Developer Guide

Amazon GameLift Streams

Amazon GameLift Streams: Developer Guide

Table of Contents

What is Amazon GameLift Streams?

With Amazon GameLift Streams, game publishers and others can provide on-demand, low-latency streaming experiences to players and viewers globally. Amazon GameLift Streams uses its own streaming technology combined with the AWS global infrastructure to operate and maintain application streaming at scale. Publishers have the flexibility to provision both on-demand and reserved streaming resources to effectively manage capacity and costs.

Topics

Features

Amazon GameLift Streams offers these key features:

- High-quality streaming to players around the world by deploying computing resources in multiple AWS Regions.
- Streaming technology that delivers real-time gameplay experiences with minimal player-to-cloud latency. You can choose the optimal stream quality for your content, with up to 1080p resolution and 60 frames per second (fps).
- Scaling tools to adjust your streaming capacity to meet customer demand. For example, with these tools you can keep game streaming costs in line while maintaining enough capacity to accommodate new players into stream sessions quickly.
- Stream performance analysis using the Amazon GameLift Streams console to track metrics, view stream logs, and review data on stream resource usage.
- Direct streaming of Windows and Linux-based games with little to no modification.
- Amazon GameLift Streams SDK to help you integrate your existing identity services, storefront, and client applications.

How to get started with Amazon GameLift Streams

If you're a first-time Amazon GameLift Streams user, we recommend that you begin with the following topics:

- Setting up Amazon GameLift Streams covers one-time setup tasks, including getting an AWS account with user access and setting up the software you need for Amazon GameLift Streams.
- Starting your first stream in Amazon GameLift Streams guides you through the critical steps in the content streaming workflow. Starting with your content, such as a game build, prepare it for streaming, provision some Amazon GameLift Streams streaming cloud resources, set up a backend service and client, and initiate a streaming session.

Accessing Amazon GameLift Streams

You can create, access, and manage your application content and streaming resources with the following tools:

- AWS Management Console – Provides a web interface that you can use to create and manage your Amazon GameLift Streams applications and stream groups.
- AWS Command Line Interface (AWS CLI) – Provides commands for a broad set of AWS services and is supported on Windows, Mac, and Linux. For more information on this tool, see the AWS Command Line Interface page.
- AWS SDK – Provides language-specific APIs and takes care of connection details, such as calculating signatures, handling request retries, and error handling. For documentation on the Amazon GameLift Streams service API, see the Amazon GameLift Streams API Reference. For more general information on the AWS SDK, see Tools to Build on AWS.

For additional information, see Regions, quotas, and limitations.

Setting up Amazon GameLift Streams

To start using the Amazon GameLift Streams service with your projects, complete these basic setup tasks. If you already have an AWS account and a user under that account that you want to use with Amazon GameLift Streams, you can skip these steps.

For more information on what you can do with an AWS account, see Getting started with AWS.

After you've completed these setup tasks, we recommend that you go to Starting your first stream in Amazon GameLift Streams and step through the tutorial, which covers the entire workflow for getting your content streaming in a web client.

Topics

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Get programmatic access

In addition to your user sign-in credentials for the AWS Management Console, you need credentials for programmatic access, such as when working with the AWS Command Line Interface (AWS CLI). Programmatic credentials consist of a two-part set of access keys. Use one of the following methods to generate your access keys:

- Method 1 – If you're using an administrative user created with the IAM Identity Center, see Getting IAM role credentials for AWS CLI access to generate temporary security credentials for short-term access to AWS resources. When following these instructions, make sure you're signed in through your account's AWS access portal URL with your administrative user name and password (not your root user).
- Method 2 – If you're using an existing IAM user and you haven't yet transitioned to using the IAM Identity Center, see Managing access keys for IAM users (console) to generate long-term credentials for your user.

> **Note**
>
> As a best practice, use temporary credentials instead of long-term access keys. Temporary credentials include an access key ID, a secret access key, and a security token that indicates when the credentials expire. For more information, see Best practices for managing AWS access keys in the *AWS General Reference*.

Get Amazon GameLift Streams materials

You can get started without any additional materials by using the in-console streaming experience. We recommend this as a starting point because it allows you to evaluate how your application performs on the Amazon GameLift Streams without setting up any additional infrastructure. For more information, proceed to Getting started with Amazon GameLift Streams.

When you're ready to build your own Amazon GameLift Streams integration, you can download the Amazon GameLift Streams Web SDK, available in the Resources section of the Getting Started product page. This is the recommended starting point for developing your own Amazon GameLift Streams solution because it contains everything you need to get up and running quickly. It contains the JavaScript AWS SDK with Amazon GameLift Streams APIs, a sample web server that uses the Amazon GameLift Streams service, and a sample web client for connecting to streams.

Then, set up the AWS CLI by completing the instructions in the next section.

For more information about setting up your own Amazon GameLift Streams solution, refer to Amazon GameLift Streams backend service and web client.

Download the AWS CLI

To use Amazon GameLift Streams with your content, we recommend that you get the AWS Command Line Interface (AWS CLI). The AWS CLI is an open source tool that gives you equivalent AWS SDK functionality by running commands from a terminal program.

1. Download and install the latest version of the AWS CLI for your operating system. See these install instructions in the *AWS Command Line Interface User Guide*.
2. Configure the tool with your user access credentials and other preferences, as described in Setting up the AWS CLI. With this configuration, you won't have to explicitly specify your credentials and other settings with every command.
3. Use the following command to verify your installation and get a list of available Amazon GameLift Streams commands:

```
aws gameliftstreams help
```

Set up billing alerts

A stream group incurs cost per active stream capacity per second. To make sure your cost and usage stays within your budget, see Create billing alerts to monitor usage.

Getting started with Amazon GameLift Streams

This section can help you successfully get started streaming your applications and games through Amazon GameLift Streams. The topics in this section cover the end-to-end process—from uploading your application to Amazon GameLift Streams, to testing how your content performs in a stream. It also covers important steps to help you prepare for streaming, such as choosing the right runtime and stream class configuration to optimize performance and cost.

Topics

Choosing a configuration in Amazon GameLift Streams

This guide can help you choose the optimal runtime environment and configuration settings for streaming your applications and games through Amazon GameLift Streams. The configuration settings directly impact your content's performance and the costs associated with running it on Amazon GameLift Streams. There are several options to support a wide variety of applications and graphical fidelity.

You can find the complete list of configuration options in Configuration options.

The following key terms can help you understand how these configuration options work together:

- *Runtimes* refer to the underlying operating system and software environment that will execute your application on Amazon GameLift Streams. The main runtime environment options are Windows, Linux, and Proton.
- *Stream classes* represent the different hardware configurations available within Amazon GameLift Streams, varying in operating system, CPU, GPU, RAM, and other specifications.
- *Multitenancy* enables multiple users to share the same underlying hardware resources, which can be a cost-effective option for applications that don't require maximum hardware capabilities. A stream class with multi-tenancy can host multiple streams for the cost of one resource. "High" stream classes have 1:2 tenancy, while "Ultra" stream classes have 1 tenancy.

When setting up your Amazon GameLift Streams configuration, the runtime environment you choose determines the specific stream class options that are compatible and available to you. Matching your application's requirements with the right runtime environment and stream class is key to optimizing performance and cost-efficiency in Amazon GameLift Streams.

The cost to stream depends on the stream class. For a detailed list of cost, refer to the Amazon GameLift Streams Pricing page.

Starting point

Depending on your application, these are good starting points to get started streaming. Later, you can explore other configuration options to optimize the cost.

For Windows applications

We recommend using the Microsoft Windows Server 2022 Base runtime environment for Microsoft Windows applications. There are two hardware configurations available for this runtime, the NVIDIA-based `gen5n_win2022` and `gen4n_win2022` stream classes. In this environment, Amazon GameLift Streams supports games and other 3D applications using DirectX 11 or DirectX 12, and game engines including Unity 2022.3, Unreal Engine 4.27, and Unreal Engine 5 up through 5.5.

This combination of runtime environment and stream classes provides a predictable, well-supported configuration with the highest compatibility and best performance for your Windows-based content.

For Linux applications

Use the Ubuntu 22.04 LTS runtime environment for applications built to run natively on Linux. To optimize for performance, choose one of the NVIDIA Ultra stream classes (`gen5n_ultra` or `gen4n_ultra`). To optimize for cost, choose one of the NVIDIA High stream classes (`gen5n_high` or `gen4n_high`) that support *multitenancy*— a cost-effective option where multiple concurrent stream sessions share the same compute resources.

Cost optimizations

While the starting point recommendations are a great place to begin, you might want to consider other configuration options to optimize for cost while maintaining good performance.

Use Proton runtime environment

Many Windows applications can run in the Proton runtime environment. Proton is a game-optimized compatibility layer that runs on Linux. The stream class options for this runtime include powerful GPU resources running on NVIDIA hardware, with support for DirectX 11 and, beginning with Proton 8.0-5, DirectX 12. Some stream classes also come with *multitenancy* — a cost-effective option that supports multiple concurrent stream sessions running on shared compute resources. Visit the Proton wiki for more details about this option. If you choose to explore running your application on Proton, we recommend that you start your testing using Proton 8.0-5.

> **Important**
>
> The compatibility of your Windows application in a Proton runtime environment depends on your specific application requirements. For example, Proton 8.0-5 has better support than Proton 8.0-2c for Unreal Engine 5. We strongly recommend thoroughly testing this runtime in your local environment to ensure optimal performance. Use our Proton troubleshooting guide to help you in this effort.

Compile Windows applications to Linux

Another cost-saving option is to compile your Windows application to run natively on Linux. Test the application on your end first to ensure that the Linux-compiled version of your application performs as needed. If you're application successfully runs on Linux, then you can follow the Amazon GameLift Streams configuration options for Linux applications.

For information about compiling Unreal Engine applications to Linux, refer to the Cross-Compile Toolchain section in Unreal Engine's developer guide.

Deciding on a configuration

To determine the best runtime environment option, consider the following key questions.

1. **What platform is your application or game built for?** If you have a Windows application, the Windows runtime environment is the simplest to set up. If your application is built for Linux, the Linux runtime environment is the most straightforward. To save costs for streaming a Windows application, you can explore the Proton runtime environment or compile the application to Linux.

2. **How important is performance versus cost for your use case?** The Windows runtime environment may offer the best performance, but it can be more expensive to run. Comparitively, the Proton runtime environment is more cost-effective, though you might experience slightly lower performance or potential compatibility issues. This is because Windows-based applications might require certain functionality that is not yet fully supported in the available Proton runtimes. As a result, you could experience functional or graphical differences when running your application on the Proton environment. We recommend that you test your application on the different runtime environments to evaluate the performance and cost trade-offs.
3. **What are the graphical requirements of your application?** The graphical requirements of your application can help determine which stream class configuration is most appropriate. If your application demands high-performance GPUs, you should consider using stream classes with greater amounts of video memory (VRAM) and system memory (RAM). Conversely, if your application can operate effectively at a lower graphical fidelity, you might save on costs by using stream classes that support multi-tenancy. This allows multiple users to share the same underlying hardware resources.
4. **How much effort are you willing to invest in the setup?** The simplest way to set up your application is to run it natively using the Windows or Linux runtimes, since they are more likely to be compatible with your application out-of-the-box. In contrast, the Proton runtime environment will require more hands-on testing to identify the optimal Proton configuration for your needs. Consider the time and resources you can allocate to the setup and testing process when deciding between the runtime environment options.
5. **Have you tested your application on the various runtime environments and stream classes?** We recommend testing your content on different runtime environments and stream classes to see how it performs. This helps you determine the best fit based on factors like stability, graphics quality, feature functionality, and input responsiveness.

How your configuration choices impact next steps

The configuration you select directly impacts the next phases of setting up your streaming environment. Specifically:

- **Creating an Amazon GameLift Streams application**: When you upload your game or application to Amazon GameLift Streams, you'll need to specify the runtime environment you want to use. This choice will determine the type of stream group you can use.

- **Linking to a stream group**: If you already have an existing stream group, your runtime environment choice will need to match the configuration of that group. For example, if you select the Windows runtime, you can only link your application to a stream group that's set up for Windows applications.
- **Creating a stream group**: When you create a new stream group, you must choose a stream class that's compatible with your chosen runtime. The stream class you choose should match the graphics requirements and compute power that your application requires.

By understanding how the configuration settings you choose influences these subsequent steps, you can better plan your overall streaming implementation and ensure a smooth integration process.

Next steps

Depending on the configuration you've chosen, there are a few different approaches you can take to set up your application for streaming.

If you've selected the Windows or Linux runtime

For Windows or Linux runtimes, the next steps are to set up streaming in Amazon GameLift Streams and then test the stream. For more information, proceed to Starting your first stream in Amazon GameLift Streams.

