: tipos de datos

Utilice la referencia del SDK del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK 5.x del servidor de C++ para Amazon GameLift Servers: tipos de datos](#).

Esta API se define en `GameLiftServerAPI.h`, `LogParameters.h` y `ProcessParameters.h`.

[SDK del servidor de C++ para Amazon GameLift Servers 4.x: acciones](#)

DescribePlayerSessionsRequest

Este tipo de datos se utiliza para especificar qué sesión o sesiones de jugador recuperar. Puede utilizarlo para las siguientes tareas:

- Proporcione una `PlayerSessionId` para solicitar una sesión de jugador específica.
- Proporcione una `GameSessionId` para solicitar todas las sesiones de los jugadores de la sesión de juego especificada.
- Proporcione una `PlayerId` para solicitar todas las sesiones de jugador para el jugador especificado.

Para grandes conjuntos de sesiones de jugador, utilice los parámetros de paginación para recuperar los resultados en bloques consecutivos.

Contenido

GameSessionId

Identificador único de la sesión de juego. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de la sesión de juego es el siguiente: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. El valor de la <cadena de ID> es una cadena de ID personalizada o (si se especificó una cuando se creó la sesión de juego) una cadena generada.

Tipo: cadena

Requerido: no

Límite

Número máximo de resultados a devolver. Utilice este parámetro con `NextToken` para obtener resultados como un conjunto de páginas secuenciales. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: entero

Obligatorio: no

NextToken

Token que indica el inicio de la siguiente página de resultados secuencial. Utilice el token devuelto con una llamada anterior a esta acción. Para especificar el inicio del conjunto de

resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: cadena

Requerido: no

PlayerId

Identificador único de un jugador. IDs Los jugadores los define el desarrollador. Consulte [Generación de ID de jugador](#).

Tipo: cadena

Requerido: no

PlayerSessionId

Identificador único de una sesión de jugador.

Tipo: cadena

Requerido: no

PlayerSessionStatusFilter

Estado de la sesión de juego para filtrar los resultados. Los posibles estados de sesión de jugador son:

- **RESERVADO:** se ha recibido la solicitud de sesión del jugador, pero el jugador aún no se ha conectado al servidor y and/or se ha validado el proceso.
- **ACTIVE:** el proceso del servidor ha validado el jugador y actualmente está conectado.
- **COMPLETED:** ha caído la conexión del jugador.
- **TIEMPO DE ESPERA:** se recibió una solicitud de sesión de un jugador, pero el jugador que no and/or se conectó no se validó dentro del límite de tiempo de espera (60 segundos).

Tipo: cadena

Requerido: no

LogParameters

Este tipo de datos se utiliza para identificar los archivos generados durante una sesión de juego que desea que Amazon GameLift Servers cargue y almacene cuando finalice la sesión de juego. Esta información se comunicará al servicio Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Contenido

logPaths

Rutas de directorio a archivos de registro del servidor de juegos que desea que Amazon GameLift Servers almacene para futuros accesos. Estos archivos se generan durante cada sesión de juego. Los nombres y las rutas de los archivos se definen en el servidor de juegos y se almacenan en el directorio raíz de compilación del juego. Las rutas del registro deben ser absolutas. Por ejemplo, si la compilación del juego almacena los logs de sesión de juego en una ruta del tipo `MyGame\sessionlogs\`, entonces la ruta de los logs sería `c:\game\MyGame\sessionLogs` (en una instancia Windows) o `/local/game/MyGame/sessionLogs` (en una instancia Linux).

Tipo: `std::vector<std::string>`

Obligatorio: no

ProcessParameters

Este tipo de datos contiene el conjunto de parámetros enviado al servicio Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Contenido

puerto

Es el número de puerto en el que escucha el proceso del servidor para recibir conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: entero

Obligatorio: sí

logParameters

Objeto con una lista de rutas de directorio a archivos de log de la sesión de juego.

Tipo: Aws:::Servidor:GameLift: [LogParameters](#)

Obligatorio: no

onStartGameSession

Nombre de la función de devolución de llamada que el Amazon GameLift Servers servicio invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a la solicitud del cliente. [CreateGameSession](#) La función de devolución de llamada pasa un [GameSession](#) objeto (definido en la referencia de la API de Amazon GameLift Servers servicio).

Tipo: `const std::function<void(Aws::GameLift::Model::GameSession)>`
`onStartGameSession`

Obligatorio: sí

onProcessTerminate

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para forzar el cierre del proceso del servidor. Después de llamar a esta función, Amazon GameLift Servers espera cinco minutos hasta que el proceso del servidor se cierre y responda con una llamada a [ProcessEnding\(\)](#). Si no recibe respuesta, cierra el proceso del servidor.

Tipo: `std::function<void()>` `onProcessTerminate`

Obligatorio: no

onHealthCheck

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos. Después de llamar a esta función, Amazon GameLift Servers espera una respuesta durante 60 segundos y si no recibe ninguna, registra el proceso del servidor como en mal estado.

Tipo: `std::function<bool()>` `onHealthCheck`

Obligatorio: no

onUpdateGameSession

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de [reposición de emparejamiento](#) para proporcionar datos actualizados del emparejador. Pasa un objeto [GameSession](#), una actualización de estado (updateReason) y el ID del ticket de reposición de emparejamiento.

Tipo: `std::function<void(Aws::GameLift::Server::Model::UpdateGameSession)>`
`onUpdateGameSession`

Obligatorio: no

StartMatchBackfillRequest

Este tipo de datos se utiliza para enviar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Contenido

GameSessionArn

Identificador único de la sesión de juego. La acción de la API [GetGameSessionId\(\)](#) devuelve el identificador en formato de ARN.

Tipo: cadena

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único, en forma de un ARN, que el creador de emparejamientos utiliza para esta solicitud. Para encontrar el creador de emparejamientos que se usó para crear la sesión de juego original, busque en el objeto de sesión de juego, en la propiedad de datos del creador de emparejamientos. Puede obtener más información sobre los datos del emparejador en [Trabajo con datos del emparejador](#).

Tipo: cadena

Obligatorio: sí

Players

Un conjunto de datos que representa a todos los jugadores que están actualmente en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Para obtener una descripción del formato del objeto Player, consulte la Guía de referencia de API de Amazon GameLift Servers. Para encontrar los atributos de los jugadores y las asignaciones del equipo, busca en el objeto de sesión del juego, en la propiedad de datos del emparejador. IDs Si el creador de emparejamientos utiliza latencia, recopile la latencia actualizada para la región actual e inclúyala en los datos de cada jugador.

Tipo: `std::vector`[<player>](#)

Obligatorio: sí

TicketId

Identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no se proporciona ningún valor aquí, Amazon GameLift Servers generará uno en forma de UUID. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario.

Tipo: cadena

Requerido: no

StopMatchBackfillRequest

Este tipo de datos se utiliza para cancelar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Contenido

GameSessionArn

Identificador único de sesión de juego asociado a la solicitud que se va a cancelar.

Tipo: cadena

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único del creador de emparejamientos al que se envió esta solicitud.

Tipo: cadena

Obligatorio: sí

TicketId

Identificador único del ticket de solicitud de reposición que se va a cancelar.

Tipo: cadena

Obligatorio: sí

[SDK del servidor de C++ para Amazon GameLift Servers 4.x: tipos de datos](#)

Temas

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [ProcessReadyAsync\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)
- [Destroy\(\)](#)

AcceptPlayerSession()

Informa al servicio Amazon GameLift Servers de que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión del jugador es válido; es decir, que el ID de jugador ha reservado un espacio de jugador en la sesión de juego. Una vez completada la validación, Amazon GameLift Servers cambia el estado de la ranura de jugador de RESERVADO a ACTIVO.

Sintaxis

```
GenericOutcome AcceptPlayerSession(const std::string& playerSessionId);
```

Parameters

playerSessionId

ID único emitido por el Amazon GameLift Servers servicio en respuesta a una llamada a la acción de la Amazon GameLift Servers API del AWS SDK [CreatePlayerSession](#). El cliente de juego hace referencia a este ID cuando se conecta al proceso del servidor.

Tipo: `std::string`

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra una función para gestionar una solicitud de conexión, que incluye validar y rechazar una sesión de jugador no válida. 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()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor ha iniciado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de completar la inicialización de todas las sesiones de juego.

Sintaxis

```
GenericOutcome ActivateGameSession();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de devolución de llamada `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()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice esta acción para obtener información para una única sesión de jugador,

para todas las sesiones de jugador de una sesión de juego o para todas las sesiones de jugador asociadas a un solo ID de jugador.

Sintaxis

```
DescribePlayerSessionsOutcome DescribePlayerSessions (  
    const Aws::GameLift::Server::Model::DescribePlayerSessionsRequest  
    &describePlayerSessionsRequest);
```

Parameters

describePlayerSessionsSolicitud

Es un objeto [DescribePlayerSessionsRequest](#) que describe las sesiones de jugador a recuperar.

Obligatorio: sí

Valor devuelto

Si funciona correctamente, devuelve un objeto `DescribePlayerSessionsOutcome` que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud. Los objetos de sesión del jugador tienen una estructura idéntica al tipo de [PlayerSession](#) datos de la Amazon GameLift Servers API del AWS SDK.

Ejemplo

Este ejemplo muestra una solicitud de todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Al omitir `NextToken` y definir el valor de `Limit` en 10, Amazon GameLift Servers devuelve los primeros 10 registros de sesiones de jugador que coincidan con la solicitud.

```
// 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()

Recupera un identificador único de la sesión de juego alojada actualmente por el proceso del servidor, siempre que esté activo. El identificador se devuelve en el formato de ARN: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`.

En el caso de los procesos inactivos que no activados aún con una sesión de juego, la llamada devuelve `Success=True` y `GameSessionId=""` (una cadena vacía).

Sintaxis

```
AwsStringOutcome GetGameSessionId();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto `AwsStringOutcome`. Si no funciona, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsStringOutcome sessionIdOutcome =  
    Aws::GameLift::Server::GetGameSessionId();
```

GetInstanceCertificate()

Recupera la ubicación del archivo de un certificado TLS codificado con pem que está asociado a la flota y sus instancias. AWS Certificate Manager genera este certificado al crear una nueva flota con la configuración del certificado establecida en `GENERATED`. Utilice este certificado para establecer una conexión segura con un cliente de juego y para cifrar la comunicación cliente/servidor.

Sintaxis

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si se ejecuta correctamente, devuelve un objeto `GetInstanceCertificateOutcome` que contiene la ubicación del archivo de certificado TLS y la cadena de certificados de la flota, que se almacena en la instancia. En la instancia también se almacena un archivo de certificado raíz extraído de la cadena de certificados. Si no funciona, devuelve un mensaje de error.

Para obtener más información sobre el certificado y los datos de la cadena de certificados, consulte [los elementos de GetCertificate respuesta](#) en la referencia de la AWS Certificate Manager API.

Ejemplo

```
Aws::GameLift::GetInstanceCertificateOutcome certificateOutcome =  
    Aws::GameLift::Server::GetInstanceCertificate();
```

GetSdkVersion()

Devuelve el número de versión actual del SDK en uso.

Sintaxis

```
AwsStringOutcome GetSdkVersion();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto `AwsStringOutcome`. La cadena devuelta solo incluye el número de versión (por ejemplo, «3.1.5»). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

GetTerminationTime()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción tras recibir una devolución

de llamada `onProcessTerminate()` del servicio Amazon GameLift Servers. Amazon GameLift Servers puede llamar a `onProcessTerminate()` por los siguientes motivos: (1) cuando el proceso del servidor ha informado de un estado deficiente o no ha respondido a Amazon GameLift Servers; (2) cuando se finaliza la instancia durante un evento de reducción vertical; o (3) cuando una instancia se finaliza debido a una [interrupción de spot](#).

Si el proceso ha recibido una devolución de llamada `onProcessTerminate()`, el devuelto es la hora de terminación estimada. Si el proceso no ha recibido ninguna devolución de llamada `onProcessTerminate()`, se devuelve un mensaje de error. Más información acerca del [apagado de un proceso de servidor](#).

Sintaxis

```
AwsLongOutcome GetTerminationTime();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si el proceso se realiza correctamente, devuelve la hora de terminación como un objeto `AwsLongOutcome`. El valor es la hora de terminación expresado en ciclos transcurridos desde 0001 00:00:00. Por ejemplo, el valor de fecha y hora 13-09-2020 12:26:40 -000Z es igual a 6373559680000000000 ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsLongOutcome TermTimeOutcome =  
    Aws::GameLift::Server::GetTerminationTime();
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers. Este método debe llamarse en el momento del lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers.

Sintaxis

```
InitSDKOutcome InitSDK();
```


Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si se ejecuta correctamente, devuelve un `InitSdkOutcome` objeto que indica que el proceso del servidor está listo para la llamada [ProcessReady\(\)](#).

Ejemplo

```
Aws::GameLift::Server::InitSDKOutcome initOutcome =  
    Aws::GameLift::Server::InitSDK();
```

ProcessEnding()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor se va a detener. Este método debe llamarse después de realizar las demás tareas de limpieza, incluido el cierre de todas las sesiones de juego activas. Se debe salir de este método con un código de salida de 0; un código de salida que no sea 0 provoca un mensaje de evento que afirma que no se ha salido del proceso correctamente.

Después de que el método finalice con un código de 0, puede terminar el proceso con un código de salida correcto. También puede salir del proceso con un código de error. Si sale con un código de error, el evento de la flota indicará que el proceso ha finalizado incorrectamente (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Sintaxis

```
GenericOutcome ProcessEnding();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
Aws::GameLift::GenericOutcome outcome = Aws::GameLift::Server::ProcessEnding();
```

```
if (outcome.Success)
    exit(0); // exit with success
// otherwise, exit with error code
exit(errorCode);
```

ProcessReady()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Llame a este método después de invocar correctamente a [InitSDK\(\)](#) y completar las tareas de configuración necesarias antes de que el proceso del servidor pueda alojar una sesión de juego. Se debe llamar a este método solo una vez por proceso.

La llamada es síncrona. Para realizar una llamada asíncrona, utilice [ProcessReadyAsync\(\)](#). Consulte [Inicialización del proceso del servidor](#) para obtener más detalles.

Sintaxis

```
GenericOutcome ProcessReady(
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra las implementaciones tanto de la función de llamada como de la función de devolución de llamada [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()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Este método debe llamarse cuando el proceso del servidor esté listo para alojar una sesión de juego. Los parámetros especifican los nombres de las funciones de devolución de llamada a las que Amazon GameLift Servers llama en determinadas circunstancias. El código de servidor de juegos debe implementar estas funciones.

La llamada es asíncrona. Para realizar una llamada síncrona, utilice [ProcessReady\(\)](#). Consulte [Inicialización del proceso del servidor](#) para obtener más detalles.

Sintaxis

```
GenericOutcomeCallable ProcessReadyAsync(  
    const Aws::GameLift::Server::ProcessParameters &processParameters);
```

Parameters

processParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Notifica al servicio Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha desconectado del proceso del servidor. En respuesta, Amazon GameLift Servers cambia el estado de la ranura de jugador a disponible, por lo que se le puede asignar un jugador nuevo.

Sintaxis

```
GenericOutcome RemovePlayerSession(  
    const std::string& playerSessionId);
```

Parameters

playerSessionId

ID único emitido por el Amazon GameLift Servers servicio en respuesta a una llamada a la acción de la Amazon GameLift Servers API del AWS SDK [CreatePlayerSession](#). El cliente de juego hace referencia a este ID cuando se conecta al proceso del servidor.

Tipo: std::string

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
Aws::GameLift::GenericOutcome disconnectOutcome =  
    Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Consulte también la acción del AWS SDK [StartMatchBackfill\(\)](#). Con esta acción, un proceso del servidor de juegos que aloja la sesión de juego puede iniciar solicitudes de reposición de emparejamiento. Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan correctamente nuevos jugadores, el servicio Amazon GameLift Servers ofrece datos actualizados del emparejador invocando la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
StartMatchBackfillOutcome StartMatchBackfill (  
    const Aws::GameLift::Server::Model::StartMatchBackfillRequest  
    &startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

Un objeto [StartMatchBackfillRequest](#) que comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers lo generará automáticamente.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se puede obtener de los datos del creador de emparejamientos de la sesión de juego.
- El ID de la sesión de juego que está en fase de reposición.
- Datos de emparejamiento disponibles para los jugadores actuales de la sesión de juego.

Obligatorio: sí

Valor devuelto

Devuelve un `StartMatchBackfillOutcome` objeto con un mensaje de error que indica que se ha rellenado el partido o que ha fallado. El estado de las entradas se puede rastrear mediante la acción del AWS SDK [DescribeMatchmaking\(\)](#).

Ejemplo

```
// 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::Server::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()

Cancela una solicitud de reposición de emparejamiento activa que se creó con [StartMatchBackfill\(\)](#). Consulte también la acción del AWS SDK [StopMatchmaking\(\)](#). Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Sintaxis

```

GenericOutcome StopMatchBackfill (
    const Aws::GameLift::Server::Model::StopMatchBackfillRequest &stopBackfillRequest);

```

Parameters

StopMatchBackfillRequest

Un objeto [StopMatchBackfillRequest](#) que identifica el ticket de emparejamiento que se va a cancelar:

- ID del ticket asignado a la solicitud de reposición que se va a cancelar
- el creador de emparejamientos al que se envió la solicitud de reposición

- sesión de juego asociada a la solicitud de reposición

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Este método está obsoleto con la versión 4.0.1. En su lugar, el proceso del servidor debería llamar a [ProcessEnding\(\)](#) una vez finalizada la sesión de juego.

Informa al servicio de Amazon GameLift Servers de que el proceso del servidor ha finalizado la sesión de juego actual. Se llama a esta acción cuando el proceso del servidor permanece activo y listo para alojar una nueva sesión de juego. Solo debe llamarse una vez finalizado el procedimiento de finalización de la sesión de juego, ya que indica a Amazon GameLift Servers que el proceso del servidor está disponible inmediatamente para alojar una nueva sesión de juego.

No se llamará a esta acción si el proceso del servidor se interrumpe una vez finalizada la sesión de juego. En su lugar, se llamará a [ProcessEnding\(\)](#) para indicar que tanto la sesión de juego como el proceso del servidor están finalizando.

Sintaxis

```
GenericOutcome TerminateGameSession();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

UpdatePlayerSessionCreationPolicy()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores. Consulte también la acción del AWS SDK [UpdateGameSession\(\)](#).

Sintaxis

```
GenericOutcome UpdatePlayerSessionCreationPolicy(  
    Aws::GameLift::Model::PlayerSessionCreationPolicy newPlayerSessionPolicy);
```

Parameters

newPlayerSessionPolítica

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos.

Tipo: `Aws::GameLift::Model::PlayerSessionCreationPolicy` enum. Los valores válidos son:

- `ACCEPT_ALL`: se aceptan todas las sesiones de jugador nuevas.
- `DENY_ALL`: se rechazan todas las sesiones de jugador nuevas.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
Aws::GameLift::GenericOutcome outcome =  
    Aws::GameLift::Server::UpdatePlayerSessionCreationPolicy(Aws::GameLift::Model::PlayerSessionCr
```

Destroy()

Limpia la memoria asignada por `initSDK()` durante la inicialización del servidor de juegos. Utilice este método después de finalizar un proceso de servidor de juegos para evitar desperdiciar memoria del servidor.

Sintaxis

```
GenericOutcome Aws::GameLift::Server::Destroy();
```

Parameters

No hay parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

En este ejemplo, se limpia la memoria asignada por `InitSDK` una vez finalizado el proceso del servidor de juegos.

```
if (Aws::GameLift::Server::ProcessEnding().IsSuccess()) {  
    Aws::GameLift::Server::Destroy();  
    exit(0);  
}
```

SDK del servidor de C# para Amazon GameLift Servers 4.x: acciones

Utilice la referencia del SDK del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#).

SDK del servidor de C# para Amazon GameLift Servers 4.x: tipos de datos

Utilice la referencia del SDK del servidor para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK 5.x del servidor de C# para Amazon GameLift Servers: tipos de datos](#).

[SDK del servidor de C# para Amazon GameLift Servers 4.x: acciones](#)

Temas

- [LogParameters](#)
- [DescribePlayerSessionsRequest](#)
- [ProcessParameters](#)
- [StopMatchBackfillRequest](#)

LogParameters

Este tipo de datos se utiliza para identificar los archivos generados durante una sesión de juego que desea que Amazon GameLift Servers cargue y almacene cuando finalice la sesión de juego. Esta información se comunicará al servicio Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Contenido

logPaths

Lista de las rutas de directorio a archivos de log del servidor de juegos que desea que Amazon GameLift Servers almacene para futuros accesos. Estos archivos los genera un proceso del servidor durante cada sesión de juego; las rutas de archivo y los nombres se definen en el servidor de juegos y se almacenan en el directorio raíz de compilación de juegos. Las rutas del registro deben ser absolutas. Por ejemplo, si la compilación del juego almacena los logs de sesión de juego en una ruta del tipo `MyGame\sessionlogs\`, entonces la ruta de los logs sería

c:\game\MyGame\sessionLogs (en una instancia Windows) o /local/game/MyGame/sessionLogs (en una instancia Linux).

Tipo: Lista<String>

Obligatorio: no

DescribePlayerSessionsRequest

Este tipo de datos se utiliza para especificar qué sesión o sesiones de jugador recuperar. Se puede usar de varias maneras: (1) proporcionar una `PlayerSessionId` para solicitar una sesión de jugador específica; (2) proporcionar una `GameSessionId` para solicitar todas las sesiones de jugador de la sesión de juego especificada; o (3) proporcionar una `PlayerId` para solicitar todas las sesiones de jugador para el jugador especificado. Para grandes recopilaciones de sesiones de jugador, utilice los parámetros de paginación para recuperar resultados como páginas secuenciales.

Contenido

GameSessionId

Identificador único de la sesión de juego. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de la sesión de juego es el siguiente: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. El valor de la `<ID string>` es una cadena de ID personalizada (si se especificó una cuando se creó la sesión de juego) o una cadena generada.

Tipo: cadena

Requerido: no

Límite

Número máximo de resultados a devolver. Utilice este parámetro con `NextToken` para obtener los resultados como un conjunto de páginas secuenciales. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: entero

Obligatorio: no

NextToken

Token que indica el inicio de la siguiente página de resultados secuencial. Utilice el token devuelto con una llamada anterior a esta acción. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: cadena

Requerido: no

PlayerId

Identificador único de un jugador. IDs Los jugadores los define el desarrollador. Consulte [Generación de ID de jugador](#).

Tipo: cadena

Requerido: no

PlayerSessionId

Identificador único de una sesión de jugador.

Tipo: cadena

Requerido: no

PlayerSessionStatusFilter

Estado de la sesión de juego para filtrar los resultados. Los posibles estados de sesión de jugador son:

- **RESERVADO:** se ha recibido la solicitud de sesión del jugador, pero el jugador aún no se ha conectado al servidor y and/or se ha validado el proceso.
- **ACTIVE:** el proceso del servidor ha validado el jugador y actualmente está conectado.
- **COMPLETED:** ha caído la conexión del jugador.
- **TIEMPO DE ESPERA:** se recibió una solicitud de sesión de un jugador, pero el jugador que no and/or se conectó no se validó dentro del límite de tiempo de espera (60 segundos).

Tipo: cadena

Requerido: no

ProcessParameters

Este tipo de datos contiene el conjunto de parámetros enviado al servicio Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Contenido

puerto

Es el número de puerto al que escucha el proceso del servidor para conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: entero

Obligatorio: sí

logParameters

Objeto con una lista de rutas de directorio a archivos de log de la sesión de juego.

Tipo: `Aws::GameLift::Server::LogParameters`

Obligatorio: sí

onStartGameSession

Nombre de la función de devolución de llamada que el Amazon GameLift Servers servicio invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a la solicitud del cliente. [CreateGameSession](#) La función de devolución de llamada toma un [GameSession](#) objeto (definido en la referencia de la API de Amazon GameLift Servers servicio).

Tipo: `void OnStartGameSessionDelegate(GameSession gameSession)`

Obligatorio: sí

onProcessTerminate

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para forzar el cierre del proceso del servidor. Después de llamar a esta función, Amazon GameLift Servers espera cinco minutos hasta que el proceso del servidor se cierre y responde con una llamada a [ProcessEnding\(\)](#) antes de cerrar el proceso del servidor.

Tipo: `void OnProcessTerminateDelegate()`

Obligatorio: sí

`onHealthCheck`

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos. Después de llamar a esta función, Amazon GameLift Servers espera una respuesta durante 60 segundos y si no recibe ninguna, registra el proceso del servidor como en mal estado.

Tipo: `bool OnHealthCheckDelegate()`

Obligatorio: sí

`onUpdateGameSession`

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de [reposición de emparejamiento](#) para proporcionar datos actualizados del emparejador. Pasa un objeto [GameSession](#), una actualización de estado (`updateReason`) y el ID del ticket de reposición de emparejamiento.

Tipo: `void OnUpdateGameSessionDelegate (UpdateGameSession
updateGameSession)`

Obligatorio: no

`StartMatchBackfillRequest`

Este tipo de datos se utiliza para enviar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Contenido

`GameSessionArn`

Identificador único de la sesión de juego. El método del SDK [GetGameSessionId\(\)](#) devuelve el identificador en formato de ARN.

Tipo: cadena

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único, en forma de un ARN, que el creador de emparejamientos utiliza para esta solicitud. Para encontrar el creador de emparejamientos que se usó para crear la sesión de juego original, busque en el objeto de sesión de juego, en la propiedad de datos del creador de emparejamientos. Puede obtener más información sobre los datos del emparejador en [Trabajo con datos del emparejador](#).

Tipo: cadena

Obligatorio: sí

Players

Un conjunto de datos que representa a todos los jugadores que están actualmente en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Para obtener una descripción del formato del objeto Player, consulte la Guía de referencia de API de Amazon GameLift Servers. Para encontrar los atributos de los jugadores y las asignaciones del equipo, busca en el objeto de sesión del juego, en la propiedad de datos del emparejador. IDs Si el creador de emparejamientos utiliza latencia, recopile la latencia actualizada para la región actual e inclúyala en los datos de cada jugador.

Tipo: [Player](#)[]

Obligatorio: sí

TicketId

Identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no se proporciona ningún valor aquí, Amazon GameLift Servers generará uno en forma de UUID. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario.

Tipo: cadena

Requerido: no

StopMatchBackfillRequest

Este tipo de datos se utiliza para cancelar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Contenido

GameSessionArn

Identificador único de sesión de juego asociado a la solicitud que se va a cancelar.

Tipo: cadena

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único del creador de emparejamientos al que se envió esta solicitud.

Tipo: cadena

Obligatorio: sí

TicketId

Identificador único del ticket de solicitud de reposición que se va a cancelar.

Tipo: cadena

Obligatorio: sí

[SDK del servidor de C# para Amazon GameLift Servers 4.x: tipos de datos](#)

Temas

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [GetTerminationTime\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)

- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Informa al servicio Amazon GameLift Servers de que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión del jugador es válido; es decir, que el ID de jugador ha reservado un espacio de jugador en la sesión de juego. Una vez completada la validación, Amazon GameLift Servers cambia el estado de la ranura de jugador de RESERVADO a ACTIVO.