If you're considering using Proton

An application's compatibility with Proton depends on the application's specific requirements. Therefore, we recommend that you test your application on different Proton versions before bringing it to Amazon GameLift Streams. This helps you identify the Proton setup that provides the best performance and compatibility for your needs. By testing outside of Amazon GameLift Streams, you can validate the application's performance and functionality, and debug issues that are specific to the runtime. For information, see Troubleshoot compatibility with Proton for Amazon GameLift Streams.

When you've selected a specific Proton configuration, you're ready to set up streaming in Amazon GameLift Streams. For more information, proceed to Starting your first stream in Amazon GameLift Streams.

Configuration options

Runtime environments

Runtimes refer to the underlying operating system and software environment that executes your application on Amazon GameLift Streams. The main runtime options are Windows, Linux, and Proton. You specify the runtime environment in Step 2: Configure your application for Amazon GameLift Streams of the getting started worflow.

Proton is a compatibility layer that enables many Windows applications to run in a Linux-based environment. If you plan to use Proton, we recommand that you test how your application runs on a local machine. For more information, refer to Troubleshoot compatibility with Proton for Amazon GameLift Streams.

Runtime	Description
Microsoft Windows Server 2022 Base	Compatible with Windows applications.
Ubuntu 22.04 LTS	Compatible with Linux applications.
Proton 8.0-5	Compatible with Windows applications. Based on the Proton experimental_8.0 branch.
Proton 8.0-2c	Compatible with Windows applications. Based on the Proton experimental_8.0 branch.

Limitations

Gamepad support is not available on Ubuntu 22.04 LTS. Other runtime environments support gamepads, depending on the end user's operating system and browser. For more information, see Supported browsers and input.

Stream classes

Stream classes represent the different hardware configurations available within Amazon GameLift Streams, varying in CPU, GPU, RAM, and other specifications. You specify the stream class in

Step 3: Manage how Amazon GameLift Streams streams your application of the getting started workflow.

Stream class	Amazon EC2 configuration	Description
`gen5n_win_2022`	Windows runtime on a g5.2xlarge Amazon EC2 instance	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applications, and anti-cheat technology. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
`gen5n_high`	Linux runtime on a g5.2xlarge Amazon EC2 instance with 2:1 tenancy	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 12 GB. Tenancy: Supports up to two concurrent stream sessions.
`gen5n_ultra`	Linux runtime on a g5.2xlarge Amazon EC2 instance	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.

Stream class	Amazon EC2 configuration	Description
`gen4n_win2022`	Windows runtime on a g4dn.2xlarge Amazon EC2 instance	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applications, and anti-cheat technology. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.
`gen4n_high`	Linux runtime on a g4dn.2xlarge Amazon EC2 instance with 2:1 tenancy	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 8 GB. Tenancy: Supports up to two concurrent stream sessions.
`gen4n_ultra`	Linux runtime on a g4dn.2xlarge Amazon EC2 instance	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.

Runtime environment and stream class compatibility

Runtime environment	Compatible stream classes
Windows	`gen5n_win2022` `gen4n_win2022`

Linux (Ubuntu 22.04 LTS)	`gen5n_high`
	`gen5n_ultra`
	`gen4n_high`
	`gen4n_ultra`

Proton	`gen5n_high`
	`gen5n_ultra`
	`gen4n_high`
	`gen4n_ultra`

Starting your first stream in Amazon GameLift Streams

This tutorial walks you through the steps to get started with Amazon GameLift Streams to stream your application or game. Amazon GameLift Streams runs your application and streams them directly to your end-users' web browser. You'll learn how to upload and configure the application you want to stream, and how to manage the way Amazon GameLift Streams streams. By the end, you will test how your application streams on Amazon GameLift Streams by interacting with it directly in the Amazon GameLift Streams console.

Before you start, understand Amazon GameLift Streams pricing.

You can find the cost of Amazon GameLift Streams in the Pricing page. To learn more, refer to Managing usage and bills for Amazon GameLift Streams.
You incur cost for using Amazon GameLift Streams, specifically when you:

- Create an Amazon GameLift Streams application in Step 2: Configure your application for Amazon GameLift Streams
- Create a stream group in Step 3: Manage how Amazon GameLift Streams streams your application

Do not skip Step 5: Clean up (don't skip). To avoid unnecessary charges after you're done trying Amazon GameLift Streams, you must clean up all your resources.

Topics

Prerequisites

Complete the following tasks before you start the tutorial.

- Complete all the steps in Setting up Amazon GameLift Streams. Specifically, you must have an AWS account with the proper credentials for programmatic access. You do not need to set up AWS CLI at this time — you will complete the following steps using the AWS Console.
- Get a version of your application content files with no digital rights management (DRM). Collect the files required to run the application, including executables and assets, into a folder, but *do not* compress the folder.

Step 1: Upload your application to an Amazon S3 bucket

Amazon GameLift Streams uses Amazon Simple Storage Service (Amazon S3) to store your application or game files in the cloud and access it for streaming. In this step, you upload your application files to an Amazon S3 bucket. Complete this step in the Amazon S3 Console.

Note

The Amazon S3 storage class that Amazon GameLift Streams requires is the default **S3 Standard**. Other storage classes like **S3 Glacier** or objects being moved to **Infrequent Access** or **Archive Access** by **S3 Intelligent-Tiering** are not supported by Amazon GameLift Streams.

To optimize storage cost, you can delete the application from your S3 bucket after you complete Step 2: Configure your application for Amazon GameLift Streams and the application is in **Ready** status.

Application limitations

Name	Default	Adjuste	Description
Files per application	30,000 files	Yes*	The maximum number of files that you can have in an application, in this account.
Single file size	80 GiB	No	The maximum size of a single file in an application. Note that a gibibyte (GiB) equals 1024*1024*1024 bytes.
Application size	100 GiB	Yes*	The maximum total size of an Amazon GameLift Streams application, in this account. Note that a gibibyte (GiB) equals 1024*1024*1024 bytes.

*To request an increase, sign in to the AWS Management Console and open the Service Quotas console to Amazon GameLift Streams, where you can review your current quotas in the **Applied account-level quota value** column and submit a request to increase a value.

To upload your application to Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create an Amazon S3 bucket. Enter a bucket name and select an AWS Region. This region must be the same as the application and stream group that you will create later. See AWS Regions and remote locations supported by Amazon GameLift Streams for a list of AWS Regions where Amazon GameLift Streams is available. For the remaining fields, keep the default settings.

   For more instructions, refer to Creating a bucket in the *Amazon Simple Storage Service User Guide*.
3. Open the new bucket and upload the folder with your application files.

> **Warning**
>
> You must upload your application files as an uncompressed folder. Don't upload a `.zip` folder.

> **Warning**
>
> Ensure that the application files you uploaded are the correct ones, and are within the application file size limitations. If you want to update your files later, you must repeat Step 2: Configure your application for Amazon GameLift Streams, which can take a few minutes.

Step 2: Configure your application for Amazon GameLift Streams

What is an application in Amazon GameLift Streams?

An application in Amazon GameLift Streams is a game or other software that Amazon GameLift Streams can stream and users can play or interact with. It is a resource that contains your game or application files, as well as configuration settings to run it. This is also referred to as an *Amazon GameLift Streams application* when the context is ambiguous.

In this step, you configure the application you want to stream with Amazon GameLift Streams by creating an application. Complete this step in the Amazon GameLift Streams console.

To create an Amazon GameLift Streams application using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console. Choose the same AWS Region as the Amazon S3 bucket where you uploaded your set of files. For more information, refer to Choosing a Region in the *AWS Management Console Getting Started Guide*.
2. In the navigation bar, choose **Applications** and then choose **Create application**.
3. In **Runtime settings**, enter the following:

   - **Runtime environment**

     This is the runtime environment to run your application on. Amazon GameLift Streams can run on either Windows, Ubuntu 22.04 LTS, or Proton.

     You cannot edit this field after the creation workflow.

     Choose from one of the following runtime environments.

     - For Linux applications:
       - Ubuntu 22.04 (`UBUNTU, 22_04_LTS`)
     - For Windows applications:
       - Microsoft Windows Server 2022 Base (`WINDOWS, 2022`)
       - Proton 8.0-5 (`PROTON, 20241007`)
       - Proton 8.0-2c (`PROTON, 20230704`)

     Review the descriptions and use the comparison checklist to help you select the optimal runtime environment for your application.
4. In **General settings**, enter the following:

   a. **Description**

      This is a human-readable label for your application. This value does not have to be unique. For best practice, use a meaningful description, name, or label for the application. You can edit this field at any time.

   b. **Base path**

This is the Amazon S3 URI to your application's root folder in the Amazon S3 bucket. The folder and any subfolders should contain your build executable and any supporting files.

A valid URI is the bucket prefix that contains all the files needed to run and stream the application. For example, a bucket called `mygamebuild` contains three complete versions of the game build files, each in a separate folder. You want to stream the build in the folder `mygamebuild-EN101`. In this example, the URI is `s3://amzn-s3-demo-bucket/mygamebuild-EN101`.

You cannot edit this field after the creation workflow.

c. **Executable launch path**

This is the Amazon S3 URI to the executable file that Amazon GameLift Streams will stream. The file must be contained within the application's root folder.

You cannot edit this field after the creation workflow.

5. (Optional) In **Application log path**, enter the following:

a. **Application log path**

This is the path (or paths) to the application folder or file which contains logs that you want to save. Specify each log path relative to your application base path. If you use this feature, then at the end of every stream session, Amazon GameLift Streams will copy the file(s) that you specify to the Amazon S3 bucket that you name. The copy operation is not performed recursively in an application folder's subfolders.

You can edit this field at any time.

b. **Application log output**

This is the URI to the Amazon S3 bucket where Amazon GameLift Streams will copy application log files. This field is required if you specify an application log path.

You can edit this field at any time.

To save log files on your behalf, Amazon GameLift Streams must be given permission to your S3 bucket for saving. If you let Amazon GameLift Streams create the bucket for logging, the permission policy will be applied automatically upon creation. If you provide your own bucket, you will need to apply the permission policy, yourself.

Bucket permission policy template

Copy the following policy code and apply it to the bucket that you want to use for application logs. Be sure to replace **amzn-s3-demo-bucket** with the name of your existing S3 bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gameliftstreams.amazonaws.com"
        ]
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

6. (Optional) In **Tags**, assign tags to this application.

   Tags are labels that can help you organize your AWS resources. For more information, refer to Tagging your AWS resources.

   For example to track application versions, use a tag such as `application-version : my-game-1121`.

7. Choose **Create application**.

Amazon GameLift Streams takes a few minutes to prepare your application. In the **Applications** page, the new application is in **Processing** status. When your application is in **Ready** status, you can go to the next step, Step 3: Manage how Amazon GameLift Streams streams your application.

If the request returns an error, or if the application is created but in **Error** status, make sure that you're working with user credentials that include access to both Amazon S3 and Amazon GameLift Streams.

> **Note**
>
> When an application is in **Ready** status, you can safely delete the application files in your Amazon S3 bucket, without affecting your new application. This also helps optimize storage cost. For more information, see Delete an application.

For more information, refer to Prepare an application in Amazon GameLift Streams.

Step 3: Manage how Amazon GameLift Streams streams your application

What is a stream group?

Manage how Amazon GameLift Streams streams your applications by using a stream group. A stream group is a collection of compute resources that Amazon GameLift Streams uses to stream your application to end users. When you create a stream group, you specify the type of hardware to use, such as the graphical processing unit (GPU). You must select a default application to stream. However, you can also link additional applications. Depending on your expected users, you also specify the stream capacity, the number of concurrent streams you want to support at one time. Then, Amazon GameLift Streams allocates compute resources in the Region where you create the stream group.

With your application ready, the next thing you need is compute resources for Amazon GameLift Streams to stream it. In this step, you manage how Amazon GameLift Streams streams your application by creating a stream group. Complete this step in the Amazon GameLift Streams console.

To create a stream group in the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console. Choose the AWS Region where you want to create your stream group. This Region must be the same as that of the application that you want to stream with the stream group. For more information, refer to Choosing a Region in the *AWS Management Console Getting Started Guide*.
2. To open the creation workflow, in the navigation pane, choose **Stream groups**, and then choose **Create stream group**.

3. In **Define stream group**, enter the following:

   a. **Description**

      A human-readable label for your stream group. This value doesn't have to be unique. As a best practice, use a meaningful description, name, or label for the stream group. You can edit this field at any time.

   b. **Tags**

      Tags are labels that can help you organize your AWS resources. For more information, refer to Tagging your AWS resources.

4. In **Select stream class**, choose a stream class for the stream group.

   - **Stream class options**

     The type of compute resources to run and stream applications with. This choice impacts the quality of the streaming experience and the cost. You can specify only one stream class per stream group. Choose the class that best fits your application.

Stream class	Description
`gen5n_win2022`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applicati ons, and anti-cheat technology. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
`gen5n_high`	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 12 GB. Tenancy: Supports up to two concurrent stream sessions.
`gen5n_ultra`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA A10G Tensor GPU.