Sintaxis

```
GenericOutcome AcceptPlayerSession(String playerId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador. El ID de sesión de un jugador se especifica en un `PlayerSession` objeto, que se devuelve en respuesta a una llamada de un cliente a las acciones de la GameLift API [StartGameSessionPlacement CreateGameSession](#), [DescribeGameSessionPlacement](#), o [DescribePlayerSessions](#).

Tipo: cadena

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra una función para gestionar una solicitud de conexión, incluida la validación y el rechazo de una sesión de jugador no válida. IDs

```
void ReceiveConnectingPlayerSessionID (Connection connection, String playerId){
    var acceptPlayerSessionOutcome =
    GameLiftServerAPI.AcceptPlayerSession(playerSessionId);
    if(acceptPlayerSessionOutcome.Success)
    {
        connectionToSessionMap.emplace(connection, playerId);
        connection.Accept();
    }
    else
    {
        connection.Reject(acceptPlayerSessionOutcome.Error.ErrorMessage);    }
}
```

ActivateGameSession()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor ha activado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de completar la inicialización de todas las sesiones de juego.

Sintaxis

```
GenericOutcome ActivateGameSession()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo muestra cómo se llama a `ActivateGameSession()` como parte de la función de delegación `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()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice esta acción para obtener información para una única sesión de jugador, para todas las sesiones de jugador de una sesión de juego o para todas las sesiones de jugador asociadas a un solo ID de jugador.

Sintaxis

```
DescribePlayerSessionsOutcome DescribePlayerSessions(DescribePlayerSessionsRequest  
describePlayerSessionsRequest)
```

Parameters

describePlayerSessionsSolicitud

Es un objeto [DescribePlayerSessionsRequest](#) que describe las sesiones de jugador a recuperar.

Obligatorio: sí

Valor devuelto

Si funciona correctamente, devuelve un objeto `DescribePlayerSessionsOutcome` que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud. Los objetos de sesión del jugador tienen una estructura idéntica al tipo de [PlayerSession](#) datos de la Amazon GameLift Servers API del AWS SDK.

Ejemplo

Este ejemplo muestra una solicitud de todas las sesiones de jugador conectadas activamente a una sesión de juego específica. Al omitir `NextToken` y establecer el valor límite en 10, Amazon GameLift Servers se devolverán los registros de las primeras 10 sesiones de jugador que coincidan con la solicitud.

```
// 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()

Recupera el ID de la sesión de juego alojada actualmente por el proceso del servidor, siempre que esté activo.

En el caso de los procesos inactivos que no activados aún con una sesión de juego, la llamada devuelve `Success=True` y `GameSessionId=""` (una cadena vacía).

Sintaxis

```
AwsStringOutcome GetGameSessionId()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto `AwsStringOutcome`. Si no funciona, devuelve un mensaje de error.

Ejemplo

```
var getGameSessionIdOutcome = GameLiftServerAPI.GetGameSessionId();
```

GetInstanceCertificate()

Recupera la ubicación del archivo de un certificado TLS codificado con PEM que está asociado a la flota y sus instancias. AWS Certificate Manager genera este certificado al crear una nueva flota con la configuración del certificado establecida en `GENERATED`. Utilice este certificado para establecer una conexión segura con un cliente de juego y para cifrar la comunicación cliente/servidor.

Sintaxis

```
GetInstanceCertificateOutcome GetInstanceCertificate();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si se ejecuta correctamente, devuelve un objeto `GetInstanceCertificateOutcome` que contiene la ubicación del archivo de certificado TLS y la cadena de certificados de la flota, que se almacena en la instancia. En la instancia también se almacena un archivo de certificado raíz extraído de la cadena de certificados. Si no funciona, devuelve un mensaje de error.

Para obtener más información sobre el certificado y los datos de la cadena de certificados, consulte [los elementos de `GetCertificate` respuesta](#) en la referencia de la AWS Certificate Manager API.

Ejemplo

```
var getInstanceCertificateOutcome = GameLiftServerAPI.GetInstanceCertificate();
```

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
AwsStringOutcome GetSdkVersion()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto `AwsStringOutcome`. La cadena devuelta solo incluye el número de versión (por ejemplo, «3.1.5»). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
var getSdkVersionOutcome = GameLiftServerAPI.GetSdkVersion();
```

GetTerminationTime()

Devuelve la hora a la que está programada el cierre de un proceso de servidor, si hay una hora de terminación disponible. Un proceso del servidor realiza esta acción tras recibir una devolución de llamada `onProcessTerminate()` del servicio Amazon GameLift Servers. Amazon GameLift Servers puede llamar a `onProcessTerminate()` por los siguientes motivos: (1) por un estado deficiente (el proceso del servidor ha informado de un estado deficiente o no ha respondido a Amazon GameLift Servers); (2) cuando se finaliza la instancia durante un evento de reducción vertical; o (3) cuando una instancia se finaliza debido a la [interrupción de una instancia de spot](#).

Si el proceso ha recibido una devolución de llamada `onProcessTerminate()`, el devuelto es la hora de terminación estimada. Si el proceso no ha recibido ninguna devolución de llamada `onProcessTerminate()`, se devuelve un mensaje de error. Más información acerca del [apagado de un proceso de servidor](#).

Sintaxis

```
AwsDateTimeOutcome GetTerminationTime()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si el proceso se realiza correctamente, devuelve la hora de terminación como un objeto `AwsDateTimeOutcome`. El valor es la hora de terminación expresado en ciclos transcurridos desde 0001 00:00:00. Por ejemplo, el valor de fecha y hora 13-09-2020 12:26:40 -000Z es igual a 6373559680000000000 ciclos. Si no hay una hora de terminación disponible, devuelve un mensaje de error.

Ejemplo

```
var getTerminationTimeOutcome = GameLiftServerAPI.GetTerminationTime();
```


InitSDK()

Inicializa el SDK de Amazon GameLift Servers. Este método debe llamarse en el momento del lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers.

Sintaxis

```
InitSDKOutcome InitSDK()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si se ejecuta correctamente, devuelve un `InitSdkOutcome` objeto que indica que el proceso del servidor está listo para la llamada [ProcessReady\(\)](#).

Ejemplo

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

ProcessEnding()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor se va a detener. Este método debe llamarse después de realizar las demás tareas de limpieza, incluido el cierre de todas las sesiones de juego activas. Se debe salir de este método con un código de salida de 0; un código de salida que no sea 0 provoca un mensaje de evento que afirma que no se ha salido del proceso correctamente.

Después de que el método finalice con un código de 0, puede terminar el proceso con un código de salida correcto. También puede salir del proceso con un código de error. Si sale con un código de error, el evento de la flota indicará que el proceso ha finalizado incorrectamente (`SERVER_PROCESS_TERMINATED_UNHEALTHY`).

Sintaxis

```
GenericOutcome ProcessEnding()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
var processEndingOutcome = GameLiftServerAPI.ProcessEnding();
if (processReadyOutcome.Success)
    Environment.Exit(0);
// otherwise, exit with error code
Environment.Exit(errorCode);
```

ProcessReady()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Llame a este método después de invocar correctamente a [InitSDK\(\)](#) y completar las tareas de configuración necesarias antes de que el proceso del servidor pueda alojar una sesión de juego. Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
GenericOutcome ProcessReady(ProcessParameters processParameters)
```

Parameters

processParameters

Es un objeto [ProcessParameters](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra las implementaciones tanto de la función de llamada [ProcessReady\(\)](#) como de la función de delegación.

```
// 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
        {
            "C:\\game\\logs",
            "C:\\game\\error"
        })
    written by the game server
);

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()

Notifica al servicio Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha desconectado del proceso del servidor. En respuesta, Amazon GameLift Servers cambia el estado de la ranura de jugador a disponible, por lo que se le puede asignar un jugador nuevo.

Sintaxis

```
GenericOutcome RemovePlayerSession(String playerId)
```

Parameters

playerSessionId

ID único emitido por Amazon GameLift Servers cuando se crea una nueva sesión de jugador. El identificador de sesión de un jugador se especifica en un `PlayerSession` objeto, que se devuelve en respuesta a una llamada del cliente a las acciones de la GameLift API [StartGameSessionPlacement](#), [CreateGameSession](#), [DescribeGameSessionPlacement](#), o [DescribePlayerSessions](#).

Tipo: cadena

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
Aws::GameLift::GenericOutcome disconnectOutcome =
```

```
Aws::GameLift::Server::RemovePlayerSession(playerSessionId);
```

StartMatchBackfill()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Consulta también la acción del AWS SDK [StartMatchBackfill\(\)](#). Con esta acción, un proceso del servidor de juegos que aloja la sesión de juego puede iniciar solicitudes de reposición de emparejamiento. Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan correctamente nuevos jugadores, el servicio Amazon GameLift Servers ofrece datos actualizados del emparejador por medio de la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
StartMatchBackfillOutcome StartMatchBackfill (StartMatchBackfillRequest  
startBackfillRequest);
```

Parameters

StartMatchBackfillRequest

Un objeto [StartMatchBackfillRequest](#) que comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers lo generará automáticamente.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se puede obtener de los datos del creador de emparejamientos de la sesión de juego.
- El ID de la sesión de juego que está en fase de reposición.
- Datos de emparejamiento disponibles para los jugadores actuales de la sesión de juego.

Obligatorio: sí

Valor devuelto

Devuelve un `StartMatchBackfillOutcome` objeto con el identificador del billete de relleno correspondiente o un error con un mensaje de error.

Ejemplo

```
// 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()

Cancela una solicitud de reposición de emparejamiento activa que se creó con [StartMatchBackfill\(\)](#). Consulte también la acción del AWS SDK [StopMatchmaking\(\)](#). Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Sintaxis

```
GenericOutcome StopMatchBackfill (StopMatchBackfillRequest stopBackfillRequest);
```

Parameters

StopMatchBackfillRequest

Un objeto [StopMatchBackfillRequest](#) que identifica el ticket de emparejamiento que se va a cancelar:

- ID del ticket asignado a la solicitud de reposición que se va a cancelar
- el creador de emparejamientos al que se envió la solicitud de reposición
- sesión de juego asociada a la solicitud de reposición

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

```
// 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()

Este método está obsoleto con la versión 4.0.1. En su lugar, el proceso del servidor debería llamar a [ProcessEnding\(\)](#) una vez finalizada la sesión de juego.

Informa al servicio de Amazon GameLift Servers de que el proceso del servidor ha finalizado la sesión de juego actual. Se llama a esta acción cuando el proceso del servidor permanece activo y listo para alojar una nueva sesión de juego. Solo debe llamarse una vez finalizado el procedimiento

de finalización de la sesión de juego, ya que indica a Amazon GameLift Servers que el proceso del servidor está disponible inmediatamente para alojar una nueva sesión de juego.

No se llamará a esta acción si el proceso del servidor se interrumpe una vez finalizada la sesión de juego. En su lugar, se llamará a [ProcessEnding\(\)](#) para indicar que tanto la sesión de juego como el proceso del servidor están finalizando.

Sintaxis

```
GenericOutcome TerminateGameSession()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo ilustra un proceso del servidor al final de una sesión de juego.

```
// 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()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores. (Consulte también la acción [UpdateGameSession\(\)](#) en la Referencia de la API de Amazon GameLift Servers servicio).

Sintaxis

```
GenericOutcome UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy  
playerSessionPolicy)
```


Parameters

newPlayerSessionPolítica

Valor de cadena que indica si la sesión de juego acepta jugadores nuevos.

Tipo: [PlayerSessionCreationPolicy](#) enum. Los valores válidos son:

- ACCEPT_ALL: se aceptan todas las sesiones de jugador nuevas.
- DENY_ALL: se rechazan todas las sesiones de jugador nuevas.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Este ejemplo establece la política de participación en la sesión de juego actual para aceptar todos los jugadores.

```
var updatePlayerSessionCreationPolicyOutcome =  
  
    GameLiftServerAPI.UpdatePlayerSessionCreationPolicy(PlayerSessionCreationPolicy.ACCEPT_ALL);
```

SDK del servidor (Unreal) para Amazon GameLift Servers: acciones

Utilice el SDK del servidor para Unreal como ayuda para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#).

Esta API se define en `GameLiftServerSDK.h` y `GameLiftServerSDKModels.h`.

Para configurar el complemento Unreal Engine y ver ejemplos de código, consulte [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#).

SDK del servidor (Unreal) para Amazon GameLift Servers: tipos de datos

Utilice la referencia del SDK del servidor de Amazon GameLift Servers para Unreal como ayuda para integrar su juego multijugador para el alojamiento con Amazon GameLift Servers. Para obtener ayuda con el proceso de integración, consulte [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#).

Note

Esta referencia es para una versión anterior del SDK del servidor de Amazon GameLift Servers. Para obtener la última versión, consulte [SDK del servidor de C++ \(Unreal\) 5.x para Amazon GameLift Servers: tipos de datos](#).

Esta API se define en `GameLiftServerSDK.h` y `GameLiftServerSDKModels.h`.

Para configurar el complemento Unreal Engine y ver ejemplos de código, consulte [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#).

[SDK del servidor \(Unreal\) para Amazon GameLift Servers: acciones](#)

Temas

- [FDescribePlayerSessionsRequest](#)
- [FProcessParámetros](#)
- [FStartMatchBackfillRequest](#)
- [FStopMatchBackfillRequest](#)

FDescribePlayerSessionsRequest

Este tipo de datos se utiliza para especificar qué sesión o sesiones de jugador recuperar. Puede utilizarlo para las siguientes tareas:

- Proporcione una `PlayerSessionId` para solicitar una sesión de jugador específica.
- Proporciona una `GameSessionId` para solicitar todas las sesiones de los jugadores de la sesión de juego especificada.

- Proporciona una `PlayerId` para solicitar todas las sesiones de jugador del jugador especificado.

Para grandes conjuntos de sesiones de jugador, utilice los parámetros de paginación para recuperar los resultados en bloques consecutivos.

Contenido

GameSessionId

Identificador único de la sesión de juego. Use este parámetro para solicitar todas las sesiones de jugador de la sesión de juego especificada. El formato de ID de la sesión de juego es el siguiente: `arn:aws:gamelift:<region>::gamesession/fleet-<fleet ID>/<ID string>`. El valor de la `<cadena de ID>` es una cadena de ID personalizada o (si se especificó una cuando se creó la sesión de juego) una cadena generada.

Tipo: cadena

Requerido: no

Límite

Número máximo de resultados a devolver. Utilice este parámetro con `NextToken` para obtener resultados como un conjunto de páginas secuenciales. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: entero

Obligatorio: no

NextToken

Token que indica el inicio de la siguiente página de resultados secuencial. Utilice el token devuelto con una llamada anterior a esta acción. Para especificar el inicio del conjunto de resultados, no indique ningún valor. Si se especifica un ID de sesión de jugador, este parámetro se ignora.

Tipo: cadena

Requerido: no

PlayerId

Identificador único de un jugador. IDs Los jugadores los define el desarrollador. Consulte [Generación de ID de jugador](#).

Tipo: cadena

Requerido: no

PlayerSessionId

Identificador único de una sesión de jugador.

Tipo: cadena

Requerido: no

PlayerSessionStatusFilter

Estado de la sesión de juego para filtrar los resultados. Los posibles estados de sesión de jugador son:

- **RESERVADO:** se ha recibido la solicitud de sesión del jugador, pero el jugador aún no se ha conectado al servidor y and/or se ha validado el proceso.
- **ACTIVE:** el proceso del servidor ha validado el jugador y actualmente está conectado.
- **COMPLETED:** ha caído la conexión del jugador.
- **TIEMPO DE ESPERA:** se recibió una solicitud de sesión de un jugador, pero el jugador que no and/or se conectó no se validó dentro del límite de tiempo de espera (60 segundos).

Tipo: cadena

Requerido: no

FProcessParámetros

Este tipo de datos contiene el conjunto de parámetros enviado al servicio Amazon GameLift Servers en una llamada a [ProcessReady\(\)](#).

Contenido

puerto

Es el número de puerto al que escucha el proceso del servidor para conexiones de jugador nuevas. El valor debe estar en el rango de puertos configurado para cualquier flota que implemente esa compilación de servidor de juegos. Este número de puerto se incluye en

los objetos de sesión de juego y de jugador, que las sesiones de juego utilizan a la hora de conectarse a un proceso del servidor.

Tipo: entero

Obligatorio: sí

logParameters

Objeto con una lista de rutas de directorio a archivos de log de la sesión de juego.

Tipo: TArray < FString >

Obligatorio: no

onStartGameSession

Nombre de la función de devolución de llamada que el Amazon GameLift Servers servicio invoca para activar una nueva sesión de juego. Amazon GameLift Servers llama a esta función en respuesta a la solicitud del cliente. [CreateGameSession](#) La función de devolución de llamada toma un [GameSession](#) objeto (definido en la referencia de la API de Amazon GameLift Servers servicio).

Tipo: FOnStartGameSession

Obligatorio: sí

onProcessTerminate

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para forzar el cierre del proceso del servidor. Después de llamar a esta función, Amazon GameLift Servers espera cinco minutos hasta que el proceso del servidor se cierre y responde con una llamada a [ProcessEnding\(\)](#) antes de cerrar el proceso del servidor.

Tipo: delegado FSimple

Obligatorio: no

onHealthCheck

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para solicitar un informe de estado del proceso del servidor. Amazon GameLift Servers llama a esta función cada 60 segundos. Después de llamar a esta función, Amazon GameLift Servers

espera una respuesta durante 60 segundos y si no recibe ninguna, registra el proceso del servidor como en mal estado.

Tipo: FOnHealthCheck

Obligatorio: no

onUpdateGameSession

Nombre de la función de devolución de llamada que invoca el servicio Amazon GameLift Servers para proporcionar un objeto de sesión de juego actualizado. Amazon GameLift Servers llama a esta función una vez que se ha procesado una solicitud de [reposición de emparejamiento](#) para proporcionar datos actualizados del emparejador. Pasa un objeto [GameSession](#), una actualización de estado (updateReason) y el ID del ticket de reposición de emparejamiento.

Tipo: FOnUpdateGameSession

Obligatorio: no

FStartMatchBackfillRequest

Este tipo de datos se utiliza para enviar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StartMatchBackfill\(\)](#).

Contenido

GameSessionArn

Identificador único de la sesión de juego. La acción de la API [GetGameSessionId\(\)](#) devuelve el identificador en formato de ARN.

Tipo: FString

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único, en forma de un ARN, que el creador de emparejamientos utiliza para esta solicitud. Para encontrar el creador de emparejamientos que se usó para crear la sesión de juego original, busque en el objeto de sesión de juego, en la propiedad de datos del creador de emparejamientos. Puede obtener más información sobre los datos del emparejador en [Trabajo con datos del emparejador](#).

Tipo: FString

Obligatorio: sí

Players

Un conjunto de datos que representa a todos los jugadores que están actualmente en la sesión de juego. El creador de emparejamientos utiliza esta información para buscar nuevos jugadores que son idóneos para los jugadores actuales. Para obtener una descripción del formato del objeto Player, consulte la Guía de referencia de API de Amazon GameLift Servers. Para encontrar los atributos de los jugadores y las asignaciones del equipo, busca en el objeto de sesión del juego, en la propiedad de datos del emparejador. IDs Si el creador de emparejamientos utiliza latencia, recopile la latencia actualizada para la región actual e inclúyala en los datos de cada jugador.

Escribe: TArray [< > FPlayer](#)

Obligatorio: sí

TicketId

Identificador único para un ticket de solicitud de emparejamiento o reposición de emparejamiento. Si no se proporciona ningún valor aquí, Amazon GameLift Servers generará uno en forma de UUID. Use este identificador para realizar un seguimiento del estado del ticket de reposición de emparejamiento o cancelar la solicitud si es necesario.

Tipo: FString

Obligatorio: no

FStopMatchBackfillRequest

Este tipo de datos se utiliza para cancelar una solicitud de reposición de emparejamiento. La información se comunica al servicio Amazon GameLift Servers en una llamada a [StopMatchBackfill\(\)](#).

Contenido

GameSessionArn

Identificador único de sesión de juego asociado a la solicitud que se va a cancelar.

Tipo: FString

Obligatorio: sí

MatchmakingConfigurationArn

Identificador único del creador de emparejamientos al que se envió esta solicitud.

Tipo: FString

Obligatorio: sí

TicketId

Identificador único del ticket de solicitud de reposición que se va a cancelar.

Tipo: FString

Obligatorio: sí

SDK del servidor (Unreal) para Amazon GameLift Servers: tipos de datos

Temas

- [AcceptPlayerSession\(\)](#)
- [ActivateGameSession\(\)](#)
- [DescribePlayerSessions\(\)](#)
- [GetGameSessionId\(\)](#)
- [GetInstanceCertificate\(\)](#)
- [GetSdkVersion\(\)](#)
- [InitSDK\(\)](#)
- [ProcessEnding\(\)](#)
- [ProcessReady\(\)](#)
- [RemovePlayerSession\(\)](#)
- [StartMatchBackfill\(\)](#)
- [StopMatchBackfill\(\)](#)
- [TerminateGameSession\(\)](#)
- [UpdatePlayerSessionCreationPolicy\(\)](#)

AcceptPlayerSession()

Informa al servicio Amazon GameLift Servers de que un jugador con el ID de sesión de jugador especificado se ha conectado al proceso del servidor y requiere validación. Amazon GameLift Servers verifica que el ID de sesión del jugador es válido; es decir, que el ID de jugador ha reservado un espacio de jugador en la sesión de juego. Una vez completada la validación, Amazon GameLift Servers cambia el estado de la ranura de jugador de RESERVADO a ACTIVO.

Sintaxis

```
FGameLiftGenericOutcome AcceptPlayerSession(const FString& playerId)
```

Parameters

playerSessionId

ID único emitido por el Amazon GameLift Servers servicio en respuesta a una llamada a la acción de la Amazon GameLift Servers API del AWS SDK [CreatePlayerSession](#). El cliente de juego hace referencia a este ID cuando se conecta al proceso del servidor.

Tipo: FString

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

ActivateGameSession()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor ha activado una sesión de juego y que está listo para recibir las conexiones de los jugadores. Esta acción debe llamarse como parte de la función de devolución de llamada `onStartGameSession()`, después de completar la inicialización de todas las sesiones de juego.

Sintaxis

```
FGameLiftGenericOutcome ActivateGameSession()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

DescribePlayerSessions()

Recupera datos de sesión de jugador, incluida la configuración, los metadatos de la sesión y los datos de jugador. Utilice esta acción para obtener información para una única sesión de jugador, para todas las sesiones de jugador de una sesión de juego o para todas las sesiones de jugador asociadas a un solo ID de jugador.

Sintaxis

```
FGameLiftDescribePlayerSessionsOutcome DescribePlayerSessions(const  
FGameLiftDescribePlayerSessionsRequest &describePlayerSessionsRequest)
```

Parameters

describePlayerSessionsSolicitud

Es un objeto [FDescribePlayerSessionsRequest](#) que describe las sesiones de jugador a recuperar.

Obligatorio: sí

Valor devuelto

Si funciona correctamente, devuelve un objeto [FDescribePlayerSessionsRequest](#) que contiene un conjunto de objetos de sesión de jugador que se ajusta a los parámetros de la solicitud. Los objetos de sesión del jugador tienen una estructura idéntica al tipo de [PlayerSession](#) datos de la Amazon GameLift Servers API del AWS SDK.

GetGameSessionId()

Recupera el ID de la sesión de juego alojada actualmente por el proceso del servidor, siempre que esté activo.

Sintaxis

```
FGameLiftStringOutcome GetGameSessionId()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve el ID de sesión del juego como objeto `FGameLiftStringOutcome`. Si no funciona, devuelve un mensaje de error.

GetInstanceCertificate()

Recupera la ubicación del archivo de un certificado TLS codificado con pem que está asociado a la flota y sus instancias. AWS Certificate Manager genera este certificado al crear una nueva flota con la configuración del certificado establecida en `GENERATED`. Utilice este certificado para establecer una conexión segura con un cliente de juego y para cifrar la comunicación cliente/servidor.

Sintaxis

```
FGameLiftGetInstanceCertificateOutcome GetInstanceCertificate()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si se ejecuta correctamente, devuelve un objeto `GetInstanceCertificateOutcome` que contiene la ubicación del archivo de certificado TLS y la cadena de certificados de la flota, que se almacena en la instancia. En la instancia también se almacena un archivo de certificado raíz extraído de la cadena de certificados. Si no funciona, devuelve un mensaje de error.

Para obtener más información sobre el certificado y los datos de la cadena de certificados, consulte [los elementos de GetCertificate respuesta](#) en la referencia de la AWS Certificate Manager API.

GetSdkVersion()

Devuelve el número de versión actual del SDK integrado en el proceso del servidor.

Sintaxis

```
FGameLiftStringOutcome GetSdkVersion();
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Si funciona correctamente, devuelve la versión del SDK actual como objeto `FGameLiftStringOutcome`. La cadena devuelta solo incluye el número de versión (por ejemplo, «3.1.5»). Si no funciona, devuelve un mensaje de error.

Ejemplo

```
Aws::GameLift::AwsStringOutcome SdkVersionOutcome =  
    Aws::GameLift::Server::GetSdkVersion();
```

InitSDK()

Inicializa el SDK de Amazon GameLift Servers. Este método debe llamarse en el momento del lanzamiento, antes de cualquier otra inicialización relacionada con Amazon GameLift Servers.

Sintaxis

```
FGameLiftGenericOutcome InitSDK()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

ProcessEnding()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor se va a detener. Este método debe llamarse después de realizar las demás tareas de limpieza, incluido el cierre de todas las sesiones de juego activas. Se debe salir de este método con un código de salida de 0; un código

de salida que no sea 0 provoca un mensaje de evento que afirma que no se ha salido del proceso correctamente.

Sintaxis

```
FGameLiftGenericOutcome ProcessEnding()
```

Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

ProcessReady()

Notifica al servicio Amazon GameLift Servers que el proceso del servidor está listo para alojar sesiones de juego. Llame a este método después de invocar correctamente a [InitSDK\(\)](#) y completar las tareas de configuración necesarias antes de que el proceso del servidor pueda alojar una sesión de juego. Se debe llamar a este método solo una vez por proceso.

Sintaxis

```
FGameLiftGenericOutcome ProcessReady(FProcessParameters &processParameters)
```

Parameters

FProcessParámetros

Es un objeto [FProcessParámetros](#) que comunica la siguiente información acerca del proceso del servidor:

- Nombres de métodos de devolución de llamada, implementados en el código de servidor de juegos, que el servicio Amazon GameLift Servers invoca para comunicarse con el proceso del servidor.
- Número de puerto de escucha del servidor de proceso.
- Ruta a cualquier archivo específico de la sesión de juego que desea que Amazon GameLift Servers capture y almacene.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Ejemplo

Consulte el código de ejemplo en [Using the Unreal Engine Plugin](#).

RemovePlayerSession()

Notifica al servicio Amazon GameLift Servers que un jugador con el ID de sesión de jugador especificado se ha desconectado del proceso del servidor. En respuesta, Amazon GameLift Servers cambia el estado de la ranura de jugador a disponible, por lo que se le puede asignar un jugador nuevo.

Sintaxis

```
FGameLiftGenericOutcome RemovePlayerSession(const FString& playerSessionId)
```

Parameters

playerSessionId

ID único emitido por el Amazon GameLift Servers servicio en respuesta a una llamada a la acción de la Amazon GameLift Servers API del AWS SDK [CreatePlayerSession](#). El cliente de juego hace referencia a este ID cuando se conecta al proceso del servidor.

Tipo: FString

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

StartMatchBackfill()

Envía una solicitud para encontrar nuevos jugadores para ranuras abiertas en una sesión de juego creada con FlexMatch. Consulte también la acción del AWS SDK [StartMatchBackfill\(\)](#). Con esta acción, un proceso del servidor de juegos que aloja la sesión de juego puede iniciar solicitudes de reposición de emparejamiento. Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Esta acción es asíncrona. Si se emparejan correctamente nuevos jugadores, el servicio Amazon GameLift Servers ofrece datos actualizados del emparejador por medio de la función de devolución de llamada `OnUpdateGameSession()`.

Un proceso del servidor solo puede tener una solicitud de reposición de emparejamiento activa a la vez. Para enviar una nueva solicitud, en primer lugar llame a [StopMatchBackfill\(\)](#) para cancelar la solicitud original.

Sintaxis

```
FGameLiftStringOutcome StartMatchBackfill (FStartMatchBackfillRequest
&startBackfillRequest);
```

Parameters

FStartMatchBackfillRequest

Un objeto [FStartMatchBackfillRequest](#) que comunica la siguiente información:

- Un ID de ticket que se asignará a la solicitud de reposición. Esta información es opcional; si no se proporciona ningún ID, Amazon GameLift Servers lo generará automáticamente.
- El creador de emparejamientos al que se enviará la solicitud. El ARN de configuración completo es obligatorio. Este valor se puede obtener de los datos del creador de emparejamientos de la sesión de juego.
- El ID de la sesión de juego que está en fase de reposición.
- Datos de emparejamiento disponibles para los jugadores actuales de la sesión de juego.

Obligatorio: sí

Valor devuelto

Si funciona correctamente, devuelve el ticket de reposición de emparejamiento como un objeto `FGameLiftStringOutcome`. Si no funciona, devuelve un mensaje de error. Se puede realizar un seguimiento del estado de los tickets mediante la acción del AWS SDK [DescribeMatchmaking\(\)](#).

StopMatchBackfill()

Cancela una solicitud de reposición de emparejamiento activa que se creó con [StartMatchBackfill\(\)](#). Consulte también la acción del AWS SDK [StopMatchmaking\(\)](#). Obtenga más información sobre la [característica de reposición de FlexMatch](#).

Sintaxis

```
FGameLiftGenericOutcome StopMatchBackfill (FStopMatchBackfillRequest  
&stopBackfillRequest);
```

Parameters

StopMatchBackfillRequest

Un objeto [FStopMatchBackfillRequest](#) que identifica el ticket de emparejamiento que se va a cancelar:

- ID del ticket asignado a la solicitud de reposición que se va a cancelar
- el creador de emparejamientos al que se envió la solicitud de reposición
- sesión de juego asociada a la solicitud de reposición

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

TerminateGameSession()

Este método está obsoleto con la versión 4.0.1. En su lugar, el proceso del servidor debería llamar a [ProcessEnding\(\)](#) una vez finalizada la sesión de juego.

Informa al servicio de Amazon GameLift Servers de que el proceso del servidor ha finalizado la sesión de juego actual. Se llama a esta acción cuando el proceso del servidor permanece activo y listo para alojar una nueva sesión de juego. Solo debe llamarse una vez finalizado el procedimiento de finalización de la sesión de juego, ya que indica a Amazon GameLift Servers que el proceso del servidor está disponible inmediatamente para alojar una nueva sesión de juego.

No se llamará a esta acción si el proceso del servidor se interrumpe una vez finalizada la sesión de juego. En su lugar, se llamará a [ProcessEnding\(\)](#) para indicar que tanto la sesión de juego como el proceso del servidor están finalizando.

Sintaxis

```
FGameLiftGenericOutcome TerminateGameSession()
```


Parameters

Esta acción no tiene parámetros.

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

UpdatePlayerSessionCreationPolicy()

Actualiza la capacidad de la sesión de juego actual para aceptar sesiones de jugador nuevas. Una sesión de juego se puede configurar para que acepte o deniegue todas las sesiones nuevas de los jugadores. (Consulte también la acción [UpdateGameSession\(\)](#) en la Referencia de la API del servicio Amazon GameLift Servers).

Sintaxis

```
FGameLiftGenericOutcome UpdatePlayerSessionCreationPolicy(EPlayerSessionCreationPolicy policy)
```

Parameters

Política

Valor que indica si la sesión de juego acepta jugadores nuevos.

Tipo: EPlayerSessionCreationPolicy enum. Los valores válidos son:

- ACCEPT_ALL: se aceptan todas las sesiones de jugador nuevas.
- DENY_ALL: se rechazan todas las sesiones de jugador nuevas.

Obligatorio: sí

Valor devuelto

Devuelve un resultado genérico correcto o erróneo con un mensaje de error.

Eventos de ubicación de sesión de juego

Amazon GameLift Servers emite eventos para cada solicitud de ubicación de sesión de juego a medida que se procesa. Puede publicar estos eventos en un tema de Amazon SNS, como se

describe en [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#).

Estos eventos también se transmiten a Eventos de Amazon CloudWatch casi en tiempo real y en la medida en que sea posible.

En este tema se describe la estructura de los eventos de ubicación de las sesiones de juego y se proporciona un ejemplo para cada tipo de evento. Para obtener más información sobre el estado de las solicitudes de ubicación de sesiones de juego, consulte [GameSessionPlacement](#) en la Referencia de la API de Amazon GameLift Servers.

Sintaxis de eventos de ubicación

Los eventos se representan como objetos JSON. La estructura de eventos se ajusta al patrón de eventos de CloudWatch, con campos de nivel superior y detalles específicos del servicio similares.

Los campos de nivel superior incluyen lo siguiente (consulte el [patrón de eventos](#) para obtener más información):

versión

Este campo siempre está establecido en 0 (cero).

id

Un identificador de seguimiento único para el evento.

tipo-detalle

El valor siempre es `GameLift Queue Placement Event`.

origen

El valor siempre es `aws.gamelift`.

inscrita

La cuenta de AWS que se utiliza para administrar Amazon GameLift Servers.

hora

La marca temporal del evento.

region

La región de AWS en la que se está procesando la solicitud de ubicación. Esta es la región en la que se encuentra la cola de sesiones de juego en uso.

recursos

Valor ARN de la cola de sesiones de juego que está procesando la solicitud de ubicación.

PlacementFulfilled

La solicitud de ubicación se ha completado satisfactoriamente. Se ha iniciado una nueva sesión de juego y se han creado nuevas sesiones de jugadores para cada uno de los jugadores que figuran en la solicitud de ubicación de la sesión de juego. La información sobre la conexión de los jugadores está disponible.

Sintaxis detallada:

`placementId`

Un identificador único asignado a la solicitud de ubicación de sesión de juego.

`puerto`

El número de puerto de la nueva sesión de juego.

`gameSessionArn`

Un identificador de ARN de la nueva sesión de juego.

`ipAddress`

La dirección IP de la sesión de juego.

`dnsName`

El identificador de DNS asignado a la instancia que ejecuta la nueva sesión de juego. El formato del valor varía en función de si la instancia que ejecuta la sesión de juego está habilitada para TLS. Cuando se conecte a una sesión de juego que se ejecute en una flota habilitada de TLS, debe utilizar el nombre de DNS, no la dirección IP.

Flotas habilitadas para TLS: `<unique identifier>.<region identifier>.amazongamelift.com`.

Flotas no habilitadas para TLS: `ec2-<unique identifier>.compute.amazonaws.com`.

`startTime`

Marca de tiempo que indica cuándo se puso esta solicitud en la cola.

endTime

Marca de tiempo que indica cuándo se tramitó la solicitud.

gameSessionRegion

Región de AWS de la flota que aloja la sesión de juego. Esto corresponde al token de la región en `GameSessionArn`.

placedPlayerSessions

La colección de sesiones de jugador que se creó para cada jugador en la solicitud de ubicación de la sesión de juego.

Ejemplo

```
{
  "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 solicitud de ubicación se canceló con una llamada al servicio de GameLift [StopGameSessionPlacement](#).

Detalles:

placementId

Un identificador único asignado a la solicitud de ubicación de sesión de juego.

startTime

Marca de tiempo que indica cuándo se puso esta solicitud en la cola.

endTime

Marca de tiempo que indica cuándo se canceló la solicitud.

Ejemplo

```
{
  "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

La ubicación de la sesión de juego no se tramitó correctamente antes de que expirara el límite de tiempo de la cola. La solicitud de ubicación se puede volver a enviar según sea necesario.

Detalles:

placementId

Un identificador único asignado a la solicitud de ubicación de sesión de juego.

startTime

Marca de tiempo que indica cuándo se puso esta solicitud en la cola.

endTime

Marca de tiempo que indica cuándo se canceló la solicitud.

Ejemplo

```
{  
  "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 no ha podido tramitar la solicitud de sesión de juego. Por lo general, esto se debe a un error interno inesperado. La solicitud de ubicación se puede volver a enviar según sea necesario.

Detalles:

placementId

Un identificador único asignado a la solicitud de ubicación de sesión de juego.

startTime

Marca de tiempo que indica cuándo se puso esta solicitud en la cola.

endTime

Marca de tiempo que indica cuándo falló la solicitud.

Ejemplo

```
{
  "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"
  }
}
```

```
}

```

Versiones de AMI de Amazon GameLift Servers

En la siguiente tabla se identifican las últimas imágenes de máquinas de Amazon (AMIs) que se Amazon GameLift Servers utilizan para el alojamiento de EC2 gestionado. Como se describe en [Configuración y análisis de vulnerabilidades en Amazon GameLift Servers](#), debe crear periódicamente nuevas flotas de EC2 administradas por Amazon GameLift Servers para implementar las últimas actualizaciones de versiones de AMI.

AMIs para su uso con el SDK de servidor para Amazon GameLift Servers más de 5

Amazon GameLift Servers utiliza lo siguiente AMIs para alojar servidores de juegos que están integrados con el SDK del servidor de la Amazon GameLift Servers versión 5.

Imagen de Amazon GameLift Servers	Arquitectura	Última revisión	AWS Imagen base
Amazon Linux 2023 BASE_AMI_LINUX_2023	x86	2025-12-16	Amazon Linux 2023 AMI 2023.9.20 251208.0 x86_64 HVM kernel-6.1 (notas de la versión)
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_5	x86	16 de diciembre de 2025	Amazon Linux 2 Kernel 5.10 AMI 2.0.20251208.0 x86_64 HVM gp2 (notas de la versión)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_5	x86	16 de diciembre de 2025	Windows_Server-2016-English-Full-Base-2025.12.10 (notas de la versión)
Amazon Linux 2023 BASE_AMI_LINUX_2023_ARM	ARM64	2025-12-16	Amazon Linux 2023 AMI 2023.9.20 251208.0 arm64 HVM kernel-6.1 (notas de la versión)
Amazon Linux 2	ARM64	16 de diciembre de 2020	Amazon Linux 2 LTS Arm64 Kernel 5.10 AMI 2.0.20251208.0 arm64 HVM gp2 (notas de la versión)

Imagen de Amazon GameLift Servers	Arquitectura	Última revisión	AWS Imagen base
BASE_AMI_LINUX_2_ARM			
Windows 2022 BASE_AMI_WINDOWS_2022	x86	16 de diciembre de 2025	Windows_Server--13.English-Full-Base-2025.12.10 (notas de la versión)

AMIs para usar con el SDK de servidor para 4 Amazon GameLift Servers

Amazon GameLift Servers utiliza lo siguiente AMIs para alojar servidores de juegos que están integrados con el SDK del servidor de Amazon GameLift Servers la versión 4 o anterior.

Imagen de Amazon GameLift Servers	Arquitectura	Última revisión	AWS Imagen base
Amazon Linux 2 BASE_AMI_LINUX_2_SDK_4	x86	2025-12-16	Amazon Linux 2 AMI 2.0.20251208.0 x86_64 HVM gp2 (notas de la versión)
Amazon Linux 2023 BASE_AMI_LINUX_2023_SDK_4 Disponible únicamente en las regiones de China (Pekín) y China (Ningxia).	x86	16 de diciembre de 2025	Amazon Linux 2023 AMI 2023.9.20251208.0 x86_64 HVM kernel-6.1 (notas de la versión)
Windows 2016 BASE_AMI_WINDOWS_2016_SDK_4	x86	16 de diciembre de 2025	Windows_Server-2016-English-Full-Base-2025.12.10 (notas de la versión)

Para obtener más información, consulte los siguientes recursos:

- [Amazon Linux 2023 release notes](#)
- [Amazon Linux 2 release notes](#)
- [AWS Historial de versiones de AMI de Windows](#)
- [Description of Software Update Services and Windows Server Update Services changes in content for 2024](#)

Ubicaciones de servicio de Amazon GameLift Servers

Amazon GameLift Servers las funciones están disponibles en múltiples Zonas Regiones de AWS y Zonas Locales. Puede diseñar una solución de alojamiento que coloque a sus servidores de juegos allí donde están los jugadores.

AWS Ubicaciones compatibles

La siguiente tabla es una lista de Regiones de AWS las Zonas Locales que admiten Amazon GameLift Servers recursos. Indica los tipos de recursos que puede crear en cada ubicación.

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoSProtección S (puerta de enlace para jugadores)
Este de EE. UU. (Norte de Virginia)	us-east-1	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
Este de EE. UU. (Ohio)	us-east-2	Sí		Sí	Sí	Sí		

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoProtection S (puerta de enlace para jugadores)
Oeste de EE. UU. (Norte de California)	us-west-1	Sí		Sí	Sí	Sí		
Oeste de EE. UU. (Oregón)	us-west-2	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
África (Ciudad del Cabo)	af-south-1			Sí				
Asia-Pacífico (Tailandia)	ap-southeast-7			Sí				
Asia-Pacífico (Hong Kong)	ap-east-1			Sí				
Asia-Pacífico (Malasia)	ap-southeast-5			Sí				

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoSProtección S (puerta de enlace para jugadores)
Asia-Pacífico (Mumbai)	ap-south-1	Sí		Sí	Sí	Sí		
Asia-Pacífico (Osaka)	ap-northeast-3			Sí				
Asia-Pacífico (Seúl)	ap-northeast-2	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
Asia-Pacífico (Singapur)	ap-southeast-1	Sí		Sí	Sí	Sí		
Asia-Pacífico (Sídney)	ap-southeast-2	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
Asia-Pacífico (Tokio)	ap-northeast-1	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoProtection S (puerta de enlace para jugadores)
Canadá (centro)	ca-central-1	Sí		Sí	Sí	Sí		
Europa (Fráncfort)	eu-central-1	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
Europa (Irlanda)	eu-west-1	Sí	Sí	Sí	Sí	Sí	Sí	Sí [†]
Europa (Londres)	eu-west-2	Sí		Sí	Sí	Sí		
Europa (Milán)	eu-south-1			Sí				
Europa (París)	eu-west-3			Sí				
Europa (Estocolmo)	eu-north-1			Sí				
Medio Oriente (Baréin)	me-south-1			Sí				

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoProtection S (puerta de enlace para jugadores)
América del Sur (São Paulo)	sa-east-1	Sí		Sí	Sí	Sí		
Zona local de Atlanta	us-east-1-atl-1			Sí				
Zona local de Chicago	us-east-1-chi-1			Sí				
Zona local de Dallas *	us-east-1-dfw-1			Sí				
Zona local de Dallas 2	us-east-1-dfw-2			Sí				
Zona local de Denver	us-west-2-den-1			Sí				

Ubicación geográfica	Código de la ubicación	Región de origen para flotas administradas (ubicación única)	Región de origen para flotas administradas (múltiples ubicaciones)	Ubicación remota para flotas administradas (múltiples ubicaciones)	Flota de Anywhere	Cola de sesiones de juego	Emparejador de FlexMatch y conjunto de reglas	DDoSProtección S (puerta de enlace para jugadores)
Zona local de Houston	us-east-1-iah-1			Sí				
Zona local de Kansas City	us-east-1-mci-1			Sí				
Zona local de Los Ángeles	us-west-2-lax-1			Sí				
Zona local de Phoenix	us-west-2-phx-1			Sí				
Zona local de Lagos, Nigeria	af-south-1-los-1			Sí				

* Disponible para Cuentas de AWS quienes ya se hayan suscrito.

† Región de origen de la flota válida para la protección DDoS (puerta de entrada para jugadores).

 Note

Algunas Regiones de AWS y todas las Zonas Locales no están habilitadas de forma predeterminada para un Cuenta de AWS. Para poder implementar servidores de juegos en esas ubicaciones, primero debe suscribirse a estas regiones o zonas locales. Para obtener más información sobre las regiones que no están habilitadas de forma predeterminada y cómo habilitarlas, consulte [Administración de Regiones de AWS](#) en la Referencia general de AWS. Consulte [Introducción al uso de zonas locales](#) en la Guía del usuario de zonas locales de AWS. Las flotas creadas antes del 28 de febrero de 2022 no se verán afectadas por este requisito.

Además, debe actualizar la política de administrador de Amazon GameLift Servers de modo que la acción `ec2:DescribeRegions` esté permitida. Para ver un ejemplo de política con regiones que no están habilitadas de forma predeterminada, consulte [Ejemplos de permisos de administración](#).

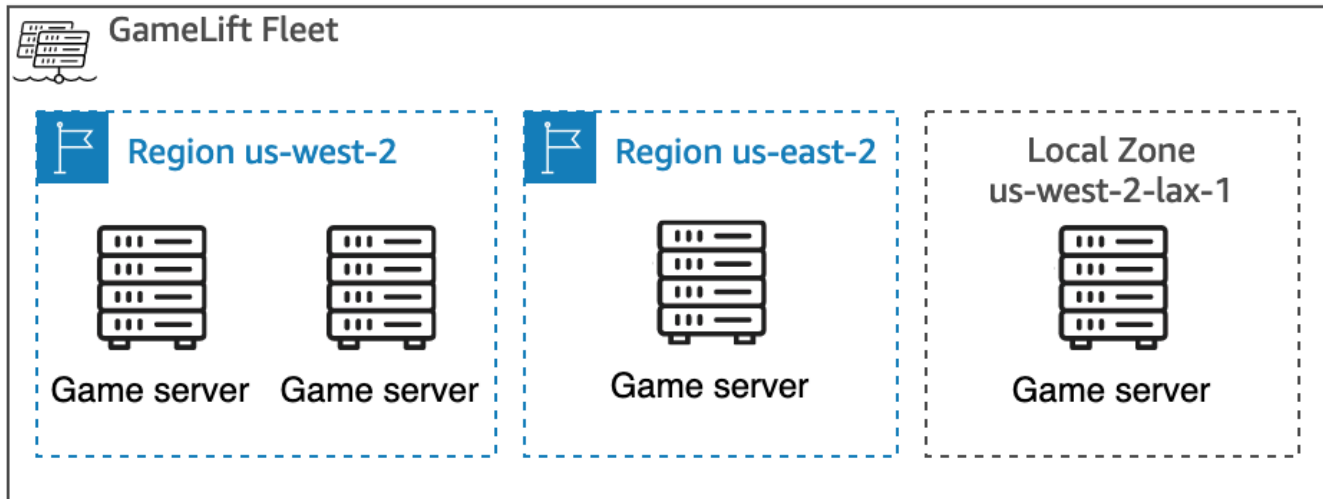
Ubicaciones para el alojamiento administrado

El alojamiento administrado de Amazon GameLift Servers implementa flotas de recursos de servidor de juegos. Cada flota se crea en una Región de AWS, que es la región de origen de la flota. Se hace referencia a la región de origen de una flota en el número de recurso de Amazon (ARN) de la flota.

Puede implementar una flota de una sola región y que los recursos de alojamiento se encuentren únicamente en la región de origen. Si lo prefiere, puede implementar una flota con múltiples ubicaciones y que los recursos de alojamiento se encuentren en varias ubicaciones geográficas. Una flota con múltiples ubicaciones tiene una región de origen y una o más ubicaciones remotas. No existen limitaciones en cuanto a la forma de agrupar las ubicaciones en una flota. Al gestionar la capacidad de alojamiento de una flota con múltiples ubicaciones, puede configurar la capacidad de cada ubicación individualmente.

Las ubicaciones remotas de una flota con múltiples ubicaciones pueden ser otras Regiones de AWS o Zonas Locales. Una zona local es una extensión de una Región de AWS. Le permite ubicar los recursos de computación más cerca de los usuarios para ofrecer una experiencia de juego con menor latencia. Para obtener más información, consulte [AWS Local Zones](#). El código de ubicación de una zona local es el código de su región principal, seguido de un identificador de ubicación física. Por ejemplo, el código para la zona local de Los Ángeles es `us-west-2-lax-1`.

El siguiente diagrama muestra una flota con múltiples ubicaciones y recursos en dos Regiones de AWS y una zona local. La región de origen de la flota es `us-west-2`, y tiene dos ubicaciones remotas: la región `us-east-2` y la zona local `us-west-2-lax-1`.



Además de los recursos de flota, el alojamiento administrado con Amazon GameLift Servers también puede utilizar los siguientes tipos de recursos. Estos recursos se crean de una forma específica Región de AWS que sea compatible con el tipo de recurso.

- **Compilación:** se trata de una compilación de servidor de juegos para ser alojada en una flota de EC2 administrada. Cree un recurso de compilación en la misma región que la flota en la que se implementará.
- **Script:** se trata de un script de configuración para alojar un juego con Amazon GameLift Servers Realtime. Cree un recurso de script en la misma región que la flota en la que se va a implementar.
- **Definición de grupo de contenedores:** se trata de una configuración para ejecutar contenedores en una flota de contenedores administrada. Identifica una o más imágenes de contenedor con un software de servidor de juegos para su implementación en la flota de contenedores. Cree una definición de grupo de contenedores en la misma región que la flota en la que van a implementar.
- **Cola de sesiones de juego:** este recurso procesa las solicitudes de sesiones de juego e inicia nuevas sesiones de juego. El procesamiento se lleva a cabo en el lugar Región de AWS donde se encuentra la cola. Para reducir la latencia en el proceso de ubicación de las sesiones de juego, cree una cola geográficamente próxima a los jugadores que la van a usar.

Ubicaciones para Amazon GameLift Servers Anywhere

Una flota de Amazon GameLift Servers Anywhere es una recopilación de hardware de alojamiento que usted aprovisiona. Usted administra toda la actividad de los recursos de alojamiento, lo que incluye implementar el software de servidor de juegos, mantenerlo actualizado e iniciar los procesos del servidor. Puede crear una flota de Anywhere para conectar el servicio Amazon GameLift Servers con sus recursos de alojamiento autoadministrados. Amazon GameLift Servers administra la ubicación de las sesiones de juego: procesa las solicitudes de incorporación de jugadores, localiza los recursos de alojamiento disponibles, inicia nuevas sesiones de juego y proporciona a los clientes de juego la información de conexión. Puede crear una flota de Anywhere en cualquier Regiones de AWS compatible.

Para añadir instancias de hardware de alojamiento a una flota de Anywhere, debe proceder a su registro. Cada instancia registrada debe tener asociada una ubicación personalizada. Las ubicaciones personalizadas no están relacionadas Regiones de AWS con las Zonas Locales. Se utilizan para representar la ubicación física del hardware.

Para obtener más información sobre cómo crear una flota de Anywhere y probar la integración del servidor de juegos, consulte [Creación de una flota de Amazon GameLift Servers Anywhere](#) y [Configuración de pruebas locales con Amazon GameLift Servers Anywhere](#).

Ubicaciones para Amazon GameLift Servers FlexMatch

Los recursos de FlexMatch se utilizan para procesar las solicitudes de emparejamiento de los jugadores. Incluyen un recurso de configuración de emparejamiento y un recurso de conjunto de reglas. El procesamiento se lleva a cabo en el lugar Región de AWS donde se encuentran FlexMatch los recursos. Para reducir la latencia en el proceso de emparejamiento, cree los recursos geográficamente cerca de los jugadores que los van a usar. La configuración de emparejamiento y el conjunto de reglas que utiliza deben estar ubicados en la misma Región de AWS. Puede crear FlexMatch recursos en cualquiera de los Regiones de AWS que los respalden.

Para obtener más información sobre cómo configurar FlexMatch para su solución de alojamiento, consulte la [Guía para desarrolladores de Amazon GameLift Servers FlexMatch](#).

Amazon GameLift Servers en China

Cuando utilice Amazon GameLift Servers para los recursos de la región de China (Pekín), a cargo de Sinnet, o la región de China (Ningxia), a cargo de NWCD, deberá tener una cuenta de AWS

(China) independiente. Debe tener en cuenta que algunas características no están disponibles en las regiones de China. Para obtener más información sobre el uso de Amazon GameLift Servers en esas regiones, consulte los siguientes recursos:

- [Amazon Web Services en China](#)
- [Amazon GameLift Servers](#) (Introducción a Amazon Web Services en China)

Cuotas y puntos de conexión de Amazon GameLift Servers

- Para saber qué puntos de conexión de servicio y qué puntos de conexión de baliza puede utilizar para conectarse al servicio mediante programación, consulte [Puntos de conexión del servicio de Amazon GameLift Servers](#).
- Para ver las cuotas de Amazon GameLift Servers sobre el uso de recursos u operaciones del servicio por cada cuenta de AWS, consulte [Cuotas de servicio de Amazon GameLift Servers](#). Para obtener más información sobre las cuotas y sobre cómo solicitar un aumento, consulte [Cuotas de servicio de AWS](#). Puede solicitar aumentos de cuota mediante la consola de Amazon GameLift Servers.

Límites de la API de Amazon GameLift Servers

En las tablas de esta sección se enumeran los límites de velocidad predeterminados para las operaciones de la API de Amazon GameLift Servers. Estos límites ayudan a mantener la estabilidad del servicio y a garantizar un uso razonable. Cada operación de la API está sujeta a unos límites específicos en función de su impacto en los recursos y de los patrones de uso.

De forma predeterminada, la limitación máxima de cada cuenta de cliente está establecido en 500 para las solicitudes de ráfagas y 500 para el tráfico normal. Además, cada API tiene su propio par de límites predeterminado, tal y como se indica a continuación.

Referencia de límites de la API

En la siguiente tabla se enumeran los límites de velocidad predeterminados para las operaciones de la API de Amazon GameLift Servers. APIs marcados con una limitación a nivel de recursos pueden estar sujetos a límites adicionales para evitar problemas con las teclas de acceso rápido de la base de datos.