Stream class	Description
	Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
`gen4n_win2022`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applicati ons, and anti-cheat technology. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.
`gen4n_high`	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 8 GB. Tenancy: Supports up to two concurrent stream sessions.
`gen4n_ultra`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.

To continue, choose **Next**.

5. In **Link application**, choose the application that you want to stream. If you change your mind, you can edit the stream group to add additional applications later. You can only link as application that's in `Ready` status and has a runtime that's compatible with the stream class you've chosen. By default, these are the only applications that are shown in the table. To see all applications in `Ready` status, choose `All runtimes` in the drop down list.

> **Note**
>
> If you don't see your application listed, then check the current AWS Region setting. You can only link an application to a stream group that's in the same Region.

To continue, choose **Next**.

6. In **Configure stream settings**, under **Locations and capacity**, choose one or more locations where your stream group will have capacity to stream your application. By default, the region where you create the stream group, known as the *primary location*, has already been added to your stream group and cannot be removed. You can add additional locations by checking the box next to each location that you want to add. For lower latency and better quality streaming, you should choose locations closer to your users.

   For each location, you can specify its *streaming capacity*. Stream capacity represents the number of concurrent streams that can be active at a time. You set stream capacity per location in each stream group. At each location, there are two types of capacity: always-on capacity and on-demand capacity.

   - **Always-on capacity:** The streaming capacity that is pre-allocated and ready to handle stream requests without delay. You pay for this capacity whether it's in use or not. Best for quickest time from streaming request to streaming session.
   - **On-demand capacity:** The streaming capacity that Amazon GameLift Streams can allocate in response to stream requests, and then de-allocate when the session has terminated. This offers a cost control measure at the expense of a greater stream start time (typically under 5 minutes).

   You can increase or decrease your total stream capacity at any time to meet changes in user demand for a location by adjusting either capacity. Amazon GameLift Streams fulfills streaming requests using the idle, pre-allocated resources in the always-on capacity pool if any are available. If all always-on capacity is in use, Amazon GameLift Streams will provision additional compute resources up to the maximum number specified in on-demand capacity. As allocated capacity scales, the change is reflected in your total cost for the stream group.

   Linked applications will automatically be replicated to each enabled location. An application must finish replicating in a remote location before the remote location can host a stream. To

check on the replication status, open the stream group after it has been created and refer to the **Replication status** column in the table of linked applications. Click on the current status to see the replication status for each added location.

> **Note**
>
> Application data will be stored in all enabled locations including the primary location for this stream group. Stream session data will be stored in both the primary location and the location where the streaming occurred.

7. In **Review and create stream group**, verify your stream group configuration and make changes as needed. When everything is correct, choose **Create stream group**.

For more information, refer to Manage streaming with an Amazon GameLift Streams stream group.

Step 4: Test your stream in Amazon GameLift Streams

What is a stream session?

Refers to the stream itself. This is an instance of a stream that Amazon GameLift Streams transmits from the server to the end-user. A stream session runs on a compute resource, or stream capacity, that a stream group has allocated. Also referred to as *stream* for short.

You can see how your application streams by running it directly in the Amazon GameLift Streams console. When you start a stream, Amazon GameLift Streams uses one of the compute resources that your stream group allocates. So, you must have available capacity in your stream group.

To test your stream in the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. You can test a stream in several ways. Start from the **Stream groups** page or **Test stream** page and follow these steps:

   a. Select a stream group that you want to use to stream.

   b. If you're starting from the **Stream groups** page, choose **Test stream**. If you're starting from the **Test stream** page, select **Choose**. This opens the **Test stream** configuration page for the selected stream group.

   c. In **Linked applications**, select an application.

d. In **Location**, choose a location with available capacity.

e. (Optional) In **Program configurations**, enter command-line arguments or environment variables to pass to the application as it launches.

f. Confirm your selection, and choose **Test stream**.

3. After your stream loads, you can do the following actions in your stream:

a. To connect input, such as your mouse, keyboard, and gamepad, choose **Attach input**. You automatically attach your mouse when you move the cursor into the stream window.

b. To have files that were created during the streaming session exported to an Amazon S3 bucket at the end of the session, choose **Export files** and specify the bucket details. Exported files can be found on the **Sessions** page.

c. To view the stream in fullscreen, choose **Fullscreen**. Press **Escape** to reverse this action.

4. To end the stream, choose **Terminate session**. When the stream disconnects, the stream capacity becomes available to start another stream.

Step 5: Clean up (don't skip)

> ⚠ **Avoid unnecessary cost**
>
> A stream groups incurs costs when it has allocated capacity, even if that capacity is unused. To avoid unnecessary costs, scale your stream group capacities to your required size. We suggest during development you scale your always-on capacity to zero when not in use. For more information, refer to Best practices to manage Amazon GameLift Streams costs.

After you complete the tutorial and no longer need to stream your application, follow these steps to clean up your Amazon GameLift Streams resources.

To delete a stream group using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. To view a list of your existing stream groups, in the navigation pane, choose **Stream groups**.
3. Choose the name of the stream group that you want to delete.
4. On the stream group detail page, choose **Delete**.
5. In the **Delete** dialog box, confirm the delete action.

Amazon GameLift Streams begins releasing compute resources and deleting the stream group. During this time, the stream group is in **Deleting** status. After Amazon GameLift Streams deletes the stream group, you can no longer retrieve it.

To delete an application using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Applications** to view a list of your existing applications. Choose the application you want to delete.
3. In the application detail page, choose **Delete**.
4. In the **Delete** dialog box, confirm the delete action.

Amazon GameLift Streams begins deleting the application. During this time, the application is in `Deleting` status. After Amazon GameLift Streams deletes the application, you can no longer retrieve it.

For more information, refer to Delete a stream group and Delete an application.

Managing your streams with Amazon GameLift Streams

This section provides detailed information about how to stream with Amazon GameLift Streams. Learn about the streaming resources (an *application* and *stream group*), the properties to scale your streaming (*stream capacity* and *locations*), and the stream itself (a *stream session*). You can handle all of the tasks required to set up streaming with Amazon GameLift Streams by using the Amazon GameLift Streams console or Amazon GameLift Streams CLI commands.

If it's your first time using Amazon GameLift Streams, then refer to Starting your first stream in Amazon GameLift Streams, which walks you through the whole workflow.

Topics

Key concepts

Application

An application in Amazon GameLift Streams is a game or other software that Amazon GameLift Streams can stream and users can play or interact with. It is a resource that contains your game or application files, as well as configuration settings to run it. This is also referred to as an *Amazon GameLift Streams application* when the context is ambiguous.

Multi-application stream groups

A stream group that's linked to multiple applications. This many-to-one relationship allows you to stream multiple applications by using the same configuration that you've set up in a single stream group . When you start a stream session, you specify any linked applications. Then, Amazon GameLift Streams streams that application by using available stream capacity in this stream group.

Multi-location stream groups

A stream group that's configured to host applications and stream sessions from multiple locations, in addition to the primary location (the AWS Region where you created the stream group). You manage capacity for each location.

Multi-tenancy

Tenancy refers to how many concurrent streams can be supported by a single compute resource in Amazon GameLift Streams. *Multi-tenancy* is a feature that enables multiple users to share the same underlying hardware resources, which can be a cost-effective option for applications that don't require maximum hardware capabilities. A stream class with multi-tenancy can host multiple streams for the cost of one resource. "High" stream classes support multi-tenancy, allowing two applications to run concurrently on a single compute resource, while "Ultra" stream classes do not support multi-tenancy.

Stream group

Manage how Amazon GameLift Streams streams your applications by using a stream group. A stream group is a collection of compute resources that Amazon GameLift Streams uses to stream your application to end users. When you create a stream group, you specify the type of hardware to use, such as the graphical processing unit (GPU). You must select a default application to stream. However, you can also link additional applications. Depending on your expected users, you also specify the stream capacity, the number of concurrent streams you want to support at one time. Then, Amazon GameLift Streams allocates compute resources in the Region where you create the stream group.

Stream capacity

Represents the number of concurrent streams that can be active at a time. You set stream capacity per location in each stream group. At each location, there are two types of capacity: always-on capacity and on-demand capacity.

Stream session

Refers to the stream itself. This is an instance of a stream that Amazon GameLift Streams transmits from the server to the end-user. A stream session runs on a compute resource, or stream capacity, that a stream group has allocated. Also referred to as *stream* for short.

Prepare an application in Amazon GameLift Streams

To set up streaming with Amazon GameLift Streams, first you upload the game or other application that you want to stream, then you configure an application resource within Amazon GameLift Streams to define metadata about your game. An Amazon GameLift Streams application consists of the files you uploaded (executable and any supporting files) and a configuration that instructs Amazon GameLift Streams what executable to run when streaming.

Each Amazon GameLift Streams application represents a single version of your content. If you have multiple versions, you must create a separate application for each version. After you create an application, you cannot update the files. If you need to update the executable or any supporting files, you must create a new Amazon GameLift Streams application.

Before you upload

Before you create an Amazon GameLift Streams application, verify that your game adheres to the following limitations.

Name	Default	Adjuse	Description
Files per application	30,000 files	Yes*	The maximum number of files that you can have in an application, in this account.
Single file size	80 GiB	No	The maximum size of a single file in an application. Note that a gibibyte (GiB) equals 1024*1024*1024 bytes.
Application size	100 GiB	Yes*	The maximum total size of an Amazon GameLift Streams application, in this account. Note that a gibibyte (GiB) equals 1024*1024*1024 bytes.

*To request an increase, sign in to the AWS Management Console and open the Service Quotas console to Amazon GameLift Streams, where you can review your current quotas in the **Applied account-level quota value** column and submit a request to increase a value.

> **Note**
>
> To save yourself time and effort, verify that the files you are ready to upload are the correct version of your application. While you can upload new versions later, you will need to repeat the Create an application step for each version.

Upload your application to an Amazon S3 bucket

Now that you have prepared your game for Amazon GameLift Streams, it's time to upload it to an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account.

> **Note**
>
> The Amazon S3 storage class that Amazon GameLift Streams requires is the default **S3 Standard**. Other storage classes like **S3 Glacier** or objects being moved to **Infrequent Access** or **Archive Access** by **S3 Intelligent-Tiering** are not supported by Amazon GameLift Streams.
> To optimize storage cost, you can delete the application from your S3 bucket after you complete Create an application and the application is in **Ready** status.

To upload your application to Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create an Amazon S3 bucket. Enter a bucket name and select an AWS Region. This region must be the same as the application and stream group that you will create later. See AWS Regions and remote locations supported by Amazon GameLift Streams for a list of AWS Regions where Amazon GameLift Streams is available. For the remaining fields, keep the default settings.

   For more instructions, refer to Creating a bucket in the *Amazon Simple Storage Service User Guide*.
3. Open the new bucket and upload the folder with your application files.

> **Warning**
>
> You must upload your application files as an uncompressed folder. Don't upload a `.zip` folder.

Create an application

An Amazon GameLift Streams application is a resource that contains the game or other software you want to stream and the settings to run it. When you create an application, you provide the path to the application files you uploaded in your Amazon S3 bucket.

Amazon GameLift Streams does not auto-sync your game files from the Amazon S3 bucket you created. If you want to update your game files, you must create a new Amazon GameLift Streams application.

Console

To create an Amazon GameLift Streams application using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console. Choose the same AWS Region as the Amazon S3 bucket where you uploaded your set of files. For more information, refer to Choosing a Region in the *AWS Management Console Getting Started Guide*.
2. In the navigation bar, choose **Applications** and then choose **Create application**.
3. In **Runtime settings**, enter the following:

   - **Runtime environment**

     This is the runtime environment to run your application on. Amazon GameLift Streams can run on either Windows, Ubuntu 22.04 LTS, or Proton.

     You cannot edit this field after the creation workflow.

     Choose from one of the following runtime environments.

     - For Linux applications:

- Ubuntu 22.04 (`UBUNTU, 22_04_LTS`)

- For Windows applications:
  - Microsoft Windows Server 2022 Base (`WINDOWS, 2022`)
  - Proton 8.0-5 (`PROTON, 20241007`)
  - Proton 8.0-2c (`PROTON, 20230704`)

Review the descriptions and use the comparison checklist to help you select the optimal runtime environment for your application.

4. In **General settings**, enter the following:

   a. **Description**

   This is a human-readable label for your application. This value does not have to be unique. For best practice, use a meaningful description, name, or label for the application. You can edit this field at any time.

   b. **Base path**

   This is the Amazon S3 URI to your application's root folder in the Amazon S3 bucket. The folder and any subfolders should contain your build executable and any supporting files.

   A valid URI is the bucket prefix that contains all the files needed to run and stream the application. For example, a bucket called `mygamebuild` contains three complete versions of the game build files, each in a separate folder. You want to stream the build in the folder `mygamebuild-EN101`. In this example, the URI is `s3://amzn-s3-demo-bucket/mygamebuild-EN101`.

   You cannot edit this field after the creation workflow.

   c. **Executable launch path**

   This is the Amazon S3 URI to the executable file that Amazon GameLift Streams will stream. The file must be contained within the application's root folder.

   You cannot edit this field after the creation workflow.

5. (Optional) In **Application log path**, enter the following:

   a. **Application log path**

This is the path (or paths) to the application folder or file which contains logs that you want to save. Specify each log path relative to your application base path. If you use this feature, then at the end of every stream session, Amazon GameLift Streams will copy the file(s) that you specify to the Amazon S3 bucket that you name. The copy operation is not performed recursively in an application folder's subfolders.

You can edit this field at any time.

b. **Application log output**

This is the URI to the Amazon S3 bucket where Amazon GameLift Streams will copy application log files. This field is required if you specify an application log path.

You can edit this field at any time.

To save log files on your behalf, Amazon GameLift Streams must be given permission to your S3 bucket for saving. If you let Amazon GameLift Streams create the bucket for logging, the permission policy will be applied automatically upon creation. If you provide your own bucket, you will need to apply the permission policy, yourself.

Bucket permission policy template

Copy the following policy code and apply it to the bucket that you want to use for application logs. Be sure to replace **amzn-s3-demo-bucket** with the name of your existing S3 bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gameliftstreams.amazonaws.com"
        ]
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
```

```
}
```

6. (Optional) In **Tags**, assign tags to this application.

   Tags are labels that can help you organize your AWS resources. For more information, refer to Tagging your AWS resources.

   For example to track application versions, use a tag such as `application-version : my-game-1121`.

7. Choose **Create application**.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To create an application using the AWS CLI

In your AWS CLI use the CreateApplication command, customized for your content.

```
aws gameliftstreams create-application \
    --description "MyGame v1" \
    --runtime-environment '{"Type":"PROTON", "Version":"20241007"}' \
    --executable-path "launcher.exe" \
    --application-source-uri "s3://amzn-s3-demo-bucket/example"
```

where

- `description`:

  This is a human-readable label for your application. This value does not have to be unique. For best practice, use a meaningful description, name, or label for the application. You can edit this field at any time.

- `runtime-environment`:

  This is the runtime environment to run your application on. Amazon GameLift Streams can run on either Windows, Ubuntu 22.04 LTS, or Proton.

  You cannot edit this field after the creation workflow.

Choose from one of the following runtime environments.

- For Linux applications
  - Ubuntu 22.04 LTS(`Type=UBUNTU, Version=22_04_LTS`)
- For Windows applications
  - Microsoft Windows Server 2022 Base (`Type=WINDOWS, Version=2022`)
  - Proton 8.0-2c (`Type=PROTON, Version=20230704`)
  - Proton 8.0-5 (`Type=PROTON, Version=20241007`)

- `executable-path`:

  This is the path to the executable file that Amazon GameLift Streams will stream. Specify a path relative to the `application-source-uri`. The file must be contained within the application's root folder.

  You cannot edit this field after the creation workflow.

- `application-source-uri`:

  This is the Amazon S3 URI to your application's root folder in the Amazon S3 bucket. The folder and any subfolders should contain your build executable and any supporting files.

  A valid URI is the bucket prefix that contains all the files needed to run and stream the application. For example, a bucket called `mygamebuild` contains three complete versions of the game build files, each in a separate folder. You want to stream the build in the folder `mygamebuild-EN101`. In this example, the URI is `s3://amzn-s3-demo-bucket/mygamebuild-EN101`.

  You cannot edit this field after the creation workflow.

If the request is successful, Amazon GameLift Streams returns a response similar to the following:

```
{
    "Arn": "arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6",
    "Description": "MyGame v1",
    "RuntimeEnvironment": {
        "Type": "PROTON",
        "Version": "20241007"
    },
```

```
    "ExecutablePath": "launcher.exe",
    "ApplicationSourceUri": "s3://amzn-s3-demo-bucket/example",
    "Id": "a-9ZY8X7Wv6",
    "Status": "PROCESSING",
    "CreatedAt": "2022-11-18T15:47:11.924000-08:00",
    "LastUpdatedAt": "2022-11-18T15:47:11.924000-08:00"
}
```

To check the status of your application, call the GetApplication command, as shown in the following example.

```
aws gameliftstreams get-application /
    --identifier a-9ZY8X7Wv6
```

Amazon GameLift Streams takes a few minutes to prepare your application. During this time, the new application is in **Processing** status. When your application is in **Ready** status, you can go to the next step, Create a stream group.

If the request returns an error, or if the application is created but placed in an **Error** status, make sure that you're working with user credentials that include access to both Amazon S3 and Amazon GameLift Streams.

> **Note**
>
> When an application is in **Ready** status, Amazon GameLift Streams has successfully copied your application files to its private Amazon S3 bucket. You can delete your original application files without affecting your new application. This also helps you optimize on storage cost. For more information, see Delete an application.

Edit an application

You can update the settings for any application in **Ready** status. If you make changes to an existing application, these changes impact the streaming behavior for both new and existing stream groups.

Console

To edit an application in the Amazon GameLift Streams console

1. In the navigation bar, choose **Applications** to view a list of your existing applications. Choose the application you want to edit.
2. In the application details page, locate the section that contains the settings you want to change and choose **Edit** or **Manage tags** accordingly.
3. You can change the following settings:

   Description

   This is a human-readable label for your application. This value does not have to be unique. For best practice, use a meaningful description, name, or label for the application. You can edit this field at any time.

   Application log path

   This is the path (or paths) to the application folder or file which contains logs that you want to save. Specify each log path relative to your application base path. If you use this feature, then at the end of every stream session, Amazon GameLift Streams will copy the file(s) that you specify to the Amazon S3 bucket that you name. The copy operation is not performed recursively in an application folder's subfolders.

   You can edit this field at any time.

   Application log output

   This is the URI to the Amazon S3 bucket where Amazon GameLift Streams will copy application log files. This field is required if you specify an application log path.

   You can edit this field at any time.

   To save log files on your behalf, Amazon GameLift Streams must be given permission to your S3 bucket for saving. If you let Amazon GameLift Streams create the bucket for logging, the permission policy will be applied automatically upon creation. If you provide your own bucket, you will need to apply the permission policy, yourself.

   For more information, see Application log bucket permission policy.

Tags

Tags are labels that can help you organize your AWS resources. For more information, refer to Tagging your AWS resources.

For example to track application versions, use a tag such as `application-version : my-game-1121`.

4. Choose **Save changes**. The Amazon GameLift Streams console returns to the application details page, displaying the updated settings.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To edit an application using the AWS CLI

In your AWS CLI use the UpdateApplication command, customized for your content.

```
aws gameliftstreams update-application \
    --identifier a-9ZY8X7Wv6 \
    --description "MyGame v2" \
    --application-log-paths '[".\\logs"]' \
    --application-log-output-uri "s3://amzn-s3-demo-bucket/mygame"
```

where

- `identifier`: The application to edit.

  This value is an Amazon Resource Name (ARN) or ID that uniquely identifies the application resource.

  ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6`

  ID example: `a-9ZY8X7Wv6`

- `description`:

This is a human-readable label for your application. This value does not have to be unique. For best practice, use a meaningful description, name, or label for the application. You can edit this field at any time.

- `application-log-paths`:

  This is the path (or paths) to the application folder or file which contains logs that you want to save. Specify each log path relative to your application base path. If you use this feature, then at the end of every stream session, Amazon GameLift Streams will copy the file(s) that you specify to the Amazon S3 bucket that you name. The copy operation is not performed recursively in an application folder's subfolders.

  You can edit this field at any time.

- `application-log-output-uri`:

  This is the URI to the Amazon S3 bucket where Amazon GameLift Streams will copy application log files. This field is required if you specify an application log path.

  You can edit this field at any time.

  To save log files on your behalf, Amazon GameLift Streams must be given permission to your S3 bucket for saving. If you let Amazon GameLift Streams create the bucket for logging, the permission policy will be applied automatically upon creation. If you provide your own bucket, you will need to apply the permission policy, yourself.

  For more information, see Application log bucket permission policy.

Delete an application

Delete an application if you no longer need it. This action permanently deletes the application, including the application content files stored with Amazon GameLift Streams. However, this does not delete the original files that you uploaded to your Amazon S3 bucket; you can delete these any time after Amazon GameLift Streams creates an application, which is the only time Amazon GameLift Streams accesses your Amazon S3 bucket.

You can only delete an application that meets the following conditions:

- The application is in the **Ready** or **Error** state.

- The application is not the default application of any stream groups. You must first delete the stream group by using the Amazon GameLift Streams console, or by using DeleteStreamGroup in the Amazon GameLift Streams API.
- The application is not linked to any stream groups. You must first unlink the stream group by using the Amazon GameLift Streams console, or by using DisassociateApplications in the Amazon GameLift Streams API.
- An application is not streaming in any ongoing stream session. You must wait until the client ends the stream session or call TerminateStreamSession in the Amazon GameLift Streams API to end the stream.

Console

To delete an application using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Applications** to view a list of your existing applications. Choose the application you want to delete.
3. In the application detail page, choose **Delete**.
4. In the **Delete** dialog box, confirm the delete action.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To delete an application using the AWS CLI

In your AWS CLI use the DeleteApplication command, customized for your content.