Referencia de límites de la API

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
AcceptMatch	100	100	No	El límite es por cuenta.
ClaimGameServer	40	20	No	El límite es por cuenta.
CreateAlias	4	1	No	El límite es por cuenta.
CreateBuild	10	1	No	El límite es por cuenta.
CreateContainerFleet	4	1	No	El límite es por cuenta.
CreateContainerGroupDefinition	10	1	No	El límite es por cuenta.
CreateFleet	4	1	No	El límite es por cuenta.
CreateFleetLocations	4	1	No	El límite es por cuenta.
CreateGameServerGroup	5	0.1	No	El límite es por cuenta.
CreateGameSession	50	50	No	El límite es por cuenta.
CreateGameSessionQueue	4	1	No	El límite es por cuenta.
CreateLocation	4	1	No	El límite es por cuenta.
CreateMatchmakingConfiguration	4	1	No	El límite es por cuenta.
CreateMatchmakingRuleSet	4	1	No	El límite es por cuenta.
CreatePlayerSession	200	200	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
CreatePlayerSessions	200	200	No	El límite es por cuenta.
CreateScript	10	1	No	El límite es por cuenta.
CreateVpcPeeringAuthorization	1	0.1	No	El límite es por cuenta.
CreateVpcPeeringConnection	1	0.1	No	El límite es por cuenta.
DeleteAlias	4	1	No	El límite es por cuenta.
DeleteBuild	10	1	No	El límite es por cuenta.
DeleteContainerFleet	4	1	No	El límite es por cuenta.
DeleteContainerGroupDefinition	10	1	No	El límite es por cuenta.
DeleteFleet	4	1	No	El límite es por cuenta.
DeleteFleetLocations	4	1	No	El límite es por cuenta.
DeleteGameServerGroup	5	0.1	No	El límite es por cuenta.
DeleteGameSessionQueue	4	1	No	El límite es por cuenta.
DeleteLocation	4	1	No	El límite es por cuenta.
DeleteMatchmakingConfiguration	4	1	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
DeleteMatchmakingRuleSet	4	1	No	El límite es por cuenta.
DeleteScalingPolicy	4	1	No	El límite es por cuenta.
DeleteScript	10	1	No	El límite es por cuenta.
DeleteVpcPeeringAuthorization	1	0.1	No	El límite es por cuenta.
DeleteVpcPeeringConnection	1	0.1	No	El límite es por cuenta.
DeregisterGameServer	40	20	No	El límite es por cuenta.
DeregisterCompute	10	2	No	El límite es por cuenta.
DescribeAlias	10	2	No	El límite es por cuenta.
DescribeBuild	20	5	No	El límite es por cuenta.
DescribeCompute	10	2	No	El límite es por cuenta.
DescribeContainerFleet	20	5	No	El límite es por cuenta.
DescribeContainerGroupDefinition	20	5	No	El límite es por cuenta.
DescribeEC2InstanceLimits	40	15	No	El límite es por cuenta.
DescribeFleetAttributes	10	2	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
DescribeFleetCapacity	4	1	No	El límite es por cuenta.
DescribeFleetDeployment	20	5	No	El límite es por cuenta.
DescribeFleetEvents	10	5	No	El límite es por cuenta.
DescribeFleetLocationAttributes	10	2	No	El límite es por cuenta.
DescribeFleetLocationCapacity	10	5	No	El límite es por cuenta.
DescribeFleetLocationUtilization	10	2	No	El límite es por cuenta.
DescribeFleetPortSettings	10	2	No	El límite es por cuenta.
DescribeFleetUtilization	10	2	No	El límite es por cuenta.
DescribeGameServer	40	20	No	El límite es por cuenta.
DescribeGameServerGroup	10	2	No	El límite es por cuenta.
DescribeGameServerInstances	10	2	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
DescribeGameSessionDetails	10	5	No	El límite es por cuenta. Considere usarlo DescribeGameSessions en su lugar, ya que tiene un límite predeterminado más alto y admite límites de nivel de recursos.
DescribeGameSessionPlacement	75	75	Sí	El límite es por cuenta. Sujeto a limitaciones de nivel de recurso.
DescribeGameSessionQueues	10	2	No	El límite es por cuenta.
DescribeGameSessions	20	20	Sí	El límite es por cuenta. Sujeto a limitaciones de nivel de recurso.
DescribeInstances	10	2	No	El límite es por cuenta.
DescribeMatchmaking	200	200	Sí	El límite es por cuenta. Sujeto a limitaciones de nivel de recurso.
DescribeMatchmakingConfigurations	15	7	No	El límite es por cuenta.
DescribeMatchmakingRuleSets	20	10	No	El límite es por cuenta.
DescribePlayerSessions	200	200	Sí	El límite es por cuenta. Sujeto a limitaciones de nivel de recurso.
DescribeRuntimeConfiguration	4	1	No	El límite es por cuenta.
DescribeScalingPolicies	20	20	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
DescribeScript	20	5	No	El límite es por cuenta.
DescribeVpcPeering Authorizations	1	0.1	No	El límite es por cuenta.
DescribeVpcPeering Connections	1	0.1	No	El límite es por cuenta.
GetComputeAccess	4	1	No	El límite es por cuenta.
GetComputeAuthToken	10	5	No	El límite es por cuenta.
GetGameSessionLogUrl	4	1	No	El límite es por cuenta.
GetInstanceAccess	4	1	No	El límite es por cuenta.
ListAliases	10	2	No	El límite es por cuenta.
ListBuilds	20	5	No	El límite es por cuenta.
ListContainerFleets	20	5	No	El límite es por cuenta.
ListCompute	10	2	No	El límite es por cuenta.
ListContainerGroup Definitions	20	5	No	El límite es por cuenta.
ListContainerGroup DefinitionVersions	20	5	No	El límite es por cuenta.
ListFleets	10	2	No	El límite es por cuenta.
ListFleetDeployments	20	5	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
ListGameServerGroups	10	2	No	El límite es por cuenta.
ListGameServers	10	2	No	El límite es por cuenta.
ListLocations	10	2	No	El límite es por cuenta.
ListScripts	20	5	No	El límite es por cuenta.
ListTagsForResource	20	5	No	El límite es por cuenta.
PutScalingPolicy	4	1	No	El límite es por cuenta.
RegisterCompute	10	5	No	El límite es por cuenta.
RegisterGameServer	40	20	No	El límite es por cuenta.
RequestUploadCredentials	4	1	No	El límite es por cuenta.
ResolveAlias	4	1	No	El límite es por cuenta.
ResumeGameServerGroup	1	0.1	No	El límite es por cuenta.
SearchGameSessions	10	5	No	El límite es por cuenta.
StartFleetActions	1	0.1	No	El límite es por cuenta.
StartGameSessionPlacement	75	75	No	El límite es por cuenta.
StartMatchBackfill	70	70	No	El límite es por cuenta.
StartMatchmaking	100	100	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
StopFleetActions	1	0.1	No	El límite es por cuenta.
StopGameSessionPlacement	20	20	No	El límite es por cuenta.
StopMatchmaking	100	100	No	El límite es por cuenta.
SuspendGameServerGroup	1	0.1	No	El límite es por cuenta.
TagResource	20	5	No	El límite es por cuenta.
TerminateGameSession	10	2	No	El límite es por cuenta.
UntagResource	20	5	No	El límite es por cuenta.
UpdateAlias	4	1	No	El límite es por cuenta.
UpdateBuild	10	1	No	El límite es por cuenta.
UpdateContainerGroupDefinition	10	1	No	El límite es por cuenta.
UpdateContainerFleet	4	1	No	El límite es por cuenta.
UpdateFleetAttributes	4	1	No	El límite es por cuenta.
UpdateFleetCapacity	4	1	No	El límite es por cuenta.
UpdateFleetPortSettings	4	1	No	El límite es por cuenta.
UpdateGameServer	40	20	No	El límite es por cuenta.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Limitación de nivel de recurso	Notas
UpdateGameServerGroup	5	0.1	No	El límite es por cuenta.
UpdateGameSession	10	2	No	El límite es por cuenta.
UpdateGameSessionQueue	4	1	No	El límite es por cuenta.
UpdateMatchmakingConfiguration	4	1	No	El límite es por cuenta.
UpdateRuntimeConfiguration	4	1	No	El límite es por cuenta.
UpdateScript	10	1	No	El límite es por cuenta.
ValidateMatchmakingRuleSet	4	1	No	El límite es por cuenta.

Limitación de nivel de recurso

Algunas operaciones de la API están sujetas a limitaciones de nivel de recurso para evitar problemas con las teclas de acceso rápido de la base de datos. Si bien aumentan los límites de la API para estas operaciones, es posible que sigan experimentando limitaciones de recursos.

Las siguientes operaciones de la API están sujetas a limitaciones de nivel de recurso:

- DescribeGameSessionPlacement
- DescribeGameSessions
- DescribeMatchmaking
- DescribePlayerSessions

La limitación de nivel de recurso se aplica mediante teclas de limitación específicas que incluyen identificadores de recurso. Por ejemplo, `DescribeGameSessionPlacement` utiliza la tecla «`Operation:GameLift/DescribeGameSessionPlacement,aws-account-placement-id:`», que impone la limitación por combinación de cuenta e ID de ubicación. Esto evita que los clientes sobrecarguen el sistema con solicitudes de recursos específicos.

Límites de la API del SDK de Amazon GameLift Servers

En la siguiente tabla se muestran los límites de velocidad predeterminados de las operaciones de la API del SDK de Amazon GameLift Servers. Estos límites optimizan el rendimiento y evitan la contención de recursos en los entornos de servidores de juegos. Es importante comprender estos límites para lograr una integración eficiente del servidor con Amazon GameLift Servers.

Los límites del SDK del servidor se aplican mediante claves de limitación específicas que incluyen identificadores de cuenta y proceso. Algunos límites se fijan por jugador para evitar el uso abusivo por parte cada jugador.

Note

Esta tabla está sujeta a cambios. La fuente autorizada de los límites actuales es la configuración interna del SDC. Póngase en contacto con AWS Support para obtener la información más actualizada si la necesita.

Referencia de los límites de la API del SDK del servidor

Acción de la API	Límite de ráfaga	Límite de frecuencia	Tecla de limitación	Notas
<code>AcceptPlayerSession</code>	3	1	<code>aws-account-and-process-id-and-player-id</code>	El límite se fija por jugador y por proceso. No debería necesitar nunca un aumento del límite, ya que este se fija por jugador.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Tecla de limitación	Notas
RemovePlayerSession	3	1	aws-account-and-process-id-and-player-id	El límite se fija por jugador y por proceso. No debería necesitar nunca un aumento del límite, ya que este se fija por jugador.
ActivateGameSession	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
ActivateHostProcessV2	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
StartMatchBackfill	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
DescribePlayerSessions	10	5	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Tecla de limitación	Notas
GetComputeCertificate	3	0.1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
GetCustomRoleCredentials	3	0.1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
InitSDK	3	0.1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
RefreshHostProcess	3	0.1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
StopMatchmaking	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Tecla de limitación	Notas
ProcessEnding	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
ProcessReady	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
UpdateGameSession	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
UpdateHostProcess	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
GetGameSessionId	10	5	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.

Acción de la API	Límite de ráfaga	Límite de frecuencia	Tecla de limitación	Notas
GetTerminationTime	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.
UpdatePlayerSessionCreationPolicy	3	1	aws-account-and-process-id	El límite se fija por proceso del servidor. No debería necesitar nunca un aumento del límite, ya que este se fija por proceso.

Note

Estos son los límites predeterminados. Si está experimentando una limitación en estas API, es posible que se deba a una limitación de la cuenta. Para solicitar un aumento de límite, póngase en contacto con AWS Support.

Los límites por proceso y por jugador están diseñados para dar cabida a los patrones de uso normales y es poco frecuente que deban ajustarse. Si está experimentando alguna limitación con estas API, revise la implementación para identificar posibles oportunidades de optimización antes de solicitar un aumento de los límites.

También se pueden aplicar limitaciones de cuenta mediante estas reglas adicionales:

- SdkWebSocket/AccountLevelHighUsage,aws-account
- SdkWebSocket/AccountLevelThrottle,aws-account

Señalizadores de pings de UDP de Amazon GameLift Servers

Los señalizadores de pings de UDP proporcionan una forma de medir la latencia de la red entre los dispositivos de los jugadores y las ubicaciones de alojamiento de Amazon GameLift Servers.

Los señalizadores de pings le permiten recopilar datos de latencia precisos para tomar decisiones informadas sobre la ubicación de los servidores de juegos y mejorar el emparejamiento de jugadores en función de los requisitos de latencia.

Cómo funcionan los señalizadores de pings de UDP

Amazon GameLift Servers proporciona puntos de conexión de UDP fijos (señalizadores de pings) en cada ubicación de alojamiento en la que puede implementar servidores de juegos. Dado que la mayoría de los servidores de juegos se comunican mediante UDP, medir la latencia con señalizadores de pings de UDP ofrece resultados más precisos que cuando se utilizan pings de ICMP. Los dispositivos de red suelen gestionar los paquetes ICMP de forma diferente a los paquetes UDP, lo que puede provocar que las mediciones de latencia no reflejen el rendimiento real que experimentarán los jugadores.

Con los señalizadores de pings de UDP, el cliente de juego puede enviar mensajes UDP a estos puntos de conexión y recibir respuestas asíncronas, lo que le proporcionará mediciones de latencia que representen mejor las condiciones reales de tráfico del juego entre el dispositivo del jugador y las posibles ubicaciones de alojamiento. Los puntos de conexión son permanentes y siguen disponibles mientras Amazon GameLift Servers admita el alojamiento de juegos en esa ubicación.

Casos de uso habituales de los señalizadores de pings de UDP

Puede utilizar señalizadores de pings de UDP de diversas formas para optimizar la experiencia en red del juego.

Selección de ubicaciones de alojamiento óptimas

Recopile datos de latencia de diferentes regiones geográficas para identificar las mejores ubicaciones principales y de respaldo para alojar los servidores de juegos para su base de jugadores.

Ubicación de las sesiones de juego en función de la latencia de los jugadores

Incluya datos de latencia de los jugadores cuando solicite nuevas sesiones de juego para ayudar a elegir las ubicaciones que ofrezcan la experiencia con la latencia más baja.

Optimización del emparejamiento basada en la latencia

Proporcione los datos de latencia de los jugadores cuando solicite el emparejamiento para favorecer el emparejamiento de jugadores con perfiles de latencia similares y colocar las sesiones de juego en ubicaciones óptimas para los jugadores emparejados.

 Note

Al crear solicitudes de emparejamiento, no debe proporcionar información de latencia a ubicaciones en las que no tenga flotas. Si lo hace, Amazon GameLift Servers podría intentar colocar la sesión de juego en ubicaciones donde no haya capacidad de flota, lo que provocará errores en las solicitudes de emparejamiento.

Obtención de puntos de conexión de señalizador

Para recuperar la información de dominio y puerto de la baliza de ping para Amazon GameLift Servers las ubicaciones, utilice la operación [ListLocations](#) API. El conjunto de ubicaciones que devuelve esta API depende de la Región de AWS que especifique al llamarla (o de tu región predeterminada si no especificas ninguna). Llamada desde:

- Una región de origen de una flota que admite múltiples ubicaciones: la API devuelve información de todas las ubicaciones de alojamiento.
- Una región de origen de una flota que admite una única ubicación: la API devuelve la información de esa ubicación.

Tenga en cuenta que si llama a esta API desde una ubicación que solo pueda ser una ubicación remota de una flota con múltiples ubicaciones, la API devolverá un error porque ese tipo de ubicación no tiene un punto de conexión de servicio.

Consulte la tabla de ubicaciones compatibles en [AWS Ubicaciones compatibles](#) para identificar las regiones de origen que admiten flotas con una o múltiples ubicaciones.

Ejemplo

```
aws gamelift list-locations --region ap-northeast-2
```

Esta Región de AWS admite flotas con varias ubicaciones, por lo que se devolverán varias ubicaciones. En este ejemplo se muestra uno de los valores devueltos:

```
[...]  
{  
  "LocationName": "ap-northeast-1",
```

```
"PingBeacon": {  
  "UDPEndpoint": {  
    "Domain": "gamelift-ping.ap-northeast-1.api.aws",  
    "Port": 7770  
  }  
}
```

Important

Almacene en caché la información de señalizadores de pings en lugar de llamar a `ListLocations` antes de cada medición de latencia. La información de dominio y puerto es estática y la API no está diseñada para solicitudes de gran volumen.

Implementación de mediciones de latencia

Siga estas prácticas recomendadas cuando implemente mediciones de latencia utilizando señalizadores de pings de UDP:

1. Almacene la información de los señalizadores de pings utilizando uno de estos métodos:
 - Codifique los puntos de conexión en el cliente de juego.
 - Guarde en caché la información en el backend del juego.
 - Implemente un mecanismo de actualización periódica (diaria/semanal) para actualizar la información.
2. Envíe mensajes de ping de UDP:
 - Incluya la información que desee en el cuerpo del mensaje (pero no puede estar vacío) y asegúrese de que el tamaño de los mensajes no supere los 300 bytes.
 - Respete los siguientes límites de velocidad para cada ubicación:
 - 3 transacciones por segundo (TPS) por combinación única de puerto y dirección IP de remitente
 - 1000 TPS por dirección IP de remitente única
3. Calcule la latencia:
 - Envíe varios pings a cada ubicación para calcular la latencia media.
 - Considere la posibilidad de enviar pings simultáneos a varias ubicaciones para obtener resultados más rápidos.

- Utilice una lógica de reintento según sea necesario para enviar nuevos paquetes para cualquier paquete que no se haya devuelto en poco tiempo (normalmente de 1 a 3 segundos), ya que UDP no garantiza la entrega al 100%.
- Calcula la diferencia de tiempo entre el envío de un mensaje y la recepción de la respuesta.
- Si una gran parte de los envíos de mensajes UDP a una ubicación no obtienen una respuesta constante, calcula la latencia utilizando los pings ICMP a nuestros [puntos de conexión de servicio de Amazon GameLift Servers](#) estándar como alternativa.

Tip

Te recomendamos que incluyas el puerto 7770 siempre que documentes la lista de puertos que tus jugadores deben tener abiertos en su red local. Esta es otra razón por la que deberías disponer de una alternativa para medir la latencia (mediante ICMP, por ejemplo) en caso de que este puerto esté bloqueado.

Ejemplos de código

Estos son algunos ejemplos sencillos que muestran cómo enviar pings UDP y calcular la latencia.