```
aws gameliftstreams delete-application \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:application/
a-9ZY8X7Wv6
```

where

- `identifier`: The application to delete.

This value is an Amazon Resource Name (ARN) or ID that uniquely identifies the application resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6`

ID example: `a-9ZY8X7Wv6`

Amazon GameLift Streams begins deleting the application. During this time, the application is in `Deleting` status. After Amazon GameLift Streams deletes the application, you can no longer retrieve it.

Application log bucket permission policy

If you provide your own application log Amazon S3 bucket, you will need to apply a permission policy to the bucket so that Amazon GameLift Streams can save log files to the bucket. Use the following template to update the permissions in Amazon S3.

Bucket permission policy template

Copy the following policy code and apply it to the bucket that you want to use for application logs. Be sure to replace **amzn-s3-demo-bucket** with the name of your existing S3 bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "gameliftstreams.amazonaws.com"
        ]
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

Linked stream groups

If you want to stream multiple applications by using the same pool of compute resources, you can link multiple applications to the same stream group. Similarly, if you want to stream an application by using different sets of compute resources, you can link an application to multiple stream groups.

For more information about linking applications to stream groups, refer to Overview of multi-application stream groups.

Manage streaming with an Amazon GameLift Streams stream group

After you set up an Amazon GameLift Streams application, you're ready to manage and deploy compute resources to run and stream your application. An Amazon GameLift Streams *stream group* represents a collection of these compute resources. You specify the maximum number of concurrent streams to support by scaling the stream capacity.

Amazon GameLift Streams allocates compute resources in the AWS Region where you create a stream group. You can also add remote locations to a stream group and manage capacity per location. It's a best practice to host stream sessions in locations that are geographically near your end users. This helps minimize latency and improve stream quality. For more information, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

In a stream group, you must specify the Amazon GameLift Streams applications that the stream group can stream. A single application can be in multiple stream groups, so you can set up different configurations or types of compute resources to stream the same application. For example, to provide two graphics-quality options for streaming an application, you can set up two stream groups with different configurations and link them to the same application.

Conversely, a single stream group can have multiple applications: the *default application*, which you set when you create the stream group, and a set of *linked applications*. For more information, refer to Overview of multi-application stream groups.

How you relate your stream groups and applications together depends on your use case, but the relationship can be many-to-many.

Topics

- About stream capacity

- About locations
- Create a stream group
- Edit general settings
- Edit capacity
- Add locations in a stream group
- Delete locations in a stream group
- Delete a stream group
- Linked applications
- Stream group maintenenace

About stream capacity

You manage the number of streams you can deliver concurrently to end-users by setting the stream group's capacity, or *stream capacity*. Stream capacity represents the amount of resources that are ready to stream. At each location, there are two types of capacity: always-on capacity and on-demand capacity.

- **Always-on capacity:** The streaming capacity that is pre-allocated and ready to handle stream requests without delay. You pay for this capacity whether it's in use or not. Best for quickest time from streaming request to streaming session.
- **On-demand capacity:** The streaming capacity that Amazon GameLift Streams can allocate in response to stream requests, and then de-allocate when the session has terminated. This offers a cost control measure at the expense of a greater stream start time (typically under 5 minutes).

If you have a stream group with an always-on capacity set to 100, this means the stream group has enough resources to stream to 100 end-users concurrently. You can increase or decrease the stream capacity at any time to meet changes in user demand. You set stream capacity per location.

Scaling the capacity reflects in your total cost for the stream group. Ensure that you set up billing alerts to manage your Amazon GameLift Streams costs. Refer to Create billing alerts to monitor usage.

To request a capacity change, edit your stream group settings and enter new values for always-on capacity and/or on-demand capacity. When Amazon GameLift Streams receives this request,

the service begins working to make allocated stream capacity match the new requested stream capacity. It does this by provisioning new hosting resources or shutting down existing ones. The process of increasing resources can take some time, because Amazon GameLift Streams might have to wait for resources to become available before allocating them to your stream group.

About locations

The location is where Amazon GameLift Streams allocates compute resources to host your application and stream to users. For lower latency and better quality, you should choose locations closer to your users. By default, you can stream from the AWS Region where you created your stream group, known as the *primary location*. Additionally, a stream group can extend its coverage to stream from other supported locations.

For a complete list of supported locations, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

Multi-location stream group

A stream group that's configured to host applications and stream sessions from multiple locations, in addition to the primary location (the AWS Region where you created the stream group). You manage capacity for each location.

Create a stream group

Console

To create a stream group in the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console. Choose the AWS Region where you want to create your stream group. This Region must be the same as that of the application that you want to stream with the stream group. For more information, refer to Choosing a Region in the *AWS Management Console Getting Started Guide*.
2. To open the creation workflow, in the navigation pane, choose **Stream groups**, and then choose **Create stream group**.
3. In **Define stream group**, enter the following:

    a. **Description**

A human-readable label for your stream group. This value doesn't have to be unique. As a best practice, use a meaningful description, name, or label for the stream group. You can edit this field at any time.

b. **Tags**

Tags are labels that can help you organize your AWS resources. For more information, refer to Tagging your AWS resources.

4. In **Select stream class**, choose a stream class for the stream group.

- **Stream class options**

The type of compute resources to run and stream applications with. This choice impacts the quality of the streaming experience and the cost. You can specify only one stream class per stream group. Choose the class that best fits your application.

Stream class	Description
`gen5n_win2022`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applications, and anti-cheat technology. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
`gen5n_high`	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 12 GB. Tenancy: Supports up to two concurrent stream sessions.

Stream class	Description
gen5n_ultra	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
gen4n_win2022	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applications, and anti-cheat technology. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.
gen4n_high	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 8 GB. Tenancy: Supports up to two concurrent stream sessions.
gen4n_ultra	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.

To continue, choose **Next**.

5. In **Link application**, choose the application that you want to stream. If you change your mind, you can edit the stream group to add additional applications later. You can only link as application that's in `Ready` status and has a runtime that's compatible with the stream class you've chosen. By default, these are the only applications that are shown in the table. To see all applications in `Ready` status, choose `All runtimes` in the drop down list.

   > **Note**
   >
   > If you don't see your application listed, then check the current AWS Region setting. You can only link an application to a stream group that's in the same Region.

   To continue, choose **Next**.

6. In **Configure stream settings**, under **Locations and capacity**, choose one or more locations where your stream group will have capacity to stream your application. By default, the region where you create the stream group, known as the *primary location*, has already been added to your stream group and cannot be removed. You can add additional locations by checking the box next to each location that you want to add. For lower latency and better quality streaming, you should choose locations closer to your users.

   For each location, you can specify its *streaming capacity*. Stream capacity represents the number of concurrent streams that can be active at a time. You set stream capacity per location in each stream group. At each location, there are two types of capacity: always-on capacity and on-demand capacity.

   - **Always-on capacity:** The streaming capacity that is pre-allocated and ready to handle stream requests without delay. You pay for this capacity whether it's in use or not. Best for quickest time from streaming request to streaming session.
   - **On-demand capacity:** The streaming capacity that Amazon GameLift Streams can allocate in response to stream requests, and then de-allocate when the session has terminated. This offers a cost control measure at the expense of a greater stream start time (typically under 5 minutes).

   You can increase or decrease your total stream capacity at any time to meet changes in user demand for a location by adjusting either capacity. Amazon GameLift Streams fulfills streaming requests using the idle, pre-allocated resources in the always-on capacity pool if any are available. If all always-on capacity is in use, Amazon GameLift Streams

will provision additional compute resources up to the maximum number specified in on-demand capacity. As allocated capacity scales, the change is reflected in your total cost for the stream group.

Linked applications will automatically be replicated to each enabled location. An application must finish replicating in a remote location before the remote location can host a stream. To check on the replication status, open the stream group after it has been created and refer to the **Replication status** column in the table of linked applications. Click on the current status to see the replication status for each added location.

> **Note**
>
> Application data will be stored in all enabled locations including the primary location for this stream group. Stream session data will be stored in both the primary location and the location where the streaming occurred.

7. In **Review and create stream group**, verify your stream group configuration and make changes as needed. When everything is correct, choose **Create stream group**.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To create a stream group using the AWS CLI

In your AWS CLI use the CreateStreamGroup command, customized for your content.

```
aws gameliftstreams create-stream-group \
    --description "Test_gen4_high" \
    --default-application-identifier arn:aws:gameliftstreams:us-
west-2:111122223333:application/a-9ZY8X7Wv6 \
    --stream-class gen4n_high \
    --location-configurations '[{"LocationName": "us-east-1",
 "AlwaysOnCapacity": 10, "OnDemandCapacity": 20}]'
```

where

`description`:

A human-readable label for your stream group. This value doesn't have to be unique. As a best practice, use a meaningful description, name, or label for the stream group. You can edit this field at any time.

`default-application-identifier`

The Amazon Resource Name (ARN) value or ID assigned to an Amazon GameLift Streams application resource. The application must be in `READY` status.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6`

ID example: `a-9ZY8X7Wv6`

`stream-class`

Stream class options

The type of compute resources to run and stream applications with. This choice impacts the quality of the streaming experience and the cost. You can specify only one stream class per stream group. Choose the class that best fits your application.

Stream class	Description
`gen5n_win2022`	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applications, and anti-cheat technology. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
`gen5n_high`	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 12 GB. Tenancy: Supports up to two concurrent stream sessions.

Stream class	Description
gen5n_ultra	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA A10G Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 24 GB. Tenancy: Supports one concurrent stream session.
gen4n_win2022	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Runs applications on Microsoft Windows Server 2022 Base and supports DirectX 12 and DirectX 11. Supports Unreal Engine up through version 5.5, 64-bit applicati ons, and anti-cheat technology. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.
gen4n_high	(NVIDIA, high) Supports applications with moderate-to-high 3D scene complexity. Uses NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 4. RAM: 16 GB. VRAM: 8 GB. Tenancy: Supports up to two concurrent stream sessions.
gen4n_ultra	(NVIDIA, ultra) Supports applications with extremely high 3D scene complexity. Uses dedicated NVIDIA T4 Tensor GPU. Resources per application: vCPUs: 8. RAM: 32 GB. VRAM: 16 GB. Tenancy: Supports one concurrent stream session.

`location-configurations`

A set of locations to add to this stream group, and their capacities. By default, if no capacities are specified, Amazon GameLift Streams will allocate enough stream capacity to start only one stream. For a complete list of locations that Amazon GameLift Streams supports, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

Valid values for capacity depend on the stream class, as follows:

- `high`: Enter non-negative even numbers.
- `ultra`: Enter non-negative numbers.

If the request is successful, then Amazon GameLift Streams returns a response similar to the following:

```
{
    "Arn": "arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4",
    "Description": "Test_gen4_high",
    "DefaultApplication": {
        "Id": "a-9ZY8X7Wv6"
    },
    "StreamClass": "gen4n_high",
    "Id": "sg-1AB2C3De4",
    "Status": "ACTIVATING",
    "LastUpdatedAt": "2024-11-18T15:49:01.482000-08:00",
    "CreatedAt": "2024-11-18T15:49:01.482000-08:00"
}
```

Amazon GameLift Streams begins searching for unallocated computing resources and provisioning them for the new stream group, which can take several minutes. During this time, the new stream group is in **Activating** status.

You can adjust the stream group's capacity while in **Activating** or **Active** status. For more information, refer to Edit capacity.

When the stream group is in **Active** status, it's ready to deploy resources for streaming. To start streaming, refer to Start stream sessions with Amazon GameLift Streams.

Edit general settings

Amazon GameLift Streams groups the following settings together in the console under **Stream group settings**: **Status**, **Stream group ID**, **Description**, **Stream group ARN**, and **Stream class**. Of these, the only one that you can update without creating a new stream group is **Description**.

Console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Stream groups** to view a list of your existing stream groups. Choose the stream group you want to edit.
3. In the stream group detail page, choose **Edit settings**.
4. To update the description, enter a new value.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To edit a stream group's description using the AWS CLI

In your AWS CLI use the UpdateStreamGroup command, customized for your content.

```
aws gameliftstreams update-stream-group \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4 \
    --description "MyGame - Ultra"
```

where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

`description`

A human-readable label for your stream group. This value doesn't have to be unique. As a best practice, use a meaningful description, name, or label for the stream group. You can edit this field at any time.

Edit capacity

Scale your stream groups by adjusting the capacity for each location.

Refer to Amazon GameLift Streams service quotas to learn more about stream group capacity quotas per AWS account, per location, and how to increase these quotas.

Console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Stream groups** to view a list of your existing stream groups. Choose the stream group you want to edit.
3. In the stream group detail page, choose **Edit configuration**.
4. For each location, enter new always-on and on-demand stream capacity values in the relevant cells in the table. You can request an increase or decrease in capacity. The limits on capacity settings are as follows:
   - Use the following values based on stream group's stream class setting:
     - For "high" stream classes, set capacity to multiples of two (2).
     - For "ultra" stream classes, set capacity to multiples of one (1).
   - If you set the always-on capacity value to zero, this means that the stream group won't allocate any hosts to stream.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To edit stream capacity using the AWS CLI

In your AWS CLI use the UpdateStreamGroup command, customized for your content.

```
aws gameliftstreams update-stream-group \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4 \
    --location-configurations '[{"LocationName": "us-east-1",
 "AlwaysOnCapacity": 50}, \
        {"LocationName": "ap-northeast-1", "AlwaysOnCapacity": 50,
 "OnDemandCapacity": 20}]'
```

where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

`location-configurations`

A set of locations to add to this stream group, and their capacities. For a complete list of locations that Amazon GameLift Streams supports, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

Valid values for capacity depend on the stream class, as follows:

- `high`: Enter non-negative even numbers.
- `ultra`: Enter non-negative numbers.

When you update the stream group's desired capacity, Amazon GameLift Streams will begin processing your request, which can take a some time. During this time, Amazon GameLift Streams works to allocate or release resources in the stream group, to meet the desired always-on stream capacity you set. You can view the provisioning status of your stream capacity by viewing the **Stream group details** page in the Amazon GameLift Streams console, or by calling `get-stream-group` using the Amazon GameLift Streams CLI.

When your stream group is in **Active** status and it has available stream capacity, you can start streaming. For more information, refer to Start stream sessions with Amazon GameLift Streams.

Stream group scaling behavior

When you scale down capacity, Amazon GameLift Streams waits until the host is idle before releasing it. Since a host can support 1 or 2 sessions, the host is idle only when all active sessions on the host end. A stream session ends when the user ends their session or the session times out. Therefore, in extreme situations when existing sessions are allowed to reach the maximum possible duration, it may take up to 24 hours to reach the desired capacity. If you want to end all stream sessions, you can delete the stream group or use the TerminateStreamSession API to end active sessions.

The "high" stream classes may take longer to scale down than the "ultra" stream class. This is because the "high" stream class uses shared resources, providing two streams from a single host. When you scale capacity down, Amazon GameLift Streams waits until both sessions end before releasing the host. In contrast, the "ultra" stream class has one session per host. So when the one session ends, Amazon GameLift Streams can release the host immediately.

Add locations in a stream group

Console

To add locations to a stream group using the Amazon GameLift Streams console

1. In the navigation bar, choose **Stream groups** to view a list of your existing stream groups. Choose the stream group you want to add new locations to.
2. In the **Stream group details** page, choose **Edit configuration**.
3. Select the checkbox next to the location(s) you want to add to this stream group, and then set their capacities.
4. Review the summary of your selected locations, including the cost for stream capacity. Choose **Save** to confirm your selection.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To add locations to a stream group using the AWS CLI

In your AWS CLI use the AddStreamGroupLocations command, customized for your content.

```
aws gameliftstreams add-stream-group-locations \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4
    --location-configurations '[{"LocationName": "us-east-1", "AlwaysOnCapacity": 2,
 "OnDemandCapacity": 2}]'
```

where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

`location-configurations`

A set of locations to add to this stream group, and their capacities. For a complete list of locations that Amazon GameLift Streams supports, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

Valid values for capacity depend on the stream class, as follows:

- `high`: Enter non-negative even numbers.
- `ultra`: Enter non-negative numbers.

When your application has completed replicating to the new location(s) and your stream group has available stream capacity, you can start streaming from the new location(s). For more information on streaming, refer to Start stream sessions with Amazon GameLift Streams. Amazon GameLift Streams will begin processing your request, which can take a few minutes. During this time, Amazon GameLift Streams works to replicate your application and allocate compute resources in the new locations. You can view the status of the replication from the **Linked applications** section of the **Stream group details** page by hovering over the status in the **Replication status** column.

Delete locations in a stream group

To stop using compute resources from specific locations, you can delete the locations in your stream group. This decreases the total stream capacity in your stream group. However, you can still increase the stream capacity in the remaining locations.

You cannot delete the primary location of a stream group. However, if you don't want compute resources in that location, then you can set the stream capacities to zero.

> **Warning**
>
> When you delete a location in a stream group, Amazon GameLift Streams disconnects active streams in that location, which stops the stream of any connected end users.

Console

To delete locations from a stream group using the Amazon GameLift Streams console

1. In the navigation pane, choose **Stream groups** to view a list of your existing stream groups.
2. Choose the name of the stream group that you want to delete locations from.
3. In the **Stream group details** page, choose **Edit configuration**.
4. Uncheck the checkbox next to the name of the location that you want to delete.
5. Choose **Save**.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To delete locations from a stream group using the AWS CLI

In your AWS CLI use the RemoveStreamGroupLocations command, customized for your content.

```
aws gameliftstreams remove-stream-group-locations \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4
    --locations us-east-1 eu-central-1
```

where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

`locations`

A set of locations to delete from this stream group. For a complete list of locations that Amazon GameLift Streams supports, refer to AWS Regions and remote locations supported by Amazon GameLift Streams.

Delete a stream group

You can delete a stream group that's in any status. This action permanently deletes the stream group and releases its compute resources. If there are streams in process, then this action stops them and your end users can no longer view the stream.

As a best practice, before you delete a stream group, check for streams in process and take steps to stop them.

Console

To delete a stream group using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. To view a list of your existing stream groups, in the navigation pane, choose **Stream groups**.
3. Choose the name of the stream group that you want to delete.
4. On the stream group detail page, choose **Delete**.

5. In the **Delete** dialog box, confirm the delete action.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To delete your stream group using the AWS CLI

In your AWS CLI use the DeleteStreamGroup command, customized for your content.

```
aws gameliftstreams delete-stream-group \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4
```

where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

Amazon GameLift Streams begins releasing compute resources and deleting the stream group. During this time, the stream group is in **Deleting** status. After Amazon GameLift Streams deletes the stream group, you can no longer retrieve it.

Linked applications

If you want to stream multiple application using the same pool of compute resources, then you can link multiple applications to the same stream group. Similarly, if you want to stream an application using different sets of compute resources, then you can link an application to multiple stream groups.

For more information about linking applications to stream groups, refer to Overview of multi-application stream groups.

Stream group maintenenace

For a stream group to receive new service updates and fixes, you must recreate the stream group. As a best practice, we recommend that you recreate stream groups every 3-4 weeks.

Whenever a feature is released that requires a new stream group to use it, you will see a "Maintenance required" message at the top of the stream group's detail page to inform you that it is outdated. Recreating a stream group is a manual process, but to help you do it, use the **Create Stream Group** button in the message to start the process. Some of the fields will be filled in for you.

Stream group maintenance is also required when the stream group is over 180 days old. You will no longer be able to link new applications to these older stream groups until they are recreated.

Overview of multi-application stream groups

A *multi-application stream group* is a stream group that's linked to multiple applications. This allows you to stream multiple applications by using the same set of compute resources in a single stream group.

Most of the time, you may want a single stream group with multiple applications. A common use case for multi-application stream groups is to release different versions of your game. For example, suppose that you created a stream group and set the default application to the original version of your game. Then, suppose you create additional applications that contains other versions of your game and link them to the stream group. Since these applications are associated with the same stream group, you only have to manage a single set of compute resources, or stream capacity, to stream all of these games. This means, regardless of which application an end-user streams, the application runs on a compute resource from the same set that this stream group has allocated.

Here are other possible real-life examples:

- A game streaming platform that offers different streaming tiers to customers.
- A quality assurance team that's testing multiple versions of a game.
- To simplify stream capacity management by using a single stream group for multiple applications.
- To enable a set of applications to stream from the same pool of stream capacity.

Limitations and requirements

You can only associate applications to stream groups that have compatible runtime environments and stream classes. For more information, refer to Runtime environment and stream class compatibility.

About linking applications to a stream group

Among the set of applications in a stream group, one of the applications is considered as the *default application*. The default application is required and immutable—you only set it when you create a stream group and cannot change it to a different application. All other linked applications are additional applications that you want this stream group to run and stream. When it comes to streaming, there is no difference between a default application and other linked applications.

There are a few things to keep in mind when working with a default application and other linked applications:

- The default application is immutable. This means that the stream group will always be linked to the application you selected when creating a stream group.
- The default application is required to create a stream group. This means you must have already created an application before you create a stream group.
- The same application can be the default application for multiple stream groups.
- The set of linked applications is mutable until the stream group is 180 days old. In practical terms, this means that you can link and unlink applications until the stream group is 180 days old. After that, you will only be able to unlink applications from a stream group throughout the remainder of the group's lifecycle.

Link an application to a stream group

When you link, or associate, an application to a stream group, the stream group will be able to stream the application. You can link and unlink additional applications to a stream group until it reaches 180 days old. After that, you will only be able to unlink applications from a stream group throughout the remainder of the group's lifecycle.

Important

You cannot link an application to a stream group that is over 180 days old. To associate different applications to the stream group, you will first need to recreate it. For instructions on how to recreate a stream group, refer to Stream group maintenenace.

Before you link an application, ensure that the stream group is in **Active** status.

Console

To link using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Stream groups** to view a list of your existing stream groups.
3. Select a stream group to view its details.
4. In **Linked applications**, choose **Link application**.
5. Select an application that you want to link. Confirm your selection and choose **Link application**.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To link an application(s) using the AWS CLI

In your AWS CLI use the AssociateApplications command, customized for your content.

```
aws gameliftstreams associate-applications \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4 \
    --application-identifiers a-9ZY8X7Wv6 a-1Z78C7Wv6
```

where

- `identifier`:

A stream group to link these applications with.

This value can be an Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

- `application-identifiers`:

  A set of applications that you want to link with this stream group.

  This value is a Amazon Resource Name (ARN) or ID that uniquely identifies the application resource.

  ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6`

  ID example: `a-9ZY8X7Wv6`

Unlink an application from a stream group

When you unlink, or disassociate, an application from a stream group, you can no longer stream this application by using that stream group's allocated compute resources. Any streams in process will continue until they terminate, which helps avoid interrupting an end-user's stream. Amazon GameLift Streams will not initiate new streams using this stream group. The unlink action does not affect the stream capacity of a stream group.

You can only unlink an application if it's not a *default application* of the stream group. You set the default application when you first create a stream group.

Console

To unlink using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Stream groups** to view a list of your existing stream groups.
3. Select a stream group to view its details.

4. In **Linked applications**, select the application(s) that you want to unlink. Choose **Unlink applications**.
5. In the **Unlink applications** dialog, confirm the unlink action and choose **Unlink**.

CLI

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To unlink an application(s) using the AWS CLI

In your AWS CLI use the DisassociateApplications command, customized for your content.