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()
```


[illegible]

1	3.0.0	Python 3.6	Realtime
2	3.0.0	Python 3.6	Realtime
3	3.0.0	Python 3.6	Realtime
4	3.0.0	Python 3.6	Realtime
5	3.0.0	Python 3.6	Realtime
6	3.0.0	Python 3.6	Realtime
7	3.0.0	Python 3.6	Realtime
8	3.0.0	Python 3.6	Realtime
9	3.0.0	Python 3.6	Realtime
10	3.0.0	Python 3.6	Realtime
11	3.0.0	Python 3.6	Realtime
12	3.0.0	Python 3.6	Realtime
13	3.0.0	Python 3.6	Realtime
14	3.0.0	Python 3.6	Realtime
15	3.0.0	Python 3.6	Realtime
16	3.0.0	Python 3.6	Realtime
17	3.0.0	Python 3.6	Realtime
18	3.0.0	Python 3.6	Realtime
19	3.0.0	Python 3.6	Realtime
20	3.0.0	Python 3.6	Realtime
21	3.0.0	Python 3.6	Realtime
22	3.0.0	Python 3.6	Realtime
23	3.0.0	Python 3.6	Realtime
24	3.0.0	Python 3.6	Realtime
25	3.0.0	Python 3.6	Realtime
26	3.0.0	Python 3.6	Realtime
27	3.0.0	Python 3.6	Realtime
28	3.0.0	Python 3.6	Realtime
29	3.0.0	Python 3.6	Realtime
30	3.0.0	Python 3.6	Realtime
31	3.0.0	Python 3.6	Realtime
32	3.0.0	Python 3.6	Realtime
33	3.0.0	Python 3.6	Realtime
34	3.0.0	Python 3.6	Realtime
35	3.0.0	Python 3.6	Realtime
36	3.0.0	Python 3.6	Realtime
37	3.0.0	Python 3.6	Realtime
38	3.0.0	Python 3.6	Realtime
39	3.0.0	Python 3.6	Realtime
40	3.0.0	Python 3.6	Realtime
41	3.0.0	Python 3.6	Realtime
42	3.0.0	Python 3.6	Realtime
43	3.0.0	Python 3.6	Realtime
44	3.0.0	Python 3.6	Realtime
45	3.0.0	Python 3.6	Realtime
46	3.0.0	Python 3.6	Realtime
47	3.0.0	Python 3.6	Realtime
48	3.0.0	Python 3.6	Realtime
49	3.0.0	Python 3.6	Realtime
50	3.0.0	Python 3.6	Realtime
51	3.0.0	Python 3.6	Realtime
52	3.0.0	Python 3.6	Realtime
53	3.0.0	Python 3.6	Realtime
54	3.0.0	Python 3.6	Realtime
55	3.0.0	Python 3.6	Realtime
56	3.0.0	Python 3.6	Realtime
57	3.0.0	Python 3.6	Realtime
58	3.0.0	Python 3.6	Realtime
59	3.0.0	Python 3.6	Realtime
60	3.0.0	Python 3.6	Realtime
61	3.0.0	Python 3.6	Realtime
62	3.0.0	Python 3.6	Realtime
63	3.0.0	Python 3.6	Realtime
64	3.0.0	Python 3.6	Realtime
65	3.0.0	Python 3.6	Realtime
66	3.0.0	Python 3.6	Realtime
67	3.0.0	Python 3.6	Realtime
68	3.0.0	Python 3.6	Realtime
69	3.0.0	Python 3.6	Realtime
70	3.0.0	Python 3.6	Realtime
71	3.0.0	Python 3.6	Realtime
72	3.0.0	Python 3.6	Realtime
73	3.0.0	Python 3.6	Realtime
74	3.0.0	Python 3.6	Realtime
75	3.0.0	Python 3.6	Realtime
76	3.0.0	Python 3.6	Realtime
77	3.0.0	Python 3.6	Realtime
78	3.0.0	Python 3.6	Realtime
79	3.0.0	Python 3.6	Realtime
80	3.0.0	Python 3.6	Realtime
81	3.0.0	Python 3.6	Realtime
82	3.0.0	Python 3.6	Realtime
83	3.0.0	Python 3.6	Realtime
84	3.0.0	Python 3.6	Realtime
85	3.0.0	Python 3.6	Realtime
86	3.0.0	Python 3.6	Realtime
87	3.0.0	Python 3.6	Realtime
88	3.0.0	Python 3.6	Realtime
89	3.0.0	Python 3.6	Realtime
90	3.0.0	Python 3.6	Realtime
91	3.0.0	Python 3.6	Realtime
92	3.0.0	Python 3.6	Realtime
93	3.0.0	Python 3.6	Realtime
94	3.0.0	Python 3.6	Realtime
95	3.0.0	Python 3.6	Realtime
96	3.0.0	Python 3.6	Realtime
97	3.0.0	Python 3.6	Realtime
98	3.0.0	Python 3.6	Realtime
99	3.0.0	Python 3.6	Realtime
100	3.0.0	Python 3.6	Realtime

1	1936	Implement
2	1936	File
3	1936	Application
4	1936	Unit
5	1936	Realtime
6	1936	Physical
7	1936	CC
8	1936	C#
9	1936	++)
10	1936	+
11	1936	
12	1936	
13	1936	
14	1936	
15	1936	
16	1936	
17	1936	
18	1936	
19	1936	
20	1936	
21	1936	
22	1936	
23	1936	
24	1936	
25	1936	
26	1936	
27	1936	
28	1936	
29	1936	
30	1936	
31	1936	
32	1936	
33	1936	
34	1936	
35	1936	
36	1936	
37	1936	
38	1936	
39	1936	
40	1936	
41	1936	
42	1936	
43	1936	
44	1936	
45	1936	
46	1936	
47	1936	
48	1936	
49	1936	
50	1936	
51	1936	
52	1936	
53	1936	
54	1936	
55	1936	
56	1936	
57	1936	
58	1936	
59	1936	
60	1936	
61	1936	
62	1936	
63	1936	
64	1936	
65	1936	
66	1936	
67	1936	
68	1936	
69	1936	
70	1936	
71	1936	
72	1936	
73	1936	
74	1936	
75	1936	
76	1936	
77	1936	
78	1936	
79	1936	
80	1936	
81	1936	
82	1936	
83	1936	
84	1936	
85	1936	
86	1936	
87	1936	
88	1936	
89	1936	
90	1936	
91	1936	
92	1936	
93	1936	
94	1936	
95	1936	
96	1936	
97	1936	
98	1936	
99	1936	
100	1936	

[illegible]

50000-2
lib
posterior

50000-1
lib
posterior

50000-0
lib
posterior

50000-3
lib
posterior

50000-2-0
lib
posterior

50000-2-1
lib
posterior

50000-2-2
lib
posterior

50000-2-3
lib
posterior

13336	6	mem	
13337	6	le	
13338	6	le	
13339	6	le	
13340	6	le	
13341	6	le	
13342	6	le	
13343	6	le	
13344	6	le	
13345	6	le	
13346	6	le	
13347	6	le	
13348	6	le	
13349	6	le	
13350	6	le	
13351	6	le	
13352	6	le	
13353	6	le	
13354	6	le	
13355	6	le	
13356	6	le	
13357	6	le	
13358	6	le	
13359	6	le	
13360	6	le	
13361	6	le	
13362	6	le	
13363	6	le	
13364	6	le	
13365	6	le	
13366	6	le	
13367	6	le	
13368	6	le	
13369	6	le	
13370	6	le	
13371	6	le	
13372	6	le	
13373	6	le	
13374	6	le	
13375	6	le	
13376	6	le	
13377	6	le	
13378	6	le	
13379	6	le	
13380	6	le	
13381	6	le	
13382	6	le	
13383	6	le	
13384	6	le	
13385	6	le	
13386	6	le	
13387	6	le	
13388	6	le	
13389	6	le	
13390	6	le	
13391	6	le	
13392	6	le	
13393	6	le	
13394	6	le	
13395	6	le	
13396	6	le	
13397	6	le	
13398	6	le	
13399	6	le	
13400	6	le	
13401	6	le	
13402	6	le	
13403	6	le	
13404	6	le	
13405	6	le	
13406	6	le	
13407	6	le	
13408	6	le	
13409	6	le	
13410	6	le	
13411	6	le	
13412	6	le	
13413	6	le	
13414	6	le	
13415	6	le	
13416	6	le	
13417	6	le	
13418	6	le	
13419	6	le	
13420	6	le	
13421	6	le	
13422	6	le	
13423	6	le	
13424	6	le	
13425	6	le	
13426	6	le	
13427	6	le	
13428	6	le	
13429	6	le	
13430	6	le	
13431	6	le	
13432	6	le	
13433	6	le	
13434	6	le	
13435	6	le	
13436	6	le	
13437	6	le	
13438	6	le	
13439	6	le	
13440	6	le	
13441	6	le	
13442	6	le	
13443	6	le	
13444	6	le	
13445	6	le	
13446	6	le	
13447	6	le	
13448	6	le	
13449	6	le	
13450	6	le	
13451	6	le	
13452	6	le	
13453	6	le	
13454	6	le	
13455	6	le	
13456	6	le	
13457	6	le	
13458	6	le	
13459	6	le	
13460	6	le	
13461	6	le	
13462	6	le	

3336	Implement	
3337	File	
3338	parallel	
3339	initial	
3340	Realtime	
3341	Physical	
3342	the C	
3343	C#	
3344	++)	
3345	+	
3346	3330-2	
3347	posterior	
3348	3330-1	
3349	posterior	
3350	3330-1	
3351	posterior	
3352	3330-2	
3353	posterior	
3354	3330-02	
3355	posterior	
3356	3330-0	
3357	posterior	

336

6

Implement

337

6

File

338

6

parallel

339

6

initial

340

6

Unit

341

6

Realtime

342

6

Byte

343

6

C

344

6

C#

345

6

++)

346

6

+

347

6

348

6

349

6

350

6

351

6

352

6

353

6

354

6

355

6

356

6

357

6

358

6

359

6

360

6

361

6

362

6

363

6

364

6

365

6

366

6

367

6

368

6

369

6

370

6

371

6

372

6

373

6

374

6

375

6

376

6

377

6

378

6

379

6

380

6

381

6

382

6

383

6

384

6

385

6

386

6

387

6

388

6

389

6

390

6

391

6

392

6

393

6

394

6

395

6

396

6

397

6

398

6

399

6

400

6

401

6

402

6

403

6

404

6

405

6

406

6

407

6

408

6

409

6

410

6

411

6

412

6

413

6

414

6

415

6

416

6

417

6

418

6

419

6

420

6

421

6

422

6

423

6

424

6

425

6

426

6

427

6

428

6

429

6

430

6

431

6

432

6

433

6

434

6

435

6

436

6

437

6

438

6

439

6

440

6

441

6

442

6

443

6

444

6

445

6

446

6

447

6

448

6

449

6

450

6

451

6

452

6

453

6

454

6

455

6

456

6

457

6

458

6

459

6

460

6

461

6

462

6

463

6

464

6

465

6

466

6

467

6

468

6

469

6

470

6

471

6

472

6

473

6

474

6

475

6

476

6

477

6

478

6

479

6

480

6

481

6

482

6

483

6

484

6

485

6

486

6

487

6

488

6

489

6

490

6

491

6

492

6

493

6

494

6

495

6

496

6

497

6

498

6

499

6

500

6

501

6

502

6

503

6

504

[illegible]

1.18.0	Realtime
1.17.0	Realtime
1.16.0	Realtime
1.15.0	Realtime
1.14.0	Realtime
1.13.0	Realtime
1.12.0	Realtime
1.11.0	Realtime
1.10.0	Realtime
1.9.0	Realtime
1.8.0	Realtime
1.7.0	Realtime
1.6.0	Realtime
1.5.0	Realtime
1.4.0	Realtime
1.3.0	Realtime
1.2.0	Realtime
1.1.0	Realtime
1.0.0	Realtime
0.9.0	Realtime
0.8.0	Realtime
0.7.0	Realtime
0.6.0	Realtime
0.5.0	Realtime
0.4.0	Realtime
0.3.0	Realtime
0.2.0	Realtime
0.1.0	Realtime
0.0.0	Realtime

3.10.1	201	Realtime
3.10.0	201	Realtime
3.9.1	201	Realtime
3.9.0	201	Realtime
3.8.1	201	Realtime
3.8.0	201	Realtime
3.7.1	201	Realtime
3.7.0	201	Realtime
3.6.1	201	Realtime
3.6.0	201	Realtime
3.5.1	201	Realtime
3.5.0	201	Realtime
3.4.1	201	Realtime
3.4.0	201	Realtime
3.3.1	201	Realtime
3.3.0	201	Realtime
3.2.1	201	Realtime
3.2.0	201	Realtime
3.1.1	201	Realtime
3.1.0	201	Realtime
3.0.1	201	Realtime
3.0.0	201	Realtime
2.9.1	201	Realtime
2.9.0	201	Realtime
2.8.1	201	Realtime
2.8.0	201	Realtime
2.7.1	201	Realtime
2.7.0	201	Realtime
2.6.1	201	Realtime
2.6.0	201	Realtime
2.5.1	201	Realtime
2.5.0	201	Realtime
2.4.1	201	Realtime
2.4.0	201	Realtime
2.3.1	201	Realtime
2.3.0	201	Realtime
2.2.1	201	Realtime
2.2.0	201	Realtime
2.1.1	201	Realtime
2.1.0	201	Realtime
2.0.1	201	Realtime
2.0.0	201	Realtime
1.9.1	201	Realtime
1.9.0	201	Realtime
1.8.1	201	Realtime
1.8.0	201	Realtime
1.7.1	201	Realtime
1.7.0	201	Realtime
1.6.1	201	Realtime
1.6.0	201	Realtime
1.5.1	201	Realtime
1.5.0	201	Realtime
1.4.1	201	Realtime
1.4.0	201	Realtime
1.3.1	201	Realtime
1.3.0	201	Realtime
1.2.1	201	Realtime
1.2.0	201	Realtime
1.1.1	201	Realtime
1.1.0	201	Realtime
1.0.1	201	Realtime
1.0.0	201	Realtime

Notas de la versión

Las siguientes notas de la versión están en orden cronológico, con las últimas actualizaciones en primer lugar. En 2016 se lanzó por primera vez Amazon GameLift Servers. Para ver las notas de la

versión anteriores a las indicadas aquí, consulte los enlaces de fecha de lanzamiento en [Versiones del SDK](#).

4 de marzo de 2026: Amazon GameLift Servers ahora ofrece protección DDoS para EC2 y flotas de contenedores basadas en Linux en SDKv5

Amazon GameLift Servers ahora ofrece protección distributed-denial-of-service (DDoS) para EC2 basados en Linux y Container Fleets on. SDKv5 La red de retransmisión proxy Player Gateway está ubicada en el mismo lugar que las instancias del servidor de juegos, lo que permite validar el tráfico, limitar la velocidad por jugador y ocultar la dirección IP del servidor de juegos, todo ello con una latencia adicional insignificante y sin coste adicional. Esta función está disponible inicialmente en las siguientes regiones: EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Europa (Fráncfort), Europa (Irlanda), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Asia Pacífico (Seúl). Para obtener más información sobre las ubicaciones de servicio, consulte [Ubicaciones de servicio de Amazon GameLift Servers](#).

Más información:

- [Entrada de blog: Presentamos Amazon GameLift Servers DDo la protección S](#)
- [DDoProtección S con Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Cómo funciona Player Gateway](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integre la puerta de enlace del jugador en un juego](#), Guía para desarrolladores de Amazon GameLift Servers
- [SDK de cliente para C++](#)
- [SDK de cliente para Unreal](#). El acceso requiere ser miembro de la organización de Epic Games a partir de GitHub. Consulta [Unreal Engine en GitHub](#) para obtener más información.

24 de febrero de 2026: los Amazon GameLift Servers complementos ahora son compatibles con Unity 6.3 y Unreal Engine 5.7

Versiones de complementos actualizadas:

- [Plugin para Unity](#), versión 3.2.1 (incluye el SDK de servidor para Unity, versión 5.4.0)
- [Plugin para Unreal](#), versión 3.1.1 (incluye el SDK de servidor para Unreal, versión 5.4.0)

Amazon GameLift Servers lanza nuevas versiones del SDK del plugin y del servidor para Unreal Engine y Unity. Estas versiones más recientes incluyen las actualizaciones siguientes:

- Ahora admiten su uso con Unreal Engine 5.7 y Unity 6.3.

Más información:

- [Integración de juegos con el complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integración de juegos con el Amazon GameLift Servers complemento para Unity](#), guía para Amazon GameLift Servers desarrolladores
- [Amazon GameLift ServersIntégralo en un proyecto de Unity Engine](#), guía para Amazon GameLift Servers desarrolladores

29 de enero de 2026: Amazon GameLift Servers ahora es compatible con el escalado de instancias to/from cero de Fleets

Amazon GameLift Servers ahora admite el escalado automático desde y hacia cero instancias en función de la actividad de la sesión de juego. Las flotas se reducen a cero después de un período definido sin actividad en ninguna sesión de juego y se amplían desde cero cuando se solicitan sesiones de juego, lo que ofrece una opción para optimizar los costes.

Enlaces de referencia de API de Amazon GameLift Servers:

- [AWS Acción del SDK UpdateFleetCapacity](#)

Más información:

- [Gestione el escalado de un Amazon GameLift Servers Fleet To/From Zero](#)

22 de enero de 2026: Amazon GameLift Servers Realtime ahora es compatible con Node.js 24.x en Amazon Linux 2023

Amazon GameLift Servers Realtime ahora es compatible con Node.js 24.x en Amazon Linux 2023. Esta actualización proporciona acceso a las últimas funciones y mejoras de seguridad de Node.js mientras se ejecuta en el moderno sistema operativo Amazon Linux 2023.

Ahora también puedes incluir un script de instalación opcional al crear una Realtime flota, lo que te permitirá instalar el software durante el inicio de la instancia según tus requisitos específicos. Para usar la nueva versión en tiempo de ejecución, especifique Node.js 24.x al crear los Realtime scripts.

Enlaces de referencia de API de Amazon GameLift Servers:

- [AWS Acción del SDK CreateScript](#)

Más información:

- [Versiones de ejecución de Node.js](#)
- [Agregue un script de instalación](#)

9 de diciembre de 2025: Amazon GameLift Servers ahora ofrece asistencia con tecnología de inteligencia artificial en la consola web

Amazon GameLift Servers ahora ofrece asistencia basada en inteligencia artificial en la consola web, lo que permite a Amazon Q Developer ofrecer orientación personalizada a los desarrolladores de juegos. Esta nueva función integra Amazon GameLift Servers conocimientos especializados para ayudar a los clientes a gestionar flujos de trabajo complejos, solucionar problemas y optimizar las implementaciones de sus servidores de juegos de manera más eficiente.

Esta nueva función proporciona recomendaciones asistidas por la IA para la integración de los servidores de juegos, la configuración de la flota y la optimización del rendimiento directamente desde la consola. Amazon GameLift Servers Esta mejora tiene como objetivo agilizar los procesos de toma de decisiones, reducir el tiempo de resolución de problemas y mejorar la utilización general de los recursos, lo que se traduce en ahorros de costes y en una mejor experiencia para los jugadores.

Más información:

- [Consola de Amazon GameLift Servers](#)

6 de noviembre de 2025: Amazon GameLift Servers ahora admite las compilaciones de juegos que utilizan el sistema operativo Windows Server 2022

Los clientes de Amazon GameLift Servers ahora pueden utilizar el sistema operativo Windows Server 2022 para alojar sus servidores de juegos. Windows Server 2022 ofrece varias mejoras con respecto a Windows Server 2016, incluida la seguridad. Este sistema operativo está disponible en todas las regiones de AWS, excepto las regiones de China.

Los clientes pueden utilizar sistemas operativos Windows más recientes y seguir recibiendo actualizaciones de seguridad críticas cuando finalice el soporte para Windows Server 2016 en enero de 2027. El soporte para Windows Server 2022 continuará hasta el 13 de octubre de 2031.

Enlaces de referencia de API de Amazon GameLift Servers:

- [AWS Acción del SDK CreateBuild](#)
- [Comando de la CLI upload-build](#)
- [Comando de la CLI create-build](#)

28 de octubre de 2025: Amazon GameLift Servers agrega el recopilador de métricas OpenTelemetry (OTel) a todos los servidores SDKs y complementos

Amazon GameLift Servers ahora incluye la recopilación de métricas OpenTelemetry (OTel) integrada en todos los servidores SDKs y complementos, lo que proporciona una observabilidad estandarizada para monitorear el rendimiento de los servidores de juegos. Esta integración incorpora capacidades de telemetría estándar del sector directamente a sus servidores de juegos, permitiendo una supervisión y solución de problemas integrales sin necesidad de instrumentación o configuración adicionales.

Con OpenTelemetry la integración, ahora puedes:

- Registrar automáticamente métricas, registros y rastros esenciales del servidor de juegos mediante protocolos estandarizados
- Habilitar una integración total con plataformas de supervisión y herramientas de observabilidad existentes
- Supervisar el rendimiento y el estado de los servidores de juegos en tiempo real en todas las instancias de flota

- Solucionar problemas más rápidamente con datos de telemetría que funcionan en diferentes soluciones de supervisión

OpenTelemetry es un marco de observabilidad de código abierto que proporciona una forma estandarizada de recopilar, procesar y exportar datos de telemetría.

Más información:

- [Supervisión con métricas de telemetría de servidor](#), Guía para desarrolladores de Amazon GameLift Servers

2 de octubre de 2025: Amazon GameLift Servers añade el acceso remoto basado en consola a las instancias de flota

Amazon GameLift Servers ahora ofrece acceso remoto directo a instancias de EC2 administradas e instancias de flota de contenedores a través de la consola de Amazon GameLift Servers, lo que elimina la necesidad de administrar las autenticaciones. Esta nueva característica utiliza Amazon EC2 Systems Manager (SSM) para proporcionar un acceso seguro a los terminales desde un explorador sin necesidad de configuración adicional o administración de credenciales.

Con el acceso remoto basado en consola, ahora puede:

- Solucionar problemas con la integración del servidor de juegos en tiempo real
- Supervisar la actividad del servidor de juegos y acceder a los archivos de registro durante las sesiones de juego activas
- Ejecutar herramientas de diagnóstico y evaluación comparativa utilizando el tráfico real de jugadores

Esta característica está disponible para flotas que ejecutan la versión 5.x del SDK del servidor. Para las flotas que ejecutan versiones anteriores del SDK, puede seguir utilizando el método AWS CLI existente para el acceso remoto.

Para conectarse de forma remota a una instancia de flota, abra la consola de Amazon GameLift Servers y busque la flota de contenedores o EC2 gestionada a la que desea acceder. En los detalles de la flota, abra la pestaña Instancias, seleccione una instancia de la lista y elija Conectar.

Más información:

- [Conexión remota a instancias de flota de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers

23 de septiembre de 2025: Amazon GameLift Servers añade soporte de nueva zona local en Dallas, Texas

Con el alojamiento administrado por Amazon GameLift Servers, ahora puede implementar recursos de servidor de juegos en una nueva zona local Dallas, TX (us-east-1-dfw-2). Las zonas locales le ofrecen la capacidad de ubicar los servidores de juegos geográficamente más cerca de los jugadores a fin de reducir la latencia y aumentar la jugabilidad de forma considerable. En esta nueva zona local, puede utilizar los siguientes tipos de instancias de EC2: C6gn, C6i, C6in, M6g, M6i, M6in, M8g y R6i. Para obtener descripciones detalladas de cada tipo de instancia, consulte [Tipos de instancias de EC2 de AWS](#).

Para empezar a alojar sesiones de juego en la nueva zona local de Dallas, primero suscríbase a la zona local para su Cuenta de AWS. A continuación, puede añadirla como ubicación remota a una flota con múltiples ubicaciones nueva o existente. Si el juego usa Amazon GameLift Servers FlexMatch, actualice cualquiera de las flotas de la cola de emparejamiento para incluir la nueva zona local. Con flotas multiubicación, puede administrar directamente la capacidad de alojamiento en cada ubicación.

La matriz Región de AWS de la nueva zona local de Dallas, Texas, es us-east-1 (Virginia). La zona local original Dallas, TX (us-east-1-dfw-1) ya no acepta nuevas solicitudes de suscripción.

Más información:

- [Cómo empezar Zonas locales de AWS](#), Guía Zonas locales de AWS del usuario
- [Ubicaciones de servicio de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Actualización de ubicaciones de la flota](#), Guía para desarrolladores de Amazon GameLift Servers

12 de agosto de 2025: el complemento de Amazon GameLift Servers para Unreal Engine ahora es compatible con la versión 5.6 de UE

Versión actualizada del complemento:

- [Complemento para Unreal](#), versión 3.0.1 (incluye el SDK del servidor para Unreal, versión 5.3.1)

Amazon GameLift Servers publica nuevas versiones del complemento para Unreal Engine y del SDK del servidor para Unreal Engine. Estas versiones más recientes incluyen las actualizaciones siguientes:

- Ahora se pueden utilizar con Unreal Engine 5.6.
- Se han resuelto los problemas de compilación al empaquetar los destinos de clientes de Android.

Más información:

- [Integración de juegos con el complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers

24 de junio de 2025: Amazon GameLift Servers lanza señalizadores de pings de UDP para ayudar a medir la latencia de la red en tiempo real para los juegos multijugador

Versiones actualizadas del SDK:

- SDK de AWS 1.11.595

Amazon GameLift Servers lanza para su disponibilidad general un conjunto de puntos de conexión fijos denominados señalizadores de pings de UDP para ayudarle a medir con precisión la latencia entre los dispositivos de los jugadores y las ubicaciones de los servidores de juegos. Los puntos de conexión de señalizador de pings de UDP están disponibles en todas las regiones globales y zonas locales de AWS admitidas por Amazon GameLift Servers, a excepción de las regiones de China de AWS .

La mayoría de los juegos multijugador utilizan el UDP (protocolo de datagramas de usuario) como protocolo principal de transmisión de paquetes debido a sus ventajas de rendimiento para los juegos en tiempo real. Comprender y optimizar la latencia de la red es crucial para ofrecer la mejor experiencia posible a los jugadores. Las balizas de ping UDP proporcionan una forma coherente y fiable de medir la latencia real de los paquetes UDP entre los jugadores y los servidores del juego, lo

que te ayuda a tomar mejores decisiones sobre las player-to-server coincidencias y la ubicación de las sesiones de juego.

La [ListLocations](#) API se ha ampliado para incluir información sobre el dominio y el puerto de los terminales como parte de la lista de ubicaciones que devuelve, lo que facilita el acceso programático a los puntos finales.

El cliente de juego puede enviar mensajes UDP a estos puntos de conexión y recibir respuestas asíncronas con los mismos datos, lo que le proporcionará mediciones de latencia que ilustren mejor las condiciones reales de tráfico del juego entre el dispositivo del jugador y las posibles ubicaciones de alojamiento. Estos puntos de conexión son permanentes y siguen disponibles mientras Amazon GameLift Servers admita el alojamiento de juegos en esa ubicación.

Más información:

- [Señalizadores de pings de UDP](#), Guía para desarrolladores de Amazon GameLift Servers
- [ListLocations](#), Referencia de la API Amazon GameLift Servers

24 de junio de 2025: Amazon GameLift Servers añade soporte de ubicación para Tailandia y Malasia

Ahora, con el alojamiento administrado por Amazon GameLift Servers, puede implementar recursos de servidores de juegos en Bangkok (Tailandia) y Kuala Lumpur (Malasia), de modo que pueda ampliar el alcance de sus juegos a todo el sudeste asiático. Estas nuevas regiones ayudan a reducir la latencia y a mejorar la experiencia de juego de los jugadores de esas áreas.

Las siguientes Regiones de AWS están disponibles como ubicaciones remotas para flotas con múltiples ubicaciones. A fin de empezar inmediatamente a alojar sesiones de juego en estas ubicaciones, añádelas como ubicaciones remotas a una flota con múltiples ubicaciones nueva o existente. Con flotas multiubicación, puede administrar directamente la capacidad de alojamiento en cada ubicación.

- Asia Pacífico (Tailandia) `ap-southeast-7` ()
- Asia Pacífico (Malasia) `ap-southeast-5` ()

No Regiones de AWS están habilitadas de forma predeterminada para una AWS cuenta. Debe suscribirse a cada región antes de poder implementar recursos de Amazon GameLift Servers en ella.

Más información:

- [Ubicaciones de servicio de Amazon GameLift Servers](#), Guía para desarrolladores de alojamiento de Amazon GameLift Servers
- [Actualización de ubicaciones de la flota](#), Guía para desarrolladores de alojamiento de Amazon GameLift Servers
- [Habilite o deshabilite AWS las regiones de su AWS cuenta](#), Guía de referencia de administración de cuentas.

29 de mayo de 2025: Amazon GameLift Servers publica nuevas versiones del SDK del servidor y los complementos para Unreal Engine y Unity

Versiones actualizadas del SDK:

- [SDK del servidor de C++](#), versión 5.3.0
- [SDK del servidor de C#](#), versión 5.3.0
- [SDK del servidor de Go](#), versión 5.3.0
- [SDK del servidor para Unreal](#), versión 5.3.0
- [SDK del servidor para Unity](#), versión 5.3.0

Versiones de complementos actualizadas:

- [Complemento para Unreal](#), versión 3.0.0
- [Complemento para Unity](#), versión 3.1.0

Las nuevas versiones del SDK del servidor para C++, C#, Go, Unreal y Unity, así como las nuevas versiones de complementos para Unreal Engine y Unity, ahora son de código abierto. Todas están disponibles en la [Amazon GameLift Servers GitHub organización](#). Para obtener información detallada sobre las actualizaciones, consulte las notas de la versión y los archivos readme de cada repositorio.

Actualizaciones clave del SDK del servidor:

- Se mejoraron la validación del lado del cliente y las respuestas a los errores en todos los servidores. SDKs
- La función de [OnProcessTerminate](#) devolución de llamada ahora tiene una lógica predeterminada para finalizar el proceso del servidor del juego.

- La función [InitSDK\(\)](#) ahora usa un token de idempotencia para admitir varios reintentos.
- La [OnUpdateGameSession](#) devolución de llamada ahora puede transferir datos a `autoBackfillMode`

Actualizaciones de complementos clave:

- El complemento para Unreal Engine ahora tiene un proceso de instalación y configuración más simplificado, con más automatización y menos requisitos previos (CMakeOpenSSL y la cadena de herramientas de compilación cruzada de Unreal).
- El complemento de Unreal Engine ha mejorado la experiencia de interfaz de usuario para el flujo de trabajo de EC2 administrado, e incluye la compatibilidad con espacios en las rutas de creación del cliente y del servidor del juego. Además, ahora puede añadir argumentos de línea de comandos al iniciar un cliente de juego desde el editor.
- El complemento para Unreal Engine ahora es compatible con las versiones integradas del servidor ARM. UE5

Más información:

- [Complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#), Guía para desarrolladores de Amazon GameLift Servers
- [SDK 5.x del servidor de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Integración de Amazon GameLift Servers en un proyecto de Unity](#), Guía para desarrolladores de Amazon GameLift Servers

24 de abril de 2025: el SDK 5 del servidor C# es ahora compatible con .NET 8 Amazon GameLift Servers

Versiones actualizadas del SDK:

- SDK del servidor de C#, versión 5.2.1

Para los desarrolladores de juegos que utilizan C#, ahora pueden usar .NET 8 como marco de trabajo de destino para sus proyectos de Amazon GameLift Servers. Con .NET 8, puede aprovechar las mejoras de rendimiento, como la mejora de la compilación just-in-time (JIT), la optimización del uso de la memoria y los tiempos de inicio más rápidos. Si actualmente utiliza .NET 6, le recomendamos que planifique una migración a .NET 8, incluida la actualización del SDK del servidor de C# a la versión más reciente. Microsoft anunció la compatibilidad con .NET 8, con parches de seguridad y actualizaciones técnicas, hasta noviembre de 2026.

[Descargue la versión más reciente del SDK del servidor de C# para Amazon GameLift Servers](#)

Más información:

- [Obtención de herramientas de desarrollo de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers

27 de marzo de 2025: Amazon GameLift Servers amplía el soporte para las familias de instancias EC2 de próxima generación

Versiones actualizadas del SDK:

- AWS SDK 1.11.535

Ahora puede refinar el alojamiento de su servidor de juegos en Amazon GameLift Servers con una selección más amplia de instancias de Amazon EC2 de las familias de instancias de quinta a octava generación. Cada nueva generación de EC2 ofrece avances en el procesamiento, la memoria y las redes de EC2, y las instancias de octava generación ofrecen instancias de AWS de última generación basadas en Graviton4 e Intel Xeon. Las instancias de última generación están disponibles en las siguientes familias de instancias:

- [Uso general](#) (serie M)
- [Computación optimizada](#) (serie C)
- [Memoria optimizada](#) (serie R)

También puede elegir variantes que ofrezcan almacenamiento local (d), redes mejoradas (n) y arquitecturas de procesador específicas (). Intel/AMD/Graviton – i/a/g Las instancias de próxima generación están disponibles en todas las Regiones de AWS regiones compatiblesAmazon GameLift

Servers, a excepción de las regiones de AWS China. Para obtener más información, consulte [Ubicaciones de servicio de Amazon GameLift Servers](#).

Utilice estos nuevos tipos de instancias con flotas de EC2 administradas por Amazon GameLift Servers y flotas de contenedores administradas. Al cambiar el alojamiento de juegos actual a un nuevo tipo de instancia (misma arquitectura), simplemente implemente nuevas flotas, sin modificar los ajustes de configuración excepto el tipo de instancia.

Más información:

- [Precios de instancias de Amazon GameLift Servers](#)
- [CreateFleetAmazon GameLift Servers](#) Referencia de API
- [CreateContainerFleet](#), Referencia Amazon GameLift Servers de API

11 de marzo de 2025: Amazon GameLift Servers lanza el nuevo Game Server Wrapper para una rápida incorporación

El nuevo encapsulador de servidores de juegos para Amazon GameLift Servers reduce considerablemente el tiempo necesario para alojar un servidor de juegos en Amazon GameLift Servers. Sin necesidad de cambiar el código, puede utilizar el contenedor para añadir una funcionalidad básica de administración de sesiones de juego a su juego e implementarla en una flota de Amazon GameLift Servers Anywhere, una flota de EC2 administrada o una flota de contenedores administrada. Esta herramienta es ideal para realizar una evaluación práctica de las características de Amazon GameLift Servers utilizando su propio servidor de juegos o uno de un juego de muestra. También es útil para implementar rápidamente iteraciones de servidores de juegos, por ejemplo, para la creación rápida de prototipos o pruebas.

Con las funciones básicas de administración de sesiones de juego, el servidor de juegos puede inicializar una conexión con el servicio Amazon GameLift Servers, responder a las peticiones para iniciar y detener las sesiones de juego y cerrar una sesión de juego cuando esta ha finalizado.

[Descarga el envoltorio del servidor de juegos desde GitHub](#). Consulta el archivo readme GitHub para saber cómo instalar y usar el contenedor en tus proyectos de juegos.

6 de marzo de 2025: Amazon GameLift es ahora Amazon GameLift Servers y Amazon GameLift Streams

Amazon GameLift es un servicio totalmente gestionado que se dedica a ayudar a los desarrolladores a crear, escalar y ofrecer los juegos más exigentes del mundo. Con la versión de disponibilidad

general de Amazon GameLift Streams, Amazon GameLift ahora ofrece un servidor de juegos de gran escala y capacidades de transmisión de juegos fluidas.

[Amazon GameLift Servers](#) ofrece a los desarrolladores de juegos la capacidad de implementar, operar y escalar servidores de juegos dedicados. Puede implementar servidores de juegos de alto rendimiento en la nube en cuestión de minutos y escalar verticalmente la capacidad para satisfacer la demanda de los jugadores. Basado en un entorno informático AWS comprobado, Amazon GameLift Servers admite 100 millones de jugadores simultáneos en un solo juego, 100 000 jugadores adicionales por segundo y 9 000 nuevas instancias de cómputo por minuto. Además, con una seguridad de nivel empresarial, un sistema de emparejamiento para los jugadores más concurridos y pay-as-you-go flexibilidad, te ayuda a empezar tanto si estás trabajando en una nueva idea de juego como si estás gestionando un juego con millones de jugadores.

[Amazon GameLift Streams](#) ayuda a los desarrolladores de juegos a ofrecer experiencias de streaming de juegos con una resolución de hasta 1080p y 60 frames-per-second (fps) sin ningún retraso perceptible en dispositivos como iOS, Android, FireOS y PCs para los jugadores. Con una única AWS oferta, los editores pueden implementar el contenido de sus juegos en cuestión de minutos, sin modificaciones, en instancias de GPU basadas en la nube y totalmente administradas y distribuirlo directamente a través de AWS Network Backbone a cualquier dispositivo de usuario final con un navegador web. Los jugadores pueden empezar a jugar en tan solo unos segundos, sin esperar a que se descarguen o se instalen, y esto ofrece una experiencia de juego prácticamente indistinguible de la de jugar al juego de forma local en un PC o una consola de juegos.

14 de enero de 2025: Amazon GameLift Servers amplía el soporte para la priorización de ubicación de las sesiones de juego

Versiones actualizadas del SDK:

- AWS SDK 1.11.485

En respuesta a los comentarios de los clientes, lanzamos una nueva funcionalidad que os permitirá priorizar las ubicaciones para las solicitudes de ubicación de cada sesión de juego. Para las colas que están configuradas para priorizar las ubicaciones por ubicación, ahora puede proporcionar una lista personalizada de ubicaciones prioritarias con cada solicitud de ubicación.

Esta nueva característica permite a los clientes cambiar dinámicamente las prioridades de ubicación para cada solicitud de ubicación según sea necesario. La flexibilidad adicional significa que puede responder mejor a las condiciones cambiantes, como la ubicación de los jugadores, la carga de la

flota o el estado de los servidores. También puede ayudar a los clientes que desean personalizar aún más la forma en que se seleccionan las ubicaciones de ubicación.

Más información:

- [Priorización de la ubicación de las sesiones de juego](#), Guía para desarrolladores de Amazon GameLift Servers
- [StartGameSessionPlacement](#), Amazon GameLift Servers Referencia de API

2 de enero de 2025: Amazon GameLift Servers lanza un nuevo soporte para finalizar las sesiones de juego

Versiones actualizadas del SDK:

- AWS SDK 1.11.477

En respuesta a los comentarios de los clientes, lanzamos una nueva funcionalidad que les permitirá finalizar sesiones de juego individuales con más facilidad. Con esta versión, ahora puedes finalizar una sesión de juego directamente en la Amazon GameLift Servers consola o mediante el uso del AWS CLI AWS SDK para ello. Amazon GameLift Servers

Esta nueva característica responde a la necesidad de resolver las sesiones de juego que permanecen activas pero en mal estado, lo que impide que los recursos de computación alojen nuevas sesiones de juego. Anteriormente, los clientes tenían que acceder de forma remota al ordenador para finalizar manualmente una sesión de juego.

Existen dos métodos de finalización entre los que elegir. El primer método intenta finalizar correctamente una sesión de juego mediante su secuencia de cierre personalizada, que puede incluir acciones para notificar a los jugadores y resolver los datos del juego. El segundo método fuerza la detención del proceso del servidor, lo que finaliza la sesión de juego inmediatamente. Este segundo método garantiza que la sesión de juego finalice incluso cuando el proceso del servidor no responda.

Más información:

- [Cierre de una sesión de juego mediante la consola de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [TerminateGameSession](#), Referencia Amazon GameLift Servers de la API

19 de diciembre de 2024: Amazon GameLift Servers publica la compatibilidad con los complementos de motores de juegos para contenedores administrados

Versiones de complementos actualizadas:

Complemento de Amazon GameLift Servers para Unreal Engine, versión 2.0.0

- Se ha actualizado para admitir el SDK 5.2.0 del servidor C++ con compatibilidad para contenedores administrados.
- Se añade el soporte para Unreal Engine 5.4 y 5.5.

Complemento de Amazon GameLift Servers para Unity, versión 3.0.0

- Se ha actualizado para admitir el SDK 5.2.0 del servidor C++ con compatibilidad para contenedores administrados.
- Soporte para Unity 2021.3 LTS y 2022.3 LTS para Windows y Mac OS.

El complemento de Amazon GameLift Servers para los motores de juegos Unreal y Unity proporciona herramientas y flujos de trabajo que simplifican los pasos necesarios para poner en marcha el juego con Amazon GameLift Servers. Amazon GameLift Servers es un servicio de alojamiento en la nube totalmente administrado que los desarrolladores de juegos pueden utilizar para administrar y escalar servidores de juegos dedicados para juegos multijugador basados en sesiones.

Las versiones más recientes del complemento ofrecen las siguientes mejoras:

- Flujo de trabajo guiado para el alojamiento con contenedores administrados. Este flujo de trabajo le guía por los pasos necesarios para configurar una imagen de contenedor con el software de su servidor de juegos e implementar una solución de alojamiento basada en la nube para su servidor de juegos. El flujo de trabajo ofrece dos escenarios de implementación diferentes: una implementación simple y una implementación más completa con una cola de ubicación de las sesiones de juego y un emparejador de FlexMatch. Cada escenario genera flotas de Amazon GameLift Servers contenedores y AWS recursos de apoyo.
- Proceso mejorado para configurar los perfiles de AWS usuario y administrar las credenciales de AWS acceso para el uso de los complementos. Puedes mantener varios perfiles para trabajar con diferentes AWS cuentas, usuarios de cuentas y regiones.
- Funcionalidad adicional para actualizar flotas de contenedores existentes. Puede implementar nuevas imágenes de contenedor (por ejemplo, para las actualizaciones de versión de los

servidores de juegos) y cambiar los ajustes de configuración de la flota sin necesidad de empezar desde cero.

- Flujos de trabajo mejorados para el alojamiento con flotas de Amazon GameLift Servers Anywhere y EC2 administrado. Las mejoras basadas en los comentarios de los clientes incluyen mejores directrices, con consejos y enlaces a recursos útiles.

Los escenarios de despliegue de las soluciones Managed Containers y Managed EC2 utilizan AWS CloudFormation plantillas para crear e implementar los AWS recursos para cada escenario. Esas plantillas se incluyen en la descarga del complemento de Amazon GameLift Servers y se pueden editar. Puede utilizarlas tal cual o modificarlas para el juego.

Más información:

- [Complemento para Unreal: implementación del juego en una flota de contenedores administrados](#), Guía para desarrolladores de Amazon GameLift Servers
- [Complemento para Unity: Implementación del juego en una flota de contenedores administrados](#), Guía para desarrolladores de Amazon GameLift Servers
- [Descargue el complemento para Unreal desde GitHub](#)
- [Descarga el plugin para Unity desde GitHub](#)
- [Alojamiento de juegos con Amazon GameLift Servers](#)
- [Foro de Amazon GameLift Servers](#)

12 de noviembre de 2024: Amazon GameLift Servers lanza el alojamiento de juegos multijugador con contenedores administrados

Versiones actualizadas del SDK:

- AWS SDK 1.11.445
- SDK del servidor, versión 5.2.0 (todos los lenguajes)

Amazon GameLift Servers lanza para su disponibilidad general una nueva solución de alojamiento para cargas de trabajo de servidores de juegos en contenedores. Con esta versión, los desarrolladores de juegos ahora pueden aprovechar las ventajas de la organización en contenedores, que incluye entornos consistentes y seguros, un proceso de implementación simplificado y una utilización optimizada de los recursos.

Las flotas de contenedores administradas utilizan instancias de Amazon EC2 que Amazon GameLift Servers administra en su nombre y en función de sus configuraciones. Se crea una arquitectura de contenedores personalizada para el juego y se proporcionan imágenes de contenedores al almacenarlas en un repositorio de Amazon Elastic Container Registry (Amazon ECR). Las flotas de contenedores solo están disponibles para los servidores de juegos basados en Linux. Los servidores de juegos deben estar integrados con el SDK del servidor 5.2.0 (o superior).

Con las flotas de contenedores administradas, obtiene los mismos beneficios que con las flotas administradas de EC2. Esto incluye la compatibilidad con los tipos de instancias de spot y bajo demanda, el escalado inteligente de la capacidad, la ubicación de las sesiones de juego con colas y el emparejamiento. También obtiene las mismas métricas que otros tipos de flotas, además de algunas nuevas en el caso de los contenedores. Otras características de las flotas de contenedores incluyen:

- Alineación con la experiencia sin servidor para cargas de trabajo en contenedores. Ejecute un proceso de servidor de juegos por contenedor y empaquete varios contenedores en cada instancia de flota para un uso óptimo de los recursos. Si prefiere tener contenedores con varios procesos de servidor de juegos, puede usar el agente de Amazon GameLift Servers para administrar los hosts de forma automatizada.
- Creación simplificada de flotas. Las flotas de contenedores están diseñadas para requerir ajustes de configuración de despliegue mínimos, con valores razonables. `suggested/default` Puede implementar rápidamente una flota activa y, a continuación, personalizar los ajustes individuales según sea necesario.
- Herramientas de control de versiones para la arquitectura de contenedores. Ahora puede actualizar la definición de un grupo de contenedores (que es similar a una “tarea” de contenedores), mantener varias versiones y especificar qué versión implementar en una flota.
- Herramientas de actualización de flota. Con las flotas de contenedores, ya no necesita crear una nueva flota para lanzar una actualización de la versión del servidor del juego. En su lugar, ahora puede actualizar la imagen de contenedor e implementar las actualizaciones en las flotas existentes.

Puede crear flotas de Amazon GameLift Servers contenedores en cualquier Región de AWS lugar, Amazon GameLift Servers admite flotas con múltiples ubicaciones y puede implementar instancias de flotas de contenedores en cualquier ubicación remota compatible. Para obtener más información,

consulte [Ubicaciones de servicio de Amazon GameLift Servers](#). Los contenedores gestionados no están disponibles actualmente en las regiones de AWS China.

Más información:

- Publicación de blog: [Aproveche los contenedores totalmente administrados para alojar juegos multijugador a escala mundial en Amazon GameLift Servers](#)
- [Contenedores administrados](#) descripción general, guía para Amazon GameLift Servers desarrolladores
- [Cómo funcionan los contenedores en Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Hoja de ruta de desarrollo para el alojamiento con contenedores administrados por Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [CreateContainerFleet](#), Referencia Amazon GameLift Servers de API

19 de septiembre de 2024: Amazon GameLift Servers publica una actualización del SDK del servidor de C++ y de los complementos para Unreal Engine

Versiones actualizadas del SDK:

SDK del servidor de C++, versión 5.1.3

- Nuevas capacidades de registro. Ahora puede acceder a los registros de las solicitudes del SDK.
- Se ha mejorado la fiabilidad de la transmisión de mensajes del SDK. Ahora, el SDK utiliza mecanismos de reconexión más robustos para recuperarse en caso de que haya interrupciones en la red o caídas aleatorias de mensajes.

Versiones de complementos actualizadas:

Complemento de Amazon GameLift Servers para Unreal Engine, versión 1.1.2

- Se ha actualizado para admitir la última versión del SDK 5.1.3 del servidor C++.
- En el complemento de Amazon GameLift Servers para Unreal Engine, al buscar un ejecutable de compilación de servidor para una flota, ahora tiene la opción de examinar Todos los archivos.

Complemento del SDK del servidor de C++ para Unreal (versión 5.1.2)

- Se ha actualizado para admitir la última versión del SDK 5.1.3 del servidor C++.

Más información:

- [Integración de juegos con el complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Descargas de complementos y SDK de Amazon GameLift Servers](#)

5 de septiembre de 2024: Amazon GameLift Servers mejora la observabilidad del proceso de creación de flotas

Con base en los comentarios de los clientes, hemos aclarado el flujo de trabajo de Amazon GameLift Servers a la hora de crear una flota de EC2 administrada y de prepararla para alojar sesiones de juego. Entre las mejoras, se incluyen las características siguientes:

- Hemos proporcionado descripciones más específicas y precisas de cada fase del proceso de creación de la flota. Esta visibilidad mejorada facilita y acelera la identificación y la resolución de los problemas.
- Las fases de compilación y activación separan mejor las tareas de implementación de instancias (compilación) de las tareas para iniciar los procesos del servidor de juegos y conectarse al servicio de Amazon GameLift Servers (activación). Este cambio facilita el reconocimiento de la causa probable de los problemas. Además, ahora puede conectarse de forma remota a las flotas cuando se encuentren en la fase de activación.
- Hay dos nuevos eventos de creación de flotas que comunican si los scripts de instalación del servidor de juegos se han ejecutado correctamente o no. Si la compilación del servidor de juegos incluye un script de instalación, Amazon GameLift Servers intenta ejecutar el script y emite uno de los siguientes eventos nuevos:
 - FLEET_CREATION_COMPLETED_INSTALLER
 - FLEET_CREATION_FAILED_INSTALLER

Más información:

- [Depuración de problemas relacionados con flotas de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Tipo de datos del evento](#), referencia Amazon GameLift Servers de la API

15 de agosto de 2024: Amazon GameLift Servers mejora la experiencia de la consola

Con base en los comentarios de los clientes, hemos realizado las siguientes actualizaciones en la experiencia de la [consola de Amazon GameLift Servers](#):

- Sus preferencias de visualización de las páginas ahora se guardan automáticamente en el usuario de su AWS cuenta y se aplican cada vez que regresa a la página. Las preferencias de visualización le permiten elegir qué información quiere incluir en una tabla, por ejemplo, en la página con el listado de flotas. Personalice sus preferencias de visualización con el icono  en la esquina superior derecha de la tabla.
- El flujo de trabajo Crear flota para las flotas de EC2 administradas se ha simplificado para combinar la selección de ubicaciones de flota y tipos de instancias. Hemos facilitado la búsqueda del tipo de instancia adecuado para una flota, aunque cambie las ubicaciones seleccionadas.

Más información:

- [Cree una EC2 flota Amazon GameLift Servers gestionada](#), Guía para desarrolladores de Amazon GameLift Servers

25 de julio de 2024: Amazon GameLift Servers añade soporte para la zona local de AWS Nigeria

Ahora, con el alojamiento administrado por Amazon GameLift Servers puede implementar recursos de servidores de juegos en Nigeria y África occidental, de modo que pueda ampliar el alcance de sus juegos a toda África. Usa las Zonas AWS Locales para colocar los servidores de juegos geográficamente más cerca de tus jugadores para reducir la latencia y mejorar significativamente la jugabilidad.

A fin de empezar inmediatamente a organizar sesiones de juego en Nigeria, añade la nueva zona local de Nigeria como ubicación remota a una flota con múltiples ubicaciones nueva o existente. Si el juego usa Amazon GameLift Servers FlexMatch, actualice las flotas de la cola de emparejamiento para incluir la nueva zona local. Con flotas multiubicación, puede administrar directamente la capacidad de alojamiento en cada ubicación.

La principal Región de AWS de la zona local de Lagos (Nigeria) es la región de África (Ciudad del Cabo af-south-1) (), que Amazon GameLift Servers también es compatible como ubicación remota. El nombre de la zona local de Nigeria es af-south-1-los-1.

Más información:

- [Ubicaciones de servicio de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers
- [Actualización de ubicaciones de la flota](#), Guía para desarrolladores de Amazon GameLift Servers

2 de julio de 2024: Amazon GameLift Servers lanza una nueva herramienta de consola para ver los datos de las sesiones de los jugadores

La consola de Amazon GameLift Servers ofrece ahora una herramienta de búsqueda de sesiones de jugadores que le permite recuperar datos de las sesiones de los jugadores por ID de sesión de juego, ID de sesión de jugador o ID de jugador. Los juegos que utilizan el emparejamiento de FlexMatch generan automáticamente sesiones de jugador para cada jugador emparejado. Para todos los demás juegos, las sesiones de los jugadores son una característica opcional.

Puede encontrar la herramienta de búsqueda de sesiones de jugadores en la navegación principal de la consola de Amazon GameLift Servers. Vea las sesiones individuales de los jugadores o compare los datos de las sesiones de varios jugadores. También puede abrir los datos de las sesiones de los jugadores al ver la página de detalles de las sesiones de juego.

Más información:

- [Sesiones de juego y de jugador en la consola de Amazon GameLift Servers](#), Guía para desarrolladores de Amazon GameLift Servers

24 de abril de 2024: Amazon GameLift Servers publica una vista previa de flotas de contenedores

Amazon GameLift Servers ofrece ahora una vista previa de las flotas de contenedores, con lo que obtiene mayor portabilidad, escalabilidad, tolerancia a errores y agilidad.

En las flotas de contenedores, las instancias de Amazon EC2 alojan uno o más contenedores. Estos contenedores incluyen el servidor de juegos junto con todo lo que necesite, como las dependencias y las configuraciones. Algunos ejemplos de dependencias son los paquetes SDKs de software. Tras

cargar el contenedor a su Amazon Elastic Container Registry privado, Amazon GameLift Servers rellenará la flota con el contenedor.

Para funcionar en una flota de contenedores, el servidor del juego debe ejecutarse en Linux y estar integrado con el SDK 5.x del servidor. En una flota de contenedores, usted dispone de un control preciso de los recursos de alojamiento para poder optimizar el consumo de recursos, como las unidades de CPU y la memoria. También puede alojar varios servidores de juegos en un contenedor para reducir el uso de recursos.

Una flota de contenedores ofrece muchas de las mismas ventajas que otros tipos de flotas, como los tipos de instancias bajo demanda, el escalado (automático y manual), las colas y el emparejamiento. También obtiene las mismas métricas que otros tipos de flotas, además de algunas nuevas en el caso de los contenedores. Las flotas de contenedores le permiten llegar a los jugadores de forma global en las siguientes ubicaciones y regiones:

- ap-northeast-1
- ap-northeast-2
- ap-southeast-2
- eu-central-1
- eu-west-1
- us-east-1
- us-west-2

Para llegar a más regiones y zonas locales, cree flotas de contenedores con múltiples ubicaciones.

Más información:

- [Gestión del alojamiento con Amazon GameLift Servers contenedores](#), guía Amazon GameLift Servers para desarrolladores
- [CreateContainerGroupDefinition](#), Referencia Amazon GameLift Servers de API

13 de febrero de 2024: Amazon GameLift Servers presenta mejoras y simplifica la instalación del Amazon GameLift Servers plugin para Unreal Engine SDKs

Versiones actualizadas del SDK:

- SDK del servidor de Go, versión 5.1.0

- SDK del servidor de C#, versión 5.1.2
- SDK del servidor de C++, versión 5.1.2

Se han llevado a cabo las siguientes mejoras:

- Se ha mejorado la fiabilidad del SDK añadiendo la reconexión automática en caso de interrupción en la red.
- [Go] Ahora puede llamar a `InitSDK()` con o sin los parámetros del servidor. Los servidores de juegos que se ejecutan en flotas de EC2 administradas por Amazon GameLift Servers leen los parámetros del servidor directamente de las variables de entorno. Los servidores de juegos de las flotas de Amazon GameLift Servers Anywhere deben llamar a `InitSDK()` con los parámetros del servidor.

Versiones de complementos actualizadas:

- Complemento de Amazon GameLift Servers para Unreal Engine, versión 1.1.0
- Complemento de Amazon GameLift Servers para Unity, versión 2.1.0
- Complemento del SDK del servidor de C++ para Unreal, versión 5.1.1
- Complemento del SDK del servidor de C# para Unity, versión 5.1.2

Se han llevado a cabo las siguientes mejoras:

- [Complemento de Amazon GameLift Servers para Unreal Engine] Se han actualizado las instrucciones de instalación y se ha simplificado el empaquetado. Este complemento incluye ahora la última versión del SDK del servidor de C++ para Unreal.
- Se han actualizado los complementos para que sean compatibles con la última versión del SDK del servidor de Amazon GameLift Servers.

Más información:

- [Integración de juegos con el complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Descargas de complementos y SDK de Amazon GameLift Servers](#)

14 de diciembre de 2023: Amazon GameLift Servers añade la capacidad de actualizar las propiedades del juego de las sesiones de juego activas

Ya era posible configurar las propiedades del juego al crear sesiones de juego, así como buscar propiedades específicas en las sesiones. Ahora, también puede añadir y actualizar estas propiedades en una sesión de juego activa.

Por ejemplo, los jugadores votan en un mapa en el que quieren jugar. El cliente del juego llama a `UpdateGameSession` para modificar un valor `GameProperty` en `{"Key": "map", "Value": "jungle"}`. A continuación, el juego implementa el nuevo mapa para los jugadores de la sesión de juego.

Los administradores del juego también pueden recuperar datos útiles de las propiedades del juego mediante esta operación `SearchGameSessions`. Por ejemplo, los administradores pueden enumerar las sesiones de juego que tienen un valor `Status` de `ACTIVE` y la propiedad de juego `{"Key": "map", "Value": "desert"}`.

Más información:

- [the section called “Integración de la funcionalidad de cliente de juego”](#), Guía para desarrolladores de Amazon GameLift Servers
- [GameProperty](#), Referencia Amazon GameLift Servers de API
- [UpdateGameSession](#), Referencia Amazon GameLift Servers de API
- [SearchGameSessions](#), Referencia Amazon GameLift Servers de API

21 de noviembre de 2023: Amazon GameLift Servers lanza el soporte para herramientas de infraestructura como código, como Terraform y Pulumi, impulsadas por API de control de nube de AWS

Ahora puede administrar toda la pila de recursos de Amazon GameLift Servers mediante las herramientas de infraestructura como código (IaC). Estas herramientas incluyen AWS CloudFormation, y también herramientas de terceros, como Terraform y Pulumi. Con este soporte adicional, ahora puede centrarse en desarrollar su juego y aprovechar DevOps las estrategias para encargarse de la gestión de los recursos, la CI/CD y el despliegue para sus clientes.

Ahora también puedes aprovisionar y configurar todos los tipos de Amazon GameLift Servers recursos mediante la API de AWS Cloud Control. Puedes seguir trabajando con los recursos

utilizando las plantillas Amazon GameLift Servers APIs o las AWS CloudFormation plantillas de Amazon GameLift Servers.

Para obtener más información sobre los recursos de Amazon GameLift Servers disponibles a través de la C, consulte la [Referencia de tipos de recurso de Amazon GameLift Servers](#) Referencia de tipos de recurso de Amazon GameLift Servers.

Además, ahora puede escalar automáticamente sus flotas mediante AWS CloudFormation plantillas o la API de AWS Cloud Control mediante la nueva propiedad [Fleet](#):ScalingPolicies.

La API de Cloud Control ofrece a los desarrolladores un conjunto estándar de recursos (CRUDL) APIs para crear, leer, actualizar, eliminar y enumerar recursos (CRUDL) en cientos de AWS servicios y múltiples herramientas de terceros, como Terraform y Pulumi.

Más información:

- [AWS CloudFormation](#)
- [AWS API de control en la nube](#)
- [AWS Proveedor de CC Terraform](#)
- [Pulumi](#)

16 de noviembre de 2023: Amazon GameLift Servers actualiza el complemento independiente para Unity

Versiones actualizadas del SDK:: complemento de Amazon GameLift Servers para Unity, versión 2.0.0

El complemento de Amazon GameLift Servers para Unity proporciona herramientas y flujos de trabajo que simplifican los pasos necesarios para poner en marcha un juego de Unity para el alojamiento en la nube Amazon GameLift Servers. Amazon GameLift Servers es un servicio totalmente administrado que permite a los desarrolladores de juegos administrar y escalar servidores de juegos dedicados para juegos multijugador basados en sesiones.

Con esta versión, el complemento de Unity se actualiza para utilizar las características más recientes de Amazon GameLift Servers, lo que incluye la versión 5.x del SDK del servidor y la compatibilidad para pruebas locales con Amazon GameLift Servers Anywhere. El complemento es compatible con las versiones de Unity 2021.3 LTS y 2022.3 LTS.

Estas son algunas de las características más importantes del complemento:

- Flujos de trabajo de la IU guiados en el editor de Unity para los siguientes escenarios:
 - Pruebe la integración de sus juegos con Amazon GameLift Servers utilizando su estación de trabajo local como host. Este flujo de trabajo le ayuda a configurar una flota de Amazon GameLift Servers Anywhere para su máquina local, lanzar instancias de servidor y el cliente de juegos, solicitar una sesión de juego a través de Amazon GameLift Servers y unirse al juego.
 - Implemente una solución de alojamiento en la nube para su servidor de juegos integrado con EC2 Amazon GameLift Servers gestionado y recursos de soporte AWS . Este flujo de trabajo le ayuda a configurar el juego para el alojamiento en la nube, y le ofrece tres escenarios de implementación:
 - Implementación del servidor del juego en una sola flota.
 - Implemente el servidor de juegos en un conjunto de flotas de Spot de bajo coste en varias AWS regiones.
 - Despliega el servidor del juego con un FlexMatch emparejador.
- Posibilidad de configurar perfiles de usuario que se vinculen a un usuario de la AWS cuenta y establecer una AWS región predeterminada. Puede mantener varios perfiles para trabajar en diferentes AWS cuentas, usuarios de cuentas y regiones.
- Comodidades especiales que ayudan a agilizar los procesos de integración e implementación de Amazon GameLift Servers, entre las que se incluyen:
 - Cada solución de alojamiento incluye AWS recursos de apoyo, incluido un grupo de usuarios de Amazon Cognito que proporciona una validación única para cada jugador. Las soluciones también incluyen un depósito de Amazon S3 para almacenamiento, notificaciones de eventos de Amazon SNS, AWS Lambda funciones y otros recursos.
 - Para el flujo de trabajo de Anywhere, el complemento automatiza la configuración necesaria de los parámetros del servidor.
 - Para el flujo de trabajo de Amazon EC2, cada solución de implementación proporciona un servicio de backend de cliente integrado que utiliza funciones de Lambda. El servicio de backend se encuentra entre el cliente del juego y el servicio Amazon GameLift Servers, y administra todas las llamadas directas al servicio Amazon GameLift Servers.
- Contenido para las pruebas de integración, lo que incluye los recursos y el código para un sencillo ejemplo de juego multijugador, a fin de ilustrar la integración entre el servidor y el cliente del juego.
- Documentación del complemento con una guía de integración detallada y un código de muestra.

Todos los escenarios de despliegue, incluidos los de las flotas Anywhere y Amazon EC2, utilizan AWS CloudFormation plantillas para describir e implementar AWS los recursos de la solución de tu

juego. Esas plantillas están incluidas en la descarga del complemento de Amazon GameLift Servers. Puede utilizarlas tal cual o modificarlas para el juego.

Más información:

- [Complemento de Amazon GameLift Servers para Unity \(SDK 5.x del servidor\)](#), Guía para desarrolladores de Amazon GameLift Servers
- [Descargue el complemento desde GitHub](#)
- [Acerca del alojamiento Amazon GameLift Servers](#)
- [Foro de Amazon GameLift Servers](#)

2 de noviembre de 2023: Amazon GameLift Servers añade soporte para credenciales compartidas

Versiones del SDK actualizadas: AWS SDK 1.11.193

La nueva función de credenciales Amazon GameLift Servers compartidas permite que las aplicaciones que se despliegan en flotas de EC2 gestionadas interactúen con otros recursos. AWS Esta actualización afecta a las aplicaciones que agrupa e implementa junto con archivos binarios de servidores de juegos integrados en la versión 5.x, o posterior, del SDK del servidor. (Los archivos ejecutables del servidor de juegos ya pueden solicitar credenciales mediante la acción `GetFleetRoleCredentials()` del SDK del servidor 5.x).

Por ejemplo, si quieres implementar la compilación de tu servidor de juegos con un CloudWatch agente de Amazon para recopilar métricas de instancias de EC2 y otros datos, el agente necesita permiso para interactuar con tus CloudWatch recursos. Para ello, primero debes configurar un rol de IAM (AWS Identity and Access Management IAM) con permisos para usar los CloudWatch recursos y, a continuación, configurar una flota con el rol de IAM y las credenciales compartidas habilitados. Cuando Amazon GameLift Servers implementa la compilación del servidor de juegos en cada instancia de EC2, genera un archivo de credenciales compartido y lo almacena en la instancia. Todas las aplicaciones de la instancia pueden utilizar las credenciales compartidas. Amazon GameLift Servers actualiza automáticamente las credenciales temporales a lo largo de la vida de la instancia.

Puede habilitar las credenciales compartidas al crear una flota de EC2 administrada mediante los siguientes métodos:

- En el flujo de trabajo de creación de flotas de consolas de Amazon GameLift Servers.

- Al llamar a la operación `CreateFleet` de la API del servicio con el nuevo parámetro `InstanceRoleCredentialsProvider`.
- Al llamar a la operación AWS CLI `aws gamelift create-fleet` con el parámetro `instance-role-credentials-provider`.

Más información:

- [Comuníquese con otros AWS recursos de sus flotas, guía Amazon GameLift Servers](#) para desarrolladores
- [CreateFleet, InstanceRoleCredentialsProvider](#) Referencia de Amazon GameLift Servers API
- [Configuración de un rol de servicio de IAM](#), Guía para desarrolladores de Amazon GameLift Servers

28 de septiembre de 2023: Amazon GameLift Servers lanza un nuevo complemento independiente para Unreal Engine

Versiónes del SDK actualizadas: complemento de Amazon GameLift Servers para Unreal Engine, versión 1.0.0

El complemento de Amazon GameLift Servers para Unreal Engine proporciona herramientas y flujos de trabajo que simplifican los pasos necesarios para poner en marcha un juego con Amazon GameLift Servers el alojamiento en la nube. Amazon GameLift Servers es un servicio totalmente administrado que permite a los desarrolladores de juegos administrar y escalar servidores de juegos dedicados para juegos multijugador basados en sesiones. El complemento es compatible con las versiones de UE 5.0, 5.1 y 5.2. Entre las características principales se incluyen:

- Los flujos de trabajo de la IU guiados [en el editor Unreal] recorren las siguientes rutas:
 - Pruebe la integración de sus juegos con Amazon GameLift Servers utilizando su estación de trabajo local como host. Este flujo de trabajo le ayuda a configurar una flota de Amazon GameLift Servers Anywhere para su máquina local, lanzar instancias de su servidor y cliente de juegos, solicitar una sesión de juego a través de Amazon GameLift Servers y obtener información de conexión para la nueva sesión de juego.
 - Implemente una solución de alojamiento en la nube de Amazon EC2 para su servidor de juegos integrado. Este flujo de trabajo le ayuda a configurar su juego para el alojamiento en la nube y ofrece tres escenarios de implementación diferentes: en una sola flota, en un conjunto de flotas de spot en varias regiones o en un conjunto de flotas con un emparejamiento de FlexMatch.

La solución para cada escenario de despliegue incluye Amazon GameLift Servers recursos y recursos de apoyo AWS .

- Posibilidad de configurar perfiles de usuario que se vinculen a un usuario de la AWS cuenta y definir una AWS región predeterminada. Puede mantener varios perfiles para trabajar en diferentes AWS cuentas, usuarios de cuentas y regiones.
- Comodidades especiales que ayudan a agilizar los procesos de integración e implementación de Amazon GameLift Servers, entre las que se incluyen:
 - Cada solución de alojamiento incluye AWS recursos de apoyo, como un grupo de usuarios básico de Amazon Cognito que proporciona un reproductor único IDs, un depósito de Amazon S3 para almacenamiento, notificaciones de eventos de Amazon SNS y funciones. AWS Lambda
 - Para el flujo de trabajo de Anywhere, el complemento automatiza la configuración necesaria de los parámetros del servidor mediante argumentos de línea de comandos.
 - Para el flujo de trabajo de Amazon EC2, cada solución de implementación proporciona un servicio de backend de cliente integrado que utiliza funciones de Lambda. El servicio de backend recibe las solicitudes de los clientes de juegos y las transfiere al servicio de Amazon GameLift Servers.
- Contenido para realizar pruebas de integración, que incluye un mapa del juego inicial y dos mapas de prueba con planos básicos y elementos de la IU.
- Documentación del complemento con una guía de integración detallada y un código de muestra.

Todos los escenarios de implementación, incluidos los de las flotas Anywhere y Amazon EC2, utilizan AWS CloudFormation plantillas para describir las soluciones. El complemento utiliza estas plantillas al implementar los recursos de Amazon GameLift Servers para el juego. Esas plantillas se incluyen en la descarga del complemento de Amazon GameLift Servers y se pueden editar. Puede utilizarlas tal cual o modificarlas para el juego.

Más información:

- [Complemento de Amazon GameLift Servers para Unreal Engine](#), Guía para desarrolladores de Amazon GameLift Servers
- [Descargue el complemento desde GitHub](#)
- [Acerca del alojamiento Amazon GameLift Servers](#)
- [Foro de Amazon GameLift Servers](#)

17 de agosto de 2023: Amazon GameLift Servers ofrece alojamiento de servidores de juegos con procesadores AWS Graviton

Versiones del SDK actualizadas: AWS SDK 1.11.144

Ahora Amazon GameLift Servers puede alojar sus juegos en la nube mediante instancias EC2 con AWS procesadores Graviton. Diseñadas AWS con procesadores basados en ARM64, las instancias Graviton ofrecen la mejor relación precio-rendimiento para las cargas de trabajo en la nube con EC2, con una mejora de hasta un 40% con respecto a las instancias similares basadas en x86. Los procesadores de Graviton3 más recientes ofrecen un rendimiento informático hasta un 25 % superior al de las versiones anteriores.

Con Amazon GameLift Servers, ahora puede seleccionar entre estas nuevas instancias de la familia Graviton: AWS

- Instancias basadas en Graviton2: c6g, c6gn, r6g, m6g y g5g
- Instancias basadas en Graviton3: c7g, r7g y m7g

Más información:

- [AWS Procesador Graviton](#): conozca las ventajas y los usos prácticos de las instancias EC2 basadas en Graviton.
- [Introducción a Graviton](#): obtenga una visión general de las instancias basadas en Graviton e información sobre cómo se ejecutan las aplicaciones en ellas en función del sistema operativo, los idiomas y los tiempos de ejecución.

Note

Las instancias de Graviton ARM requieren una compilación de servidor para una AMI de Linux. Se requiere el SDK del servidor 5.1.1 o posterior para C++ y C#. Se requiere el SDK del servidor 5.1.1 o posterior para continuar. Estas instancias no out-of-the-box admiten la instalación de Mono en Amazon Linux 2023 (AL2023) o Amazon Linux 2 (AL2).

27 de julio de 2023: Amazon GameLift Servers lanza el SDK del servidor 5.1.0 con compatibilidad adicional para el desarrollo de Unity

Versiones del SDK actualizadas: SDK del servidor para C++, C#/Unity y Unreal 5.1.0

La versión más reciente del SDK del servidor de Amazon GameLift Servers incluye actualizaciones para C++, C# y el complemento de Unreal, además de un nuevo complemento para utilizarlo con el motor de juegos de Unity. Los desarrolladores de juegos integran el SDK del servidor de Amazon GameLift Servers en los servidores de juegos que implementan para su alojamiento en Amazon GameLift Servers.

La última versión del SDK del servidor contiene las siguientes actualizaciones, que incluyen una serie de solicitudes de los clientes:

- Descarga de paquetes del SDK específicos de lenguaje: el [sitio de descargas de Amazon GameLift Servers](#) actualizado contiene paquetes del SDK para cada lenguaje. Puede descargar las versiones actuales o anteriores.
- Nuevo complemento del SDK del servidor de C# para Unity: el nuevo paquete del SDK del servidor para Unity contiene bibliotecas de C# integradas que puede instalar mediante el administrador de paquetes de Unity Editor (consulte la nueva [Guía de integración de Unity](#)). Estas bibliotecas incluyen todas las dependencias necesarias. UnityNuGet Puede utilizar este complemento con Unity 2020.3 LTS, 2021.3 LTS y 2022.3 LTS para Windows y Mac OS. Es compatible con los perfiles .NET Framework y .NET Standard de Unity, con .NET Standard 2.1 y .NET 4.x.
- Solución .NET consolidada para C#: el SDK del servidor para C# ahora es compatible con .NET Framework 4.6.2 (actualizado desde la versión 4.6.1) y .NET 6.0 en una sola solución. .NET Standard 2.1 está disponible con las bibliotecas creadas por Unity.
- Actualizaciones del SDK del servidor 5.1.0
 - [C++, C#, Unreal] Ahora puede llamar a `InitSDK()` con o sin los parámetros del servidor. Los servidores de juegos que se ejecutan en flotas de EC2 administradas por Amazon GameLift Servers leen los parámetros del servidor directamente de las variables de entorno. Los servidores de juegos de las flotas de Amazon GameLift Servers Anywhere deben llamar a `InitSDK()` con los parámetros del servidor.
 - Las llamadas al SDK del servidor [C++, C# y Unreal] han mejorado los mensajes de error.
 - [SDK de C++] Para mejorar los tiempos de compilación del SDK del servidor, el indicador de compilación `-DRUN_CLANG_FORMAT` está deshabilitado de forma predeterminada. Puede habilitarlo con `-DRUN_CLANG_FORMAT=1`.

- [SDK de C++] Al crear las bibliotecas sin las bibliotecas estándar (-DGAMELIFT_USE_STD=0), `InitSDK()` deja de utilizar los tipos de datos `std::`.
- Documentación ampliada del SDK del servidor 5.x
 - Se actualizaron las guías de referencia del SDK del servidor para C++, C#/Unity y Unreal, que incluyen una cobertura ampliada de todos los tipos de datos.
 - [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)
 - [SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#)
 - [SDK 5.x del servidor de C++ \(Unreal\) para Amazon GameLift Servers: acciones](#)
 - Nuevas versiones de las guías de integración del SDK del servidor 5 para los complementos de Unity y Unreal
 - [Integración de Amazon GameLift Servers en un proyecto de Unity](#)
 - [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#)
- Actualizaciones adicionales de la documentación
 - Se revisó la documentación para las operaciones de la API de Amazon GameLift Servers servicio [GetComputeAccess](#) y [GetInstanceAccess](#) para aclarar los procedimientos de acceso remoto en función de la versión Amazon GameLift Servers del SDK del servidor utilizada.
 - Se revisaron las descripciones [GameSessionPlacement](#) para documentar cómo la información de la sesión de juego es transitoria cuando una colocación está en estado «pendiente».

13 de julio de 2023: Amazon GameLift Servers añade métricas de hardware de la flota

Ahora puede realizar un seguimiento de las métricas de rendimiento del hardware de sus flotas de EC2 administradas de Amazon GameLift Servers. Las métricas incluyen métricas de instancias EC2 sobre el uso de la CPU, el volumen de tráfico de red y la actividad del disco read/write. En el caso de Amazon GameLift Servers, estas métricas describen todas las instancias activas en una ubicación de flota. Puedes ver estas métricas de hardware de flota en el CloudWatch panel de control de Amazon en Consola de administración de AWS. También puede verlas en la consola de Amazon GameLift Servers en los detalles de la flota.

Más información:

- [Monitoriza Amazon GameLift Servers con Amazon CloudWatch](#) (Métricas para flotas), Guía para desarrolladores de Amazon GameLift Servers

29 de junio de 2023: Amazon GameLift Servers lanza el soporte para Amazon Linux 2023

Versiones del SDK actualizadas: AWS SDK 1.11.111

Amazon GameLift Servers los clientes ahora pueden usar el sistema operativo Amazon Linux 2023 para alojar sus servidores de juegos. AL2023 ofrece varias mejoras además de AL2 incluir la seguridad. Este sistema operativo está disponible en todas Regiones de AWS las regiones, excepto en las de China.

Los clientes podrán usar los sistemas operativos Linux más recientes y seguir recibiendo actualizaciones de seguridad críticas cuando finalice el soporte para Amazon Linux (AL1) en diciembre de 2023. Support para Amazon Linux 2 continuará hasta el 30 de junio de 2026.

Más información:

- [Preguntas frecuentes sobre Amazon GameLift Servers Linux Server](#)
- [Comparación entre Amazon Linux 2 y Amazon Linux 2023](#)
- Enlaces de referencia de API de Amazon GameLift Servers:
 - [AWS Acción del SDK CreateBuild](#)
 - [Comando de la CLI upload-build](#)
 - [Comando de la CLI create-build](#)

25 de mayo de 2023: se Amazon GameLift Servers FleetIQ añade un filtro para excluir las ubicaciones de las sesiones de juego en las instancias que se agotan

Versiones del SDK actualizadas: AWS SDK 1.11.87

Si utiliza Amazon GameLift Servers FleetIQ como alojamiento de juegos, ahora puede impedir que las sesiones de juego se ubiquen en instancias que se están vaciando actualmente. Las instancias de vaciado están marcadas como cerradas, pero se pueden seleccionar para alojar nuevas sesiones de juego si no hay otros recursos de alojamiento disponibles. Con esta nueva característica, puede excluir por completo el uso de instancias de vaciado.

Utilice esa característica cuando llame a `ClaimGameServer` para buscar los servidores de juegos disponibles. Añada el nuevo parámetro `FilterOption` y establezca los estados de instancia permitidos como `ACTIVOS` únicamente. Como respuesta, Amazon GameLift Servers FleetIQ solo analiza las instancias activas cuando busca y reclama un servidor de juegos disponible.

Más información:

- [ClaimGameServer](#) en la Referencia de la API de Amazon GameLift Servers
- [Cómo funciona FleetIQ](#) en la Guía para desarrolladores de Amazon GameLift Servers FleetIQ

16 de mayo de 2023: Amazon GameLift Servers admite el etiquetado de asignación de costos para flotas

Amazon GameLift Servers los clientes ahora pueden usar etiquetas de asignación de AWS Billing costos para organizar sus costos de alojamiento de juegos. Puede asignar etiquetas de asignación de costos a recursos individuales de la flota EC2 de Amazon GameLift Servers para realizar un seguimiento de la contribución de sus flotas a los costos generales de alojamiento.

Más información:

- [Herramientas de uso y costo de los recursos](#)
- [Uso de etiquetas de asignación de costos de AWS](#), Guía del usuario de AWS Billing

20 de abril de 2023: Amazon GameLift Servers lanza el soporte para Windows Server 2016

Versiones actualizadas del SDK: AWS SDK 1.11.63

Los clientes de Amazon GameLift Servers ahora pueden utilizar el sistema operativo Windows Server 2016 para alojar sus servidores de juegos. Este sistema operativo está disponible en todas. Regiones de AWS Los clientes pueden utilizar el sistema operativo Windows más reciente y seguir recibiendo actualizaciones de seguridad críticas cuando Microsoft finalice la compatibilidad con Windows Server 2012 en octubre de 2023.

A partir de hoy, los nuevos clientes que necesiten un entorno de tiempo de ejecución de Windows deberán especificar Windows Server 2016 al crear nuevas compilaciones de servidores de juegos para su alojamiento. Los clientes actuales pueden seguir creando nuevas compilaciones y flotas con Windows Server 2012, pero deben completar la migración con Windows Server 2016 antes de la fecha de fin de la compatibilidad de Microsoft, el 10 de octubre de 2023.

Esta actualización incluye los cambios de servicio siguientes:

- Al crear una compilación de servidor de juegos con el SDK o los comandos de la CLI de Amazon GameLift Servers, ahora debe configurar el sistema operativo de forma explícita. Ya no hay ningún

valor predeterminado. Para implementar el servidor de juegos en Windows Server 2016, utilice el valor `WINDOWS_2016`.

- Al crear una compilación de servidor de juegos con la consola de Amazon GameLift Servers, debe seleccionar un sistema operativo entre los valores disponibles. Si ya es cliente y tiene flotas activas de Windows Server 2012, puede elegir entre dos opciones: `WINDOWS_2012` o `WINDOWS_2016`.

Más información:

- Enlaces de referencia de API de Amazon GameLift Servers:
 - [Comando de la CLI `upload-build`](#)
 - [Comando de la CLI `create-build`](#)
 - [AWS Acción del SDK `CreateBuild`](#)
- [Preguntas frecuentes sobre Amazon GameLift Servers Windows 2012](#)

13 de abril de 2023: Amazon GameLift Servers lanza el SDK del servidor 5.x para Unreal

Versiónes actualizadas del SDK: SDK del servidor 5.0.0 para Unreal

La última versión del complemento ligero de Amazon GameLift Servers para Unreal Engine ahora se basa en el SDK del servidor de Amazon GameLift Servers 5.x. Para empezar a integrar su entorno de Unreal Engine con Amazon GameLift Servers, consulte los siguientes enlaces.

Más información:

- [Integración de Amazon GameLift Servers en un proyecto de Unreal Engine](#)
- [Adición de Amazon GameLift Servers al servidor de juegos con el SDK del servidor](#)
- [SDK 5.x del servidor de C++ para Amazon GameLift Servers: acciones](#)

14 de marzo de 2023: Amazon GameLift Servers lanza una nueva experiencia de consola

La nueva consola de Amazon GameLift Servers incluye estas mejoras:

- Navegación mejorada: el nuevo panel de navegación facilita la navegación entre los recursos de Amazon GameLift Servers.

- **Página de destino de Amazon GameLift Servers:** la nueva página de destino proporciona enlaces a documentación útil, muestra una descripción general de alto nivel de Amazon GameLift Servers y ofrece compatibilidad a través de enlaces a la documentación, las preguntas frecuentes y AWS re:Post.
- **CloudWatch Métricas de Amazon mejoradas:** Amazon GameLift Servers las métricas ahora están disponibles tanto en la Amazon GameLift Servers consola como en los CloudWatch paneles de control. Esta actualización también incluye nuevas métricas de rendimiento, utilización y sesiones de jugadores.

Más información:

- [Administración de los recursos de alojamiento de juegos con Amazon GameLift Servers](#)
- [Compilación de un emparejador de FlexMatch](#)

14 de febrero de 2023: Amazon GameLift Servers ahora admite el cifrado del servidor para temas de Amazon SNS

El cifrado del servidor (SSE) para temas de SNS cifra los datos confidenciales en reposo. SSE utiliza las teclas AWS Key Management Service (AWS KMS) para proteger el contenido de sus temas de SNS.

Más información:

- [Configuración de la notificación de eventos para la ubicación de sesiones de juego.](#)
- [Eventos de emparejamiento de FlexMatch](#)
- [Cifrado en reposo](#)

9 de febrero de 2023: El SDK del servidor de Amazon GameLift Servers es compatible con .NET 6 con C#10

Versiones actualizadas del SDK: SDK del servidor 5.0.0 para .NET 6. No se requieren actualizaciones del SDK.

Si utiliza la plataforma de desarrollo en tiempo real de Unity, siga utilizando el SDK del servidor de Amazon GameLift Servers 5.0.0 con .NET 4.6. Unity no es compatible con .NET 6.

Más información:

- Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#)
- [SDK 5.x del servidor de C# para Amazon GameLift Servers: acciones](#)

31 de enero de 2023: El SDK del servidor de Amazon GameLift Servers es compatible con el lenguaje de Go

Versiones actualizadas del SDK: SDK del servidor 5.0.0 para Go

Más información:

- Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#)
- [SDK del servidor de Go para Amazon GameLift Servers: acciones](#)

1 de diciembre de 2022: Amazon GameLift Servers lanza Amazon GameLift Servers Anywhere y Amazon GameLift Servers Server SDK 5.0

Versiones del SDK actualizadas: AWS SDK 1.10.21, SDK de servidor 5.0.0 para C++ y C#

Amazon GameLift Servers Anywhere utiliza los recursos del servidor de juegos para alojar los servidores de juegos Amazon GameLift Servers. Puede utilizar Amazon GameLift Servers Anywhere para integrar sus propios recursos informáticos con los recursos de computación de EC2 administrados por Amazon GameLift Servers para distribuir los servidores de juegos entre varios tipos de recursos de computación. También puede utilizar Amazon GameLift Servers Anywhere para probar de forma iterativa los servidores de juegos sin tener que cargar la compilación en Amazon GameLift Servers para cada iteración.

Aspectos destacados:

- Nuevos tipos de flota y recursos de computación de Amazon GameLift Servers Anywhere
- Registro de recursos de computación de Amazon GameLift Servers Anywhere
- Ciclo de iteración de pruebas mejorado

El SDK 5.0.0 del servidor de Amazon GameLift Servers introduce mejoras en el SDK del servidor existente y un nuevo tipo de recurso; el recurso de computación. El SDK del servidor 5.0.0 es compatible con Amazon GameLift Servers Anywhere y el uso de sus propios recursos informáticos para el alojamiento de servidores de juegos.

Más información:

- [SDK 5.x del servidor de Amazon GameLift Servers](#)
- [Ubicaciones geográficas](#)
- [Elección de los recursos de computación para una flota administrada](#)
- [Creación de una flota de Amazon GameLift Servers Anywhere](#)

25 de agosto de 2022: Amazon GameLift Servers lanza el soporte para zonas locales

Versiones del SDK actualizadas: SDK 1.9.333 AWS

Amazon GameLift Servers ya está disponible en ocho zonas locales de Estados Unidos, por lo que puede implementar sus flotas más cerca de los jugadores. Puede utilizar todas las características administradas de Amazon GameLift Servers con las zonas locales añadiendo las zonas locales a sus flotas.

Las Zonas Locales extienden AWS los recursos y los servicios al borde de la nube, cerca de grandes centros poblacionales, industriales y de tecnología de la información (TI). Esto significa que puede implementar aplicaciones que requieren una latencia de milisegundos de un solo dígito más cerca de los usuarios finales o de los centros de datos en las instalaciones.

Más información:

- [Amazon GameLift Servers Zonas locales de](#)
- [Ubicaciones geográficas](#)
- [Cree una EC2 flota Amazon GameLift Servers gestionada](#)

28 de junio de 2022: Amazon GameLift Servers lanza una nueva experiencia de consola opcional

La nueva consola de Amazon GameLift Servers incluye estas mejoras:

- Navegación mejorada: el nuevo panel de navegación facilita la navegación entre los recursos de Amazon GameLift Servers.
- Página de destino de Amazon GameLift Servers: la nueva página de destino proporciona enlaces a documentación útil, muestra una descripción general de alto nivel de Amazon GameLift Servers y ofrece compatibilidad a través de enlaces a la documentación, las preguntas frecuentes y AWS re:Post.
- CloudWatch Métricas de Amazon mejoradas: Amazon GameLift Servers las métricas ahora están disponibles tanto en la Amazon GameLift Servers consola como en los CloudWatch paneles de control. Esta actualización también incluye nuevas métricas de rendimiento, utilización y sesiones de jugadores.

Más información:

- [Administración de los recursos de alojamiento de juegos con Amazon GameLift Servers](#)
- [Compilación de un emparejador de FlexMatch](#)

15 de febrero de 2022: FlexMatch añade una regla compuesta y mejoras adicionales

Los usuarios de FlexMatch ahora tienen acceso a las características siguientes:

- Regla compuesta: se ha añadido compatibilidad con las reglas de emparejamiento compuestas para emparejamientos de 40 o menos jugadores. Ahora puede utilizar declaraciones lógicas para crear una regla compuesta y formar un emparejamiento. Sin una regla compuesta en su conjunto de reglas, para formar un emparejamiento, todas las reglas del conjunto de reglas deben ser verdaderas. Con las reglas compuestas, puede elegir qué reglas aplicar mediante los siguientes operadores lógicos: `and`, `or`, `not` y `xor`.
- Selección de equipos flexible: se actualizaron las expresiones de las propiedades de emparejamiento para permitir la selección de un subconjunto de todos los equipos disponibles.
- Listas de cadenas más largas: se ha aumentado el número máximo de cadenas de 10 a 100 en una lista de cadenas con los valores de los atributos de los jugadores.

Más información:

- [Guía para desarrolladores de Amazon GameLift Servers FlexMatch:](#)
 - [Tipos de reglas de FlexMatch](#)

- [Expresiones de propiedades de FlexMatch](#)
- [AttributeValue: SL](#)

28 de octubre de 2021: Amazon GameLift Servers agrega soporte para flotas multirregionales en la región de Asia Pacífico (Osaka); Amazon GameLift Servers FleetIQ agrega soporte para AWS procesadores Graviton2

[Versiones del SDK actualizadas: SDK 1.9.133AWS](#)

Amazon GameLift Servers ya está disponible en la región de Asia-Pacífico (Osaka). Los desarrolladores de juegos ahora pueden implementar instancias en Osaka con una flota GameLift multirregional.

Ahora puede utilizar los servidores de juegos alojados en Graviton2, respaldados por la arquitectura de procesador basada en ARM, para lograr un mayor rendimiento a un costo menor en comparación con las opciones de computación equivalentes basadas en Intel.

Aspectos destacados:

- Amazon GameLift Servers ya está disponible en la región de Asia-Pacífico (Osaka).
- Los grupos de servidores de juegos de Amazon GameLift Servers FleetIQ ahora se pueden configurar para administrar las familias de instancias c6g, m6g y r6g de Graviton2.

Más información:

- [Flota multirregional de Amazon GameLift Servers](#)
- [CreateGameServerGroup](#)
- [AWS procesador de gravitones](#)

20 de septiembre de 2021: Amazon GameLift Servers lanza un complemento para Unity

El complemento de Amazon GameLift Servers para la versión 1.0.0 de Unity contiene bibliotecas y una IU nativa que facilita el acceso a los recursos de Amazon GameLift Servers e integra Amazon GameLift Servers en el juego de Unity. Puedes usar el Amazon GameLift Servers complemento de Unity para acceder Amazon GameLift Servers APIs e implementar CloudFormation plantillas para

escenarios de juego comunes. El complemento también incluye un juego de muestra que funciona con los escenarios de muestra. Puede utilizar Amazon GameLift Servers Local para ver los mensajes que se transmiten entre el cliente y el servidor del juego y obtener información sobre cómo interactúa un juego normal con Amazon GameLift Servers.

El complemento para Unity es compatible con Unity 2019.4 LTS y 2020.3 LTS.

Aspectos destacados:

- Cree, ejecute y modifique un juego de ejemplo con diferentes escenarios, o cree el suyo propio.
- Usa escenarios de ejemplo para AWS CloudFormation escenarios de juego típicos, como solo autenticación, flotas de una sola región, flotas multirregionales con cola y emparejador personalizado, flotas puntuales con cola y emparejador personalizado, y. FlexMatch

Más información:

- [Integración de juegos con el complemento de Amazon GameLift Servers para Unity](#)

30 FlexMatch de junio de 2021: se añade la regla BatchDistance

Puede utilizar el tipo de regla batchDistance para especificar un atributo numérico o de cadena, lo que aporta una serie de ventajas a cada segmento.

Aspectos destacados:

- En emparejamientos de gran tamaño (más de 40 jugadores), en lugar de equilibrar uniformemente a los jugadores solo por habilidad, ahora puede conseguir el mismo equilibrio en función de la habilidad, los modos y los mapas. Asegúrese de que todos los jugadores del emparejamiento estén en un grupo de habilidades, agrupe varios atributos numéricos, como la liga o el estilo de juego, y agrupe según cadenas de atributos, como el mapa o el modo de juego. También puede crear expansiones a lo largo del tiempo. Por ejemplo, puede crear una expansión para permitir que un mayor nivel de habilidad entre en el emparejamiento cuanto más tiempo espere el jugador.

Para emparejamientos de menos de 40 jugadores, puede utilizar una nueva expresión de reglas simplificada.

3 de junio de 2021: Actualizaciones del SDK de cliente y del SDK del servidor en tiempo real de Amazon GameLift Servers

Versiones actualizadas del SDK: SDK del cliente de Realtime 1.2.0, SDK del servidor 3.4.0 para Unreal

Con esta última actualización del SDK, ahora puede integrar el IL2 CPP en sus aplicaciones móviles que utilizan el RTS Client SDK y seguir las mejores prácticas con los marcos. Ahora también puede crear el SDK de Amazon GameLift Servers para Unreal, versión 4.26. Esta actualización contiene componentes que se integran con su servidor de juegos de Windows o Linux, incluidas las versiones C++ y C# del SDK del servidor de Amazon GameLift Servers, Amazon GameLift Servers local y un complemento de Unreal Engine.

Aspectos destacados:

- Se ha añadido compatibilidad con el IL2 CPP en el SDK del cliente de RTS y con la creación de bibliotecas nativas como marcos, de forma que pueda crear clientes de RTS para los dispositivos móviles más recientes.
- Puede utilizar [DescribePlayerSessions\(\)](#) para obtener información para una única sesión de jugador, para todas las sesiones de jugador de una sesión de juego o para todas las sesiones de jugador asociadas a un solo ID de jugador.
- Puede usar [GetInstanceCertificate\(\)](#) para recuperar la ubicación del archivo de un certificado TLS codificado en PEM que esté asociado a la flota y sus instancias.
- Se creó la compatibilidad con el SDK del servidor para la versión 4.26 de Unreal.
- Se ha comprobado que el SDK de C# existente, versión 4.0.2, es compatible con Unity 2020.3. No se requieren actualizaciones del SDK.

Más información:

- [Amazon GameLift ServersGuía para desarrolladores:](#)
 - [DescribePlayerSessions\(\)](#)
 - [GetInstanceCertificate\(\)](#)

23 de marzo de 2021: Amazon GameLift Servers añade notificaciones a la ubicación de las sesiones de juego

Versiones actualizadas del SDK: AWS SDK [1.8.168](#)

Ahora puede utilizar los eventos para supervisar la actividad de ubicación de las sesiones de juego para una cola de sesiones de juego. Crea un tema del Amazon Simple Notification Service (Amazon SNS) para publicar notificaciones de eventos o configura el seguimiento de eventos mediante Events. CloudWatch

Aspectos destacados:

- Para cada cola, puede configurar una cadena de texto personalizada para incluirla en todos los mensajes de eventos.
- Al utilizar un tema de Amazon SNS, podrá establecer condiciones de acceso adicionales que limiten la publicación a colas específicas.

Más información:

- Guía para desarrolladores de Amazon GameLift Servers:
 - [Configuración de la notificación de eventos para la ubicación de sesiones de juego](#). (nuevo)
 - [Eventos de ubicación de sesión de juego](#) (nuevo)
- [Referencia de la API \(SDK de AWS \)](#)
 - Nuevos parámetros de cola de sesiones de juego NotificationTarget y CustomEventData: [GameSessionQueue](#), [CreateGameSessionQueueUpdateGameSessionQueue](#)
- [Foro de Amazon GameLift Servers](#)

16 de marzo de 2021: Amazon GameLift Servers añade flotas de varias regiones y seis nuevas regiones

[Versiones del SDK actualizadas: AWS SDK 1.8.163](#)

Amazon GameLift ServersEl alojamiento gestionado ahora está disponible en 21 AWS regiones. Las nuevas regiones son Ciudad del Cabo (af-south-1), Bahréin (me-south-1), Hong Kong (ap-east-1), Milán (eu-south-1), París (eu-west-3) y Estocolmo (eu-north-1).

Con la nueva característica de flotas con múltiples ubicaciones de Amazon GameLift Servers, ahora puede configurar una sola flota para alojar sus servidores de juegos en cualquiera de las 20 regiones compatibles con Amazon GameLift Servers o en todas ellas (excepto la región de Pekín). El objetivo de esta característica es reducir considerablemente el trabajo necesario para configurar y mantener los recursos de alojamiento de Amazon GameLift Servers en todo el mundo. Se pueden crear flotas con múltiples ubicaciones en las siguientes AWS regiones: `us-east-1` (Virginia del Norte), `us-west-2` (Oregón), `eu-central-1` (Fráncfort), `eu-west-1` (Irlanda), `ap-southeast-2` (Sídney), `ap-northeast-1` (Tokio) y `ap-northeast-2` (Seúl). En todas las demás regiones, puede seguir configurando flotas de una sola ubicación según sea necesario. Todas las flotas que se crearon antes de esta versión son flotas de una sola ubicación. El uso de flotas con múltiples ubicaciones no afecta a los costos de alojamiento. Los precios de Amazon GameLift Servers se basan en el tipo, la ubicación y el volumen de instancias que utilice. (Para obtener más información, consulte los [Amazon GameLift Servers precios](#)). CloudFormation El soporte para flotas con múltiples ubicaciones estará disponible pronto.

 Note

Las flotas con múltiples ubicaciones no están disponibles en las regiones de China. Amazon GameLift Servers los recursos que residen en las regiones de China no pueden interactuar con los recursos de otras Amazon GameLift Servers regiones ni ser utilizados por ellos.

Aspectos destacados:

- En el caso de una flota con múltiples ubicaciones, añada explícitamente una lista de ubicaciones remotas. Amazon GameLift Servers implementa instancias del mismo tipo y configuración, incluida la configuración de compilación y tiempo de ejecución, en la región de origen de la flota y en todas las ubicaciones añadidas.
- Ajuste la configuración de capacidad y el escalado de cada ubicación de forma independiente. Las políticas de escalado automático se aplican a toda la flota, pero puede activarlas o desactivarlas por ubicación.
- Inicie nuevas sesiones de juego en ubicaciones específicas de la flota. Al utilizar las colas de las sesiones de juego o el emparejamiento para ubicar las sesiones de juego, ahora puede priorizar el lugar de inicio de las nuevas sesiones de juego en función de la ubicación, el costo del alojamiento y la latencia de los jugadores.
- Obtenga métricas de alojamiento en la consola de Amazon GameLift Servers, agregadas para todas las ubicaciones de una flota o desglosadas por cada ubicación de la flota.

Más información:

- [Blog de tecnología de juegos de Amazon](#)
- [Referencia de la API \(SDK de AWS \)](#)
 - Nuevas operaciones de localización de la flota: [CreateFleetLocations](#),, [DescribeFleetLocationAttributes](#), [DescribeFleetLocationCapacity](#), [DescribeFleetLocationUtilization](#), [DeleteFleetLocations](#)
 - Operaciones de flota actualizadas, con un nuevo soporte para múltiples ubicaciones: [CreateFleet](#), [UpdateFleetCapacity](#), [Describe EC2 InstanceLimits](#),, [DescribeInstances](#), [StopFleetActions](#), [StartFleetActions](#)
 - Se actualizaron las operaciones de ubicación de las sesiones de juego, con una nueva prioridad y capacidad de filtrado: [CreateGameSessionQueue](#),, [DescribeGameSessionQueues](#), [UpdateGameSessionQueue](#)
 - Operaciones de creación de sesiones de juego actualizadas, con un nuevo soporte de ubicación: [CreateGameSession](#), [DescribeGameSessions](#), [DescribeGameSessionDetails](#), [SearchGameSessions](#)
- [Amazon GameLift Servers Guía para desarrolladores:](#)
 - [Ubicaciones de servicio de Amazon GameLift Servers](#) (actualizado)
 - [Personalizaciones de recursos de alojamiento](#) (nuevo)
 - [Escalado de la capacidad de alojamiento de juegos con Amazon GameLift Servers](#) (actualizado)
 - [Personalización de colas de sesiones de juego](#) (nuevo)
 - [Detalles de la flota en la consola de Amazon GameLift Servers](#) (actualizado)
- [Foro de Amazon GameLift Servers](#)

9 de febrero de 2021: Amazon GameLift Servers amplía la compatibilidad con las instancias de AMD, FlexMatch independiente

Versiones actualizadas del SDK: AWS SDK [1.8.139](#)

Esta versión incluye las siguientes actualizaciones:

- Los grupos de servidores de juegos de Amazon GameLift Servers FleetIQ ahora se pueden configurar para administrar las familias de instancias de AMD C5a, M5a y R5a. Los tipos de instancias de Amazon EC2 compatibles, tal y como se indican en la lista `GameServerGroupInstanceDefinition`, incluyen ahora los siguientes:

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge y c5a.24xlarge
- m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge y m5a.24xlarge
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge y r5a.24xlarge

Nota: Las instancias de AMD para actualmente no FleetIQ están disponibles para su uso en la AWS región de China (Pekín). Consulte [Disponibilidad de la característica y diferencias en la implementación](#) en China.

- El alojamiento de juegos administrado de Amazon GameLift Servers ya es compatible con instancias de AMD en la región de China (Pekín), operada por Sinnet. Las nuevas familias de instancias de AMD incluyen M5a y R5a. Los tipos de instancias EC2 compatibles, tal y como se indican para la flota [InstanceType](#), ahora incluyen los siguientes:
 - m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge y m5a.24xlarge
 - r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge y r5a.24xlarge
- Amazon GameLift Servers FlexMatch ya se puede utilizar como solución de emparejamiento independiente en la región de China (Pekín), operada por Sinnet. Los clientes pueden crear un FlexMatch emparejador en la región de Pekín y configurar el [FlexMatchMode](#) parámetro en STANDALONE. Para obtener más información sobre FlexMatch el alojamiento Amazon GameLift Servers gestionado o con una solución que no sea de Amazon GameLift Servers alojamiento, consulta la Guía para [Amazon GameLift Servers FlexMatch desarrolladores](#).
- Al configurar las notificaciones de eventos para Amazon GameLift Servers FlexMatch, ahora puede designar un tema FIFO de Amazon SNS como destino de las notificaciones. Para obtener más información, consulte lo siguiente:
 - [MatchmakingConfiguration NotificationTarget](#), Referencia Amazon GameLift Servers de API
 - [Configurar la notificación de FlexMatch eventos](#), Guía para Amazon GameLift Servers FlexMatch desarrolladores
 - [Presentamos Amazon SNS FIFO: First-in-first-out Pub/Sub mensajería, blog de](#) noticias AWS

22 de diciembre de 2020: el SDK del servidor de Amazon GameLift Servers server SDK supports Unreal Engine 4.25 and Unity 2020 es compatible con Unreal Engine 4.25 y Unity 2020

Versiones actualizadas del SDK: SDK 4.0.2 del servidor de Amazon GameLift Servers, versión 3.3.3 del complemento para Unreal

La versión más reciente del SDK del servidor de Amazon GameLift Servers contiene los siguientes componentes:

- El complemento de Unreal se ha actualizado para que sea compatible con Unreal Engine 4.25. La API no se ha modificado.
- Se ha comprobado que el SDK de C# existente, versión 4.0.2, es compatible con Unity 2020. No se requieren actualizaciones del SDK.

Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#).

24 de noviembre de 2020: Amazon GameLift Servers FlexMatch ya está disponible para juegos alojados en Anywhere

Versiones del SDK actualizadas: AWS [SDK 1.8.95](#)

Amazon GameLift Servers FlexMatch es un servicio de emparejamiento personalizable para juegos multijugador. Diseñado inicialmente para usuarios de alojamiento Amazon GameLift Servers gestionado, ahora se FlexMatch puede integrar en juegos que utilizan otros sistemas de alojamiento peer-to-peer, como la informática local propia y la informática en la nube primitiva. Los juegos que utilizan Amazon GameLift Servers FleetIQ para el alojamiento de juegos en Amazon EC2 ahora pueden implementar el emparejamiento con FlexMatch.

FlexMatch proporciona un algoritmo de emparejamiento sólido y un lenguaje de reglas que le proporciona una amplia libertad para personalizar el proceso de emparejamiento, de modo que los jugadores se emparejan en función de las características clave de los jugadores y la latencia informada. Además, FlexMatch ofrece un flujo de trabajo de solicitudes de emparejamiento que admite características como los grupos de jugadores, la aceptación de los jugadores y la reposición de emparejamientos. Cuando utiliza FlexMatch con el alojamiento administrado de Amazon GameLift Servers o Amazon GameLift Servers Realtime, el emparejador utiliza automáticamente Amazon GameLift Servers para buscar recursos de alojamiento e iniciar una nueva sesión de juego para

los emparejamientos recién formados. Al usar FlexMatch como un servicio independiente, el emparejador devuelve los resultados de los emparejamientos a su juego, que luego puede iniciar una nueva sesión de juego con la solución de alojamiento.

Las operaciones de la API FlexMatch forman parte de la API de Amazon GameLift Servers servicio, que se incluye en el AWS SDK y en AWS Command Line Interface (AWS CLI). Esta versión incluye estas actualizaciones para admitir el emparejamiento independiente:

- El elemento `MatchmakingConfiguration` del recurso de la API tiene los siguientes cambios:
 - Nueva propiedad, `FlexMatchMode` indica si el emparejador se está utilizando con el alojamiento administrado de Amazon GameLift Servers o como emparejamiento independiente.
 - La propiedad `GameSessionQueueArns` no es necesaria cuando `FlexMatchMode` se establece en independiente.
 - Estas propiedades no se utilizan con el emparejamiento independiente: `AdditionalPlayerCount`, `BackfillMode`, `GameProperties` y `GameSessionData`.
- La característica de reposición automática no está disponible con el emparejamiento independiente.

24 de noviembre de 2020: las instancias de AMD ya están disponibles en Amazon GameLift Servers

Versiones del SDK actualizadas: AWS SDK [1.8.95](#)

La lista de tipos de instancias de Amazon EC2 compatibles con Amazon GameLift Servers incluye ahora tres nuevas familias de instancias: C5a, M5a y R5a. Estas familias consisten en instancias optimizadas para computación de AMD que funcionan con procesadores AMD EPYC que funcionan a frecuencias de hasta 3.3. GHz. Las instancias de AMD son compatibles con la versión x86; los juegos que se ejecutan actualmente en Amazon GameLift Servers se pueden implementar en los tipos de instancias de AMD sin modificarlos. Las nuevas instancias están disponibles en las siguientes AWS regiones: EE.UU. Este (Norte de Virginia y Ohio), EE.UU. Oeste (Oregón y Norte de California), Centro de Canadá (Montreal), Sudamérica (São Paulo), Centro de la UE (Fráncfort), UE Oeste (Londres e Irlanda), Asia Pacífico Sur (Bombay), Asia Pacífico Noreste (Seúl y Tokio) y Asia Pacífico Sudeste (Singapur y Sídney).

Las nuevas instancias de AMD incluyen:

- c5a.large, c5a.xlarge, c5a.2xlarge, c5a.4xlarge, c5a.8xlarge, c5a.12xlarge, c5a.16xlarge y c5a.24xlarge

- m5a.large, m5a.xlarge, m5a.2xlarge, m5a.4xlarge, m5a.8xlarge, m5a.12xlarge, m5a.16xlarge y m5a.24xlarge
- r5a.large, r5a.xlarge, r5a.2xlarge, r5a.4xlarge, r5a.8xlarge, r5a.12xlarge, r5a.16xlarge y r5a.24xlarge

Más información:

- [Blog de tecnología de juegos de Amazon](#)
- [Precios de instancias de Amazon GameLift Servers](#)
- [Instancias de Amazon EC2 con procesadores EPYC de AMD](#)
- [Foro de Amazon GameLift Servers](#)

11 de noviembre de 2020: actualización de la versión del SDK del servidor de Amazon GameLift Servers

Versiones actualizadas del SDK: SDK para Amazon GameLift Servers servidores 4.0.2

En la nueva versión 4.0.2 del SDK del servidor se ha corregido un problema conocido relacionado con `StartMatchBackfill()` en relación con el funcionamiento de la API. Esta operación ahora devuelve una respuesta correcta a una solicitud de reposición de emparejamientos.

El problema no afectó al proceso de reposición de emparejamientos y no se ha producido ningún cambio en el funcionamiento de esta característica. Es posible que el problema haya afectado a los mensajes de registro y a la gestión de errores en las solicitudes de reposición de emparejamientos.

Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#).

5 de noviembre de 2020: nuevas personalizaciones del algoritmo de FlexMatch

Los usuarios de FlexMatch ahora pueden ajustar los siguientes comportamientos predeterminados para el proceso de emparejamiento. Estas personalizaciones se establecen en un conjunto de reglas de emparejamiento. No hay cambios en. Amazon GameLift Servers SDKs

- Priorice los tickets de reposición: cuando busque emparejamientos aceptables, podrá elegir subir o bajar la prioridad de los tickets de reposición para los emparejamientos. Priorizar los tickets de reposición resulta útil cuando la característica de reposición automática está habilitada. Utilice la propiedad del algoritmo `backfillPriority`.

- Clasifique previamente para optimizar la coherencia y la eficiencia de los emparejamientos: configure el emparejador para que clasifique previamente el grupo de tickets antes de agruparlos por lotes para su evaluación. Al clasificar previamente los tickets en función de los atributos clave de los jugadores, los emparejamientos resultantes suelen tener jugadores más parecidos en esos atributos. También puede aumentar la eficiencia del proceso de evaluación clasificando previamente los mismos atributos que se utilizan en las reglas de emparejamientos. Utilice la propiedad del algoritmo `sortByAttributes` con la propiedad `strategy` establecida en “ordenado”.
- Ajuste la forma en que se activan los tiempos de espera de las expansiones: elija entre activar las expansiones en función de la antigüedad del ticket más nuevo (predeterminado) o el más antiguo en un emparejamiento incompleto. Si se activa con el ticket más antiguo, los emparejamientos se completan más rápido, mientras que si se activa con el ticket más nuevo, la calidad del emparejamiento es superior. Utilice la propiedad del algoritmo `expansionAgeSelection`.

17 de septiembre de 2020: Amazon GameLift Servers actualiza el SDK del servidor

Versiones de SDK actualizadas: SDK Amazon GameLift Servers de servidor 4.0.1

El nuevo SDK del servidor contiene las siguientes actualizaciones:

- Versión de la API de C# 4.0.1
 - La operación de la API [TerminateGameSession\(\)](#) ya no es compatible. Sustitúyala por una llamada a [ProcessEnding\(\)](#) para finalizar la sesión de juego y el proceso del servidor.
 - Se ha solucionado un problema conocido relacionado con la operación [GetInstanceCertificate\(\)](#).
 - La operación [GetTerminationTime\(\)](#) ahora devuelve un valor del tipo `AwsDateTimeOutcome` de datos.
- Versión de la API de C++ 3.4.1
 - La operación [TerminateGameSession\(\)](#) ya no es compatible. Sustitúyala por una llamada a [ProcessEnding\(\)](#) para finalizar la sesión de juego y el proceso del servidor.
- Versión del complemento de Unreal Engine 3.3.2
 - La operación [TerminateGameSession\(\)](#) ya no es compatible. Sustitúyala por una llamada a [ProcessEnding\(\)](#) para finalizar la sesión de juego y el proceso del servidor.
 - Se añade la operación de devolución de llamada `OnUpdateGameSession` a [FProcessParámetros](#) para facilitar la reposición de emparejamientos.

Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#).

27 de agosto de 2020: Amazon GameLift Servers FleetIQ para alojamiento de juegos con Amazon EC2 (disponibilidad general)

Versiones del SDK actualizadas: AWS SDK [1.8.36](#)

La Amazon GameLift Servers FleetIQ solución de alojamiento de juegos de bajo costo y en la nube en Amazon EC2 ya está disponible de forma general. Amazon GameLift Servers FleetIQ ofrece a los desarrolladores la posibilidad de alojar servidores de juegos directamente en instancias de spot de Amazon EC2 al optimizar su viabilidad para el alojamiento de juegos. Los desarrolladores de juegos pueden utilizar Amazon GameLift Servers FleetIQ con juegos nuevos o para complementar la capacidad de los juegos existentes. Esta solución admite el uso de contenedores u otros AWS servicios, como AWS Shield y Amazon Elastic Container Service (Amazon ECS).

Esta versión de disponibilidad general incluye las siguientes actualizaciones de la solución Amazon GameLift Servers FleetIQ:

- La nueva operación de la API `DescribeGameServerInstances` devuelve información, incluido el estado, de todas las instancias activas de un grupo de servidores de Amazon GameLift Servers FleetIQ juegos.
- La nueva estrategia de equilibrio, `ON_DEMAND_ONLY`, configura un grupo de servidores de juegos para que utilice únicamente instancias bajo demanda. Puede actualizar la estrategia de equilibrio de un grupo de servidores de juegos en cualquier momento, lo que permite cambiar entre el uso de instancias de spot e instancias bajo demanda según sea necesario.
- Se han eliminado los siguientes elementos de la vista previa por motivos de disponibilidad general:
 - Uso de claves de clasificación personalizadas para los recursos del servidor de juegos. Los servidores de juegos se pueden ordenar según la marca de tiempo de registro.
 - Etiquetado de los recursos del servidor de juegos.

16 de abril de 2020: Amazon GameLift Servers actualiza el SDK del servidor para Unity y Unreal Engine

Versiones actualizadas del SDK: SDK 4.0.0 del servidor de Amazon GameLift Servers, Amazon GameLift Servers local 1.0.5

La versión más reciente del SDK del servidor de Amazon GameLift Servers contiene los siguientes componentes actualizados:

- SDK de C#, versión 4.0.0 actualizada para Unity 2019.
- Complemento de Unreal, versión 3.3.1 actualizada para Unreal Engine, versiones 4.22, 4.23 y 4.24
- Amazon GameLift Servers local, versión 1.0.5 actualizada para probar integraciones que utilizan el SDK del servidor de C# 4.0.0.

Descargue la versión más reciente del SDK del servidor de Amazon GameLift Servers en [Introducción a Amazon GameLift Servers](#).

2 de abril de 2020: Amazon GameLift Servers FleetIQ disponible para el alojamiento de juegos en EC2 (vista previa pública)

Versiones del SDK actualizadas: AWS SDK [1.7.310](#)

La característica FleetIQ de Amazon GameLift Servers optimiza la viabilidad de las instancias de spot de bajo costo para su uso con alojamiento de juegos. Esta característica se amplía ahora para los clientes que desean administrar sus recursos de alojamiento directamente en lugar de a través del servicio Amazon GameLift Servers administrado. Esta solución admite el uso de contenedores u otros AWS servicios, como AWS Shield y Amazon Elastic Container Service (Amazon ECS).

Más información:

GameTech publicación de [blog](#) sobre Amazon GameLift Servers FleetIQ

19 de diciembre de 2019: gestión de AWS Amazon GameLift Servers recursos mejorada

Versiones del SDK actualizadas: AWS SDK [1.7.249](#)

Ahora puede aprovechar las herramientas de administración de AWS recursos con Amazon GameLift Servers recursos. En particular, ahora se asignan valores de nombre de recurso de Amazon (ARN) a todos los recursos clave de Amazon GameLift Servers (compilaciones, scripts, flotas, colas de sesiones de juegos, configuraciones de emparejamiento y conjuntos de reglas de emparejamiento). Un ARN de recurso proporciona un identificador coherente que es único en todas las AWS regiones. Se pueden usar para crear políticas de permisos para recursos específicos AWS Identity and Access Management (IAM). A los recursos se les asigna ahora un ARN, así como el identificador de recurso preexistente, que no es específico de la región.

Además, los recursos Amazon GameLift Servers ahora admiten el etiquetado. Puede utilizar etiquetas para organizar los recursos, crear políticas de permisos de IAM para gestionar el acceso a grupos de recursos, personalizar los desgloses de AWS costes, etc. Al administrar etiquetas para recursos Amazon GameLift Servers, use las acciones `TagResource()`, `UntagResource()` y `ListTagsForResource()` de la API de Amazon GameLift Servers.

Más información:

- [TagResource](#) en la Referencia de la API de Amazon GameLift Servers
- [Etiquetado de recursos de AWS](#) en la Referencia general de AWS
- [Nombres de recursos de Amazon](#) en la Referencia general de AWS

14 de noviembre de 2019: nuevas CloudFormation plantillas y actualizaciones en la región de China (Pekín)

Versiones del SDK actualizadas: AWS SDK [1.7.210](#)

CloudFormation plantillas para Amazon GameLift Servers

Amazon GameLift Servers los recursos ahora se pueden crear y administrar mediante CloudFormation. Las plantillas de CloudFormation construcción y flota existentes se han actualizado para adaptarlas a los recursos actuales, y ahora hay nuevas plantillas disponibles para los guiones, las colas, las configuraciones de emparejamiento y los conjuntos de reglas de emparejamiento. CloudFormation Las plantillas simplifican considerablemente la tarea de gestionar grupos de AWS recursos relacionados, especialmente cuando se despliegan juegos en varias regiones.

Más información:

- [Referencia de tipos de recurso de Amazon GameLift Servers](#) en la Guía del usuario de AWS CloudFormation
- [Gestione los recursos de Amazon GameLift Servers alojamiento mediante CloudFormation](#) en la Guía para desarrolladores de Amazon GameLift Servers.

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