```
aws gameliftstreams disassociate-applications \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4 \
    --application-identifiers a-9ZY8X7Wv6 a-1Z78C7Wv6
```

where

- `identifier`:

  A stream group to unlink these applications from.

  This value can be an Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

  ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

  ID example: `sg-1AB2C3De4`
- `application-identifiers`:

  A set of applications that you want to unlink from this stream group.

  This value is a Amazon Resource Name (ARN) or ID that uniquely identifies the application resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:application/a-9ZY8X7Wv6`

ID example: `a-9ZY8X7Wv6`

Multi-application stream group quota

Quota name	Default quota	Is adjustable	Description
Number of links in multi-application stream groups	100 links	No	The maximum number of associations for a single stream group or application. For example, a single stream group can link to up to 100 applications, and vice versa.

Start stream sessions with Amazon GameLift Streams

This section covers stream sessions, the actual instance of a stream where an end user or player can interact with your application or play your game. You'll learn about how to test your own stream session and understand the stream session life cycle.

For launching stream sessions to end users, you must integrate Amazon GameLift Streams into your own service. For more information, refer to Amazon GameLift Streams backend service and web client.

About stream sessions

The prerequisites to start a stream session are an application and a stream group that has available active capacity. A stream session runs on one of the compute resources, or stream capacity, that a stream group has allocated. When you start a stream, you must specify a stream group and choose to stream the default application or a linked application. By default, Amazon GameLift Streams streams the default application. To stream a linked application, specify both the stream group and application ARN when you start a stream session.

When you successfully start a stream session, you receive a unique identifier for that stream session. Then, you use that ID to connect the stream session to an end user. For more information, refer to StartStreamSession in the *Amazon GameLift Streams API Reference*.

Testing a stream

The most direct way for you to test how your application streams is through the Amazon GameLift Streams console. When you start a stream, Amazon GameLift Streams uses one of the compute resources that your stream group allocates. So, you must have available capacity in your stream group.

To test your stream in the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. You can test a stream in several ways. Start from the **Stream groups** page or **Test stream** page and follow these steps:

   a. Select a stream group that you want to use to stream.

   b. If you're starting from the **Stream groups** page, choose **Test stream**. If you're starting from the **Test stream** page, select **Choose**. This opens the **Test stream** configuration page for the selected stream group.

   c. In **Linked applications**, select an application.

   d. In **Location**, choose a location with available capacity.

   e. (Optional) In **Program configurations**, enter command-line arguments or environment variables to pass to the application as it launches.

   f. Confirm your selection, and choose **Test stream**.
3. After your stream loads, you can do the following actions in your stream:

   a. To connect input, such as your mouse, keyboard, and gamepad, choose **Attach input**. You automatically attach your mouse when you move the cursor into the stream window.

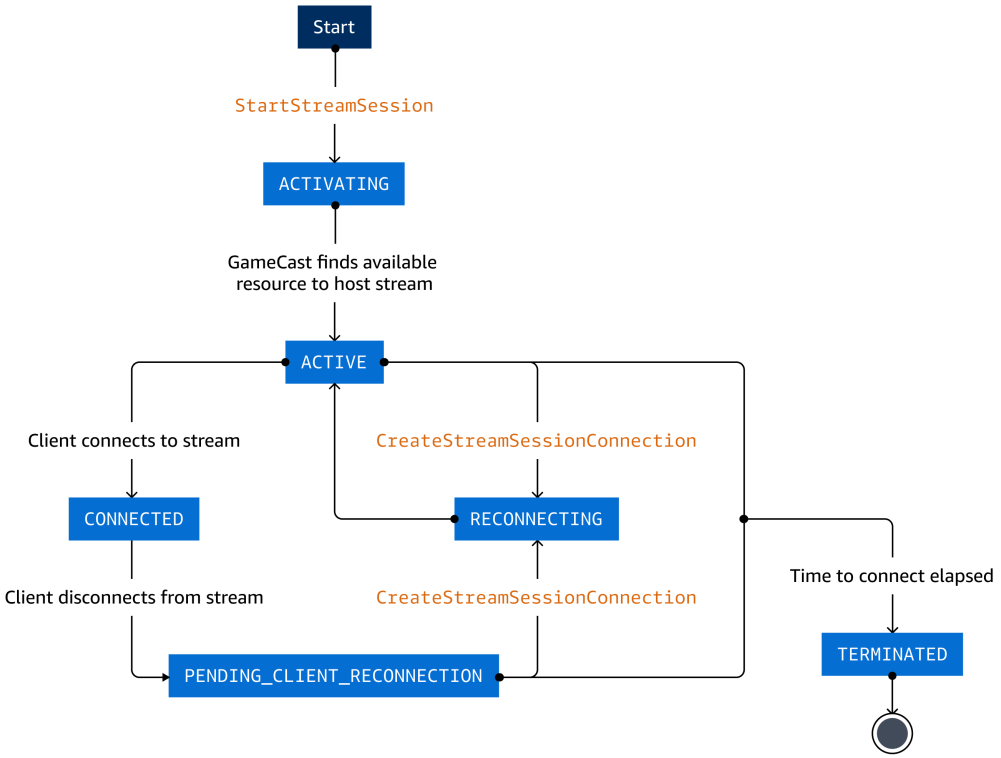
   b. To have files that were created during the streaming session exported to an Amazon S3 bucket at the end of the session, choose **Export files** and specify the bucket details. Exported files can be found on the **Sessions** page.

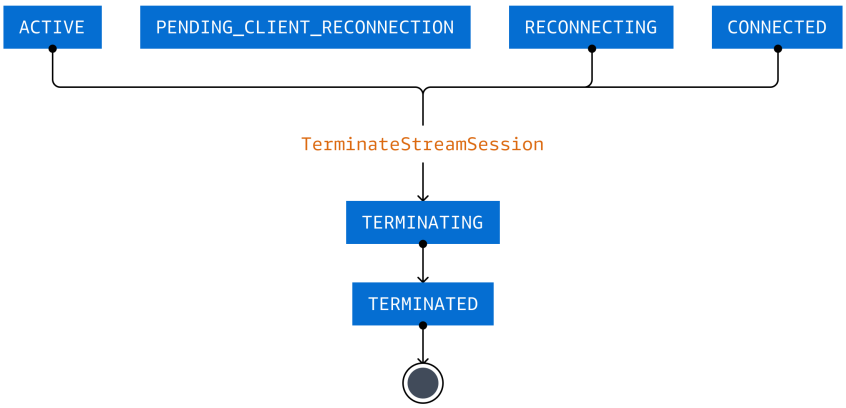
   c. To view the stream in fullscreen, choose **Fullscreen**. Press **Escape** to reverse this action.
4. To end the stream, choose **Terminate session**. When the stream disconnects, the stream capacity becomes available to start another stream.

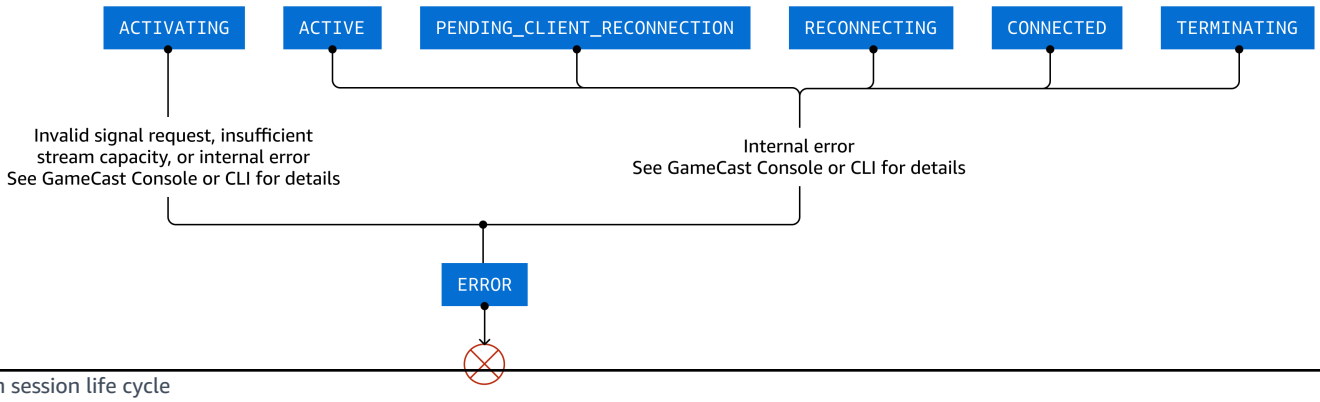
Stream session life cycle

When working with stream sessions in Amazon GameLift Streams, this diagram can help you understand the different states that a stream session transitions to throughout its life cycle.

- StartStreamSession creates a new stream session, which begins in `ACTIVATING` state. When Amazon GameLift Streams finds available resources to host the stream, the stream session transitions to `ACTIVE`. When a client connects to the active stream, the stream session transitions to `CONNECTED`.
- When a client disconnects from a stream, the stream session transitions to `PENDING_CLIENT_RECONNECTION` state. CreateStreamSessionConnection transitions the stream session to `RECONNECTING`, and will either initiate the client to reconnect to the stream or create a new stream session. When the client reconnects, it transitions back to `CONNECTED`. If a client is disconnected for longer than `ConnectionTimeoutSeconds`, the stream session ends.
- When a client doesn't connect to a stream session in `ACTIVE` or `PENDING_CLIENT_RECONNECTION` state within a period of time, then it transitions to `TERMINATED`.
- TerminateStreamSession initiates termination of the stream, and the stream session transitions to `TERMINATING` state. When the stream session terminates successfully, it transitions to `TERMINATED`.
- A stream session in any state, except `TERMINATED`, can transition to `ERROR`. When an API call returns `ERROR` as a Status value, check the value of StatusReason for a short description of the cause of the error. You can also call GetStreamSession to check these values.


Start
StartStreamSession
ACTIVATING
GameCast finds available resource to host stream
ACTIVE
Client connects to stream
CreateStreamSessionConnection
CONNECTED
RECONNECTING
Client disconnects from stream
CreateStreamSessionConnection
Time to connect elapsed
PENDING_CLIENT_RECONNECTION
TERMINATED


ACTIVE
PENDING_CLIENT_RECONNECTION
RECONNECTING
CONNECTED
TerminateStreamSession
TERMINATING
TERMINATED


ACTIVATING
ACTIVE
PENDING_CLIENT_RECONNECTION
RECONNECTING
CONNECTED
TERMINATING
Invalid signal request, insufficient stream capacity, or internal error
See GameCast Console or CLI for details
Internal error
See GameCast Console or CLI for details
ERROR

Reconnect back to your stream

If you refresh the website, switch browsers, or disconnect from your stream in some way, you can reconnect back to your stream within a grace period.

Each stream connection has a unique token that must be specified to reconnect back to that stream. In the Amazon GameLift Streams Web SDK sample web client, the stream's unique token is located in the URL address. For example: `http://localhost:8000/?token=2061cf1b-4bef-bf3e-e39165924480`.

For more information, refer to CreateStreamSessionConnection in the *Amazon GameLift Streams API Reference*.

Export stream session files

During a stream session, your application generates output files, which can help you debug or verify your application. The files can be logs, diagnostic information, crash dumps, save files, user data, screenshots, and so on. The files can be defined by the engine or frameworks that your application uses, or information that you've programmed your application to output.

Warning

Before you export files, be aware of the following things:

- Files may contain sensitive information written by your application, including credentials information.
- File sizes may be large depending on your application size, which impacts your Amazon S3 storage cost.
- If you select an Amazon S3 bucket in an AWS Region that differs from the Region of the stream group, then the exported stream session files will move across regions.

How it works

You must manually invoke this operation on an active stream session to export the files generated during that session. The stream session must be active, specifically in one of the following statuses `ACTIVE`, `CONNECTED`, `PENDING_CLIENT_RECONNECTION`, and `RECONNECTING`. At the end of the session, Amazon GameLift Streams exports the files to your bucket in Amazon Simple Storage

Service (Amazon S3). Thus, all exported data is within your ownership and is subject to the Amazon S3 bucket's permissions policy.

Here's a walkthrough of the stream session lifecycle with export files activated:

1. Amazon GameLift Streams begins a session by connecting the user to your application that's running on the compute resource.
2. While your application streams, it creates or modifies files in the filesystem of the runtime environment.
3. When the session ends, Amazon GameLift Streams gets a copy of all the new or modified files in the filesystem and exports the files to your Amazon S3 bucket.

Amazon GameLift Streams collects the following generated and modified files. Find them in the corresponding folders in the `.zip` archive.

- `application/`: The folder where your application or game is stored.
- `profile/`: The user's profile folder contains the user's personal settings, configurations, and data.
- `temp/`: The system's temp folder contains temporary files and data that your application and the system creates. This can include cache files, log files, or intermediate processing data.

To delete the files, delete the object in the Amazon S3 bucket.

Cost impact

You incur a cost for having the files stored in Amazon S3. A stream session might generate a large amount of data depending on your application. Be aware that with many stream sessions that have this feature enabled, the cost can add up.

For more information, refer to Amazon S3 pricing.

Export files (Console)

To enable export stream session files in the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Sessions** to view a list of active and past stream sessions within the last 90 days.

3. In the **Sessions** tab, select an active stream session.
4. Choose **Export files** to enable the export files feature for that stream session.
5. In the **Export stream sessions file** dialog box, choose either **Create a new S3 bucket** or **Select an existing S3 bucket**. Follow the steps in the console to create or select an S3 object to store the exported data into.

   > **Warning**
   >
   > If the ZIP file name matches an existing one in the directory, the previous one will be overwritten.

6. Choose **Confirm**. You can now find the session listed in the **Exported files** tab.
7. Wait for the session to end and for the files to export.

Amazon GameLift Streams will export the files when the session is in **Terminated** state. You can check the session status in the **Sessions** tab.

You can also check the exported files status in the **Session exports** tab. If the status is **Pending**, the stream session is still active, so Amazon GameLift Streams hasn't exported the files yet. If the status is **Succeeded**, you can download the files from Amazon S3. If the status is **Failed**, hover over the status to see the status reason.

Export files (CLI)

Prerequisite

You must configure the AWS CLI with your user credentials and your chosen AWS Region. For setup instructions, refer to Download the AWS CLI.

To export stream session files in the AWS CLI

In your AWS CLI use the ExportStreamSessionFiles command, customized for your content.

```
aws gameliftstreams export-stream-session-files \
    --identifier arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/
sg-1AB2C3De4 \
    --stream-session-identifier arn:aws:gameliftstreams:us-
west-2:111122223333:streamsession/sg-1AB2C3De4/ABC123def4567
    --ouput-uri s3://amzn-s3-demo-bucket/prefix
```

Where

`identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream group resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamgroup/sg-1AB2C3De4`

ID example: `sg-1AB2C3De4`

`stream-session-identifier`

An Amazon Resource Name (ARN) or ID that uniquely identifies the stream session resource.

ARN example: `arn:aws:gameliftstreams:us-west-2:111122223333:streamsession/sg-1AB2C3De4/ABC123def4567`

ID example: `ABC123def4567`

`output-uri`

The Amazon S3 bucket URI where Amazon GameLift Streams uploads the set of compressed exported files for this stream session.

There are two valid formats that you can provide. If the URI has a `.zip` or `.ZIP` file extension, then Amazon GameLift Streams stores the exported files at the provided URI. Otherwise, Amazon GameLift Streams generates the name for a compressed folder and stores it at the URI. The generated name follows the pattern: `date-time-applicationId-streamGroupId-streamSessionId`. For example:

- If you provide a URI called `s3://amzn-s3-demo-bucket/MyGame_Session1.zip`, then Amazon GameLift Streams saves the files in that exact ZIP folder.
- If you provide a URI called `s3://amzn-s3-demo-bucket/MyGame_Session1/`, then Amazon GameLift Streams will save the files at `s3://amzn-s3-demo-bucket/MyGame_Session1/YYYYMMDD-HHMMSS-applicationId-streamGroupId-sessionId.zip`.

Be sure that your ZIP file name complies with the Object key naming guidelines in the *Amazon Simple Storage Service User Guide*.

> **Warning**
>
> If the ZIP file name matches an existing one in the directory, the previous one will be overwritten.

You can check on the status of the active session by invoking the GetStreamSession API. From the stream session summary, you can get details about the exported files status. If the status is **Pending**, then the stream session is still active, so Amazon GameLift Streams hasn't exported the files yet. If the status is **Succeeded**, navigate to the output URI to see the files in Amazon S3. If the status is **Failed**, check the `StatusReason` in the `ExportFilesMetaData`.

Amazon GameLift Streams backend service and web client

Amazon GameLift Streams enables you to stream applications through a web browser. With the Amazon GameLift Streams Web SDK, you can set up a backend streaming service. Then, end users connect to a stream through a web client. They can play your game or interact with your application all through the cloud.

The Amazon GameLift Streams Web SDK includes a sample backend server and a sample web client, which you can use to get started on creating a backend service. You can also use these samples to test how Amazon GameLift Streams streams, without additional development. To get started, refer to Setting up a web server and client with Amazon GameLift Streams.

Topics

Supported browsers and input

The following lists the supported platforms and browsers for viewing Amazon GameLift Streams streams and their compatible input peripherals. Browsers must also be compatible with advanced video coding (AVC), or H.264.

Overall, we recommend Google Chrome, Microsoft Edge, or a custom Chromium-based desktop application for the best end-user experience and maximum compatibility, particularly with game controllers.

To learn more about which controllers are compatible with which browsers, see the Web Gamepad API. Although some guidance may not apply to Amazon GameLift Streams, we expect most game controllers to connect successfully via Bluetooth.

Operating system	Browser	Input
Windows	Chrome, Edge, Firefox	Keyboard, mouse, microphone, game controller
Mac	Chrome, Edge, Safari	Keyboard, mouse, microphone, game controller (in Bluetooth mode)
	Firefox	Keyboard, mouse, microphone
Linux	Chrome, Edge, Firefox	Keyboard, mouse
Android	Chrome, Edge	Simple touch-to-mouse emulation , microphone, external physical mouse, keyboard and game controller (in Bluetooth mode)
iOS	Chrome, Edge, Firefox, Safari	Simple touch-to-mouse emulation , microphone, external physical mouse, keyboard and game controller (in Bluetooth mode)

Known issues

Following are known issues with browsers and input:

- The PS5 and Luna game controllers are not supported in Firefox.

- Safari will immediately exit fullscreen whenever `Esc` is pressed. This cannot be overriden.
- “Embedded” or “in-app” browser views like those inside mobile apps such as LinkedIn, Yelp, Instagram, and others are not supported on iOS. These tend to disable the browser WebRTC support necessary for realtime interactive streaming. We recommend detecting non-standard browser strings and prompting the user to open in Safari.
- If the screen resolution in your application is not set to 1080p, mouse tracking might be impacted. We recommend disabling the selection of any other resolution, if possible. We also recommend disabling windowed mode, and only run in full screen.

Limitations

Most runtime environments support game controllers, except for Ubuntu 22.04 LTS. If you need game controller support, consider creating the game using another runtime environment. For a list of other runtime environments, refer to Runtime environments.

Required ports

To integrate Amazon GameLift Streams, ensure that your network infrastructure has the necessary ports open and accessible. The following is a list of the ports you should plan to have open on your network to communicate with Amazon GameLift Streams.

Port	Purpose
443 (HTTPS) TCP	AWS APIs, including Amazon GameLift Streams
33435-33465 UDP	Web RTC
40712	Data channel
8000	Amazon GameLift Streams Web SDK

Setting up a web server and client with Amazon GameLift Streams

In this tutorial, you will set up a web client application that integrates Amazon GameLift Streams' streaming service. Then, you will use the Amazon GameLift Streams Web SDK, a JavaScript library, and sample code that you can start with. The sample code includes a simple Amazon GameLift Streams backend web server and a simple web client. By the end of this tutorial, you can start a stream by using the sample code.

If it's your first time using Amazon GameLift Streams, we highly recommend starting with the Starting your first stream in Amazon GameLift Streams tutorial, which walks you through uploading a game to Amazon S3 and testing streaming it from within the Amazon GameLift Streams console in your browser.

Prerequisites

- An AWS account with proper credentials for programmatic access. For more information, see Setting up Amazon GameLift Streams.
- The AWS SDK.
- An Amazon GameLift Streams-supported web browser — see Supported browsers and input.
- Node.js 16 or newer — see Node.js downloads page.

Download the Web SDK

For this tutorial, you will need to download the following materials from the Resources section of the Getting Started product page:

- **Amazon GameLift Streams Web SDK bundle**: This includes sample code for a simple backend service and web client.
- **Amazon GameLift Streams Web SDK API Reference**: This API reference documents Amazon GameLift Streams API wrappers for JavaScript.

Set up your streaming resources

You must have stream resources—an application and a stream group—to start a stream. Specifically, you must have:

- An application in **Ready** status.
- A stream group in **Active** status with available stream capacity.

To set up an application and a stream group using either the Amazon GameLift Streams console or Amazon GameLift Streams CLI, refer to Prepare an application in Amazon GameLift Streams and Manage streaming with an Amazon GameLift Streams stream group, respectively. Alternatively, for an end-to-end walkthrough in the Amazon GameLift Streams console, refer to Starting your first stream in Amazon GameLift Streams.

Set up a backend server

The backend server is responsible for handling tasks such as authenticating users, configuring stream parameters, and performing Amazon GameLift Streams service API calls on behalf of end-users. Review the sample code and the Amazon GameLift Streams Web SDK API Reference to learn more about setting this up. Specifically, see the server.js file in the Amazon GameLift Streams Web SDK package.

> **Important**
>
> This code is example code for testing and evaluation purposes only and should not be used in a production capacity. For guidance on creating production client applications, including proper testing and evaluation procedures, refer to .

To run the sample backend service

1. Open a terminal or command prompt and navigate to the folder `AmazonGameLiftStreamsWebSDK\GameLiftStreamsSampleGamePublisherService\`.
2. Run the following commands:

```
npm install
node server.js
```

With the sample backend service running, end-users can connect to a stream through the web client. Test the web client in the next step.

Launch a web client

The web client application is responsible for receiving and decoding Amazon GameLift Streams streams, streaming to end-users, and providing the web browser UI for end-users to engage with the application. Review the sample code and the Amazon GameLift Streams Web SDK API Reference to learn more about how to integrate the JavaScript Amazon GameLift Streams Web SDK into your own web client application. Specifically, see `public/index.html` in the Amazon GameLift Streams Web SDK package. You can also look at the web page source when you launch a web client in your browser.

To launch a web client application

1. Open a web browser and navigate to `http://localhost:`*`port`*`/`. The port number is set by the backend server; by default, this is HTTP port 8000.
2. Play the game or use the software.
   a. To attach input, such as your mouse, choose **Attach input**.
   b. To exit the game, choose the **Esc** key.
   c. To stop the server process, choose **Ctrl+C** key.

Clean up streaming resources

> **Warning**
>
> A stream group incurs costs when it has allocated streaming capacity, even if that capacity is unused. To avoid unnecessary costs, scale your stream groups to your required size. We suggest during development that you scale always-on capacity in your stream groups to zero when not in use, or use on-demand capacity. For more information, refer to Scale stream groups to zero capacity.

After you complete the tutorial and no longer need to stream your application, follow these steps to clean up your Amazon GameLift Streams resources.

Deleting a stream group

When you delete a stream group, Amazon GameLift Streams works to release all stream capacity.

To delete a stream group using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. To view a list of your existing stream groups, in the navigation pane, choose **Stream groups**.
3. Choose the name of the stream group that you want to delete.
4. On the stream group detail page, choose **Delete**.
5. In the **Delete** dialog box, confirm the delete action.

Amazon GameLift Streams begins releasing compute resources and deleting the stream group. During this time, the stream group is in **Deleting** status. After Amazon GameLift Streams deletes the stream group, you can no longer retrieve it.

Deleting an application

You can only delete an application that meets the following conditions:

- The application is in the **Ready** or **Error** state.
- The application is not the default application of any stream groups. You must first delete the stream group by using the Amazon GameLift Streams console, or by using DeleteStreamGroup in the Amazon GameLift Streams API.
- The application is not linked to any stream groups. You must first unlink the stream group by using the Amazon GameLift Streams console, or by using DisassociateApplications in the Amazon GameLift Streams API.
- An application is not streaming in any ongoing stream session. You must wait until the client ends the stream session or call TerminateStreamSession in the Amazon GameLift Streams API to end the stream.

To delete an application using the Amazon GameLift Streams console

1. Sign in to the AWS Management Console and open the Amazon GameLift Streams console.
2. In the navigation bar, choose **Applications** to view a list of your existing applications. Choose the application you want to delete.
3. In the application detail page, choose **Delete**.
4. In the **Delete** dialog box, confirm the delete action.

Amazon GameLift Streams begins deleting the application. During this time, the application is in `Deleting` status. After Amazon GameLift Streams deletes the application, you can no longer retrieve it.

Customize stream appearance

Loading screen

When a customer opens a web browser to view a stream, the web client starts establishing a connection to the Amazon GameLift Streams stream session. While the stream session loads, you can display a custom background and logo to the customer's screen.

The Amazon GameLift Streams Web SDK sample client, in the `GameLiftStreamsSampleGamePublisherService/public/LoadingScreen/loadingscreen.js` file, demonstrates how you can implement an animated logo in your front-end web client. The default loading screen consists of 2 images: background and foreground. The foreground image is positioned in the middle and has a pulse animation. The animation plays only while the stream session is connecting.

To enable a loading screen

1. In the Amazon GameLift Streams Web SDK sample client, navigate to the `GameLiftStreamsSampleGamePublisherService/public/LoadingScreen/` folder.
2. Add your background and foreground images using the default names, `Background.png` and `LoadingLogo.png`. If you want to rename them or use a different image format, you must update the code in `GameLiftStreamsSampleGamePublisherService/public/loadingscreen.js`.
3. (Optional) In `GameLiftStreamsSampleGamePublisherService/public/loadingscreen.js`, update the JavaScript code to implement different animations.

Background image

You can display a custom background image in your stream. A background image appears when you're connected to the Amazon GameLift Streams server and your application either hasn't launched yet or has exited. If you don't specify a background image, then the stream displays a solid black (`#000000`) background by default.

The background image must have the following properties:

- The file must be named `Background.bmp` and placed in `s3://amzn-s3-demo-bucket/application-folder/GameLiftStreamsConfig/`.
- The file must be bmp format.
- For best fit, the resolution should match the stream resolution. Currently, the stream is fixed to 1080p, or 1920 x 1080 pixels.

If the file doesn't match this format or file path/name doesn't match, Amazon GameLift Streams will show solid black (`#000000`) background color.

To enable a background image in your stream

1. In the Amazon S3 bucket of your Amazon GameLift Streams application, navigate to the `GameLiftStreamsConfig` folder. Example: `s3://amzn-s3-demo-bucket/application-folder/GameLiftStreamsConfig/`.
2. Add an image named `Background.bmp`.

Locale preference

In Amazon GameLift Streams, you can set the locale preference per stream. This is useful if your application retrieves location-specific information from the end user's operating system, such as time or currency.

Amazon GameLift Streams supports the following languages:

Value	Description
en_US	U.S. English (default)
ja_jp.UTF-8	Japanese

To change the locale setting

When you call StartStreamSession using the Amazon GameLift Streams API, add `LANG=<language>` to your `AdditionalEnvironmentVariables`. Since locale preference is

unique per user, you set this at the stream-session level. If you don't set this, the stream uses U.S. English by default.

Example Example

```
aws gameliftstreams start-stream-session \
   --identifier arn:aws:gameliftstreams:us-west-2:123456789012:streamgroup/1AB2C3De4 \
   --protocol WebRTC \
   --signal-request "[webrtc-ice-offer json string]" \
   --user-id xnshijwh \
   --additional-environment-variables '{"LANG": "ja_JP.UTF-8"}'
```

Data channel communication between an application and web client

Data channels allow you to securely communicate arbitrary messages between your Amazon GameLift Streams application and the web client (the JavaScript code running in the end-user's web browser). This allows end-users to interact with the application that Amazon GameLift Streams is streaming, via the web browser where they're viewing the stream.

Here are some example use cases of data channels in Amazon GameLift Streams:

- Users can open URLs in the application in their local browser.
- Users can pass content in the clipboard back and forth to the application.
- Users can upload content from their local machine to the application.
- Developers can implement UI in the browser that sends commands to the application.
- Users can pass schemas to control display of visualization layers.

Features

Message size limits

Amazon GameLift Streams Web SDK imposes a maximum size limit of 64 KB (65536 bytes) per message. This ensures that the message size limits are compatible with most browsers, and that the communication has low impact on the total bandwidth of the stream.

Metrics

Metrics on your data channel usage are sent to your AWS account when a stream session ends. For more information, refer to Data channels in the *Monitoring Amazon GameLift Streams* section.

Using data channels

The Amazon GameLift Streams Web SDK provides the `sendApplicationMessage` function that sends a message as a byte array to the application. The message is processed by a callback function, `clientConnection.applicationMessage` that you define.

If the client sends messages before the application connects to the data channel port, the messages are queued. Then, when the application connects, it receives the messages. However, if the application sends messages before the client connects to the data channel port, the messages are lost. The application must check the connection state of the clients before sending a message.

On the client-side

Write the following code in your web client application.

1. Define the callback function to receive incoming messages from the application.

```
function streamApplicationMessageCallback(message) {
    console.log('Received ' + message.length + ' bytes of message from
    Application');
}
```

2. Set `clientConnection.applicationMessage` to your callback function.

```
clientConnection: {
    connectionState: streamConnectionStateCallback,
    channelError: streamChannelErrorCallback,
    serverDisconnect: streamServerDisconnectCallback,
    applicationMessage: streamApplicationMessageCallback,
}
```

3. Call the `GameLiftStreams.sendApplicationMessage` function to send messages to your application. You can call this any time, as long as the stream session is active and the input is attached.

As an example, refer to the Amazon GameLift Streams Web SDK sample client, which demonstrates how to set up a simple data channel on the client-side.

On the application-side

Write the following logic in your application.

Step 1. Connect to the data channel port

When your application starts, connect to port `40712` on `localhost`. Your application should maintain this connection for the entire duration of execution. If the application closes the connection, it cannot be reopened.

Step 2. Listen for events

An event starts with a fixed-size header, followed by variable-length associated data. When your application receives an event, parse the event to retrieve the information.

Event format

- **Header**: A 4-byte header in the form `abcc`
  - `a` : Client id byte. This identifies a specific client connection, in the case of multiple connections (due to disconnection and reconnection).
  - `b` : Event type byte. `0` - the client connected, `1` - the client disconnected, `2` - a message is sent from the client. Other event types may be received with future Amazon GameLift Streams service updates, and should be ignored.
  - `cc` : Length of the associated event data. This is represented as 2 bytes with big-endian ordering (first byte is the most significant). If the event type is 2, the event data represents the message contents from the client.
- **Data**: The remaining bytes contain the event data, such as a client message. The length of the data is indicated by `cc` in the header.

To listen for events

1. Read the four header bytes to retrieve the client id, event type, and length of the event data.
2. Read the variable-length event data, regardless of the client id and event type, according to the length described in the header. It's important to read the data unconditionally so that event data is never left in the buffer, where it could be confused with the next event header. Do not make assumptions about the length of the data based on event types.

3. Take appropriate action based on the event type, if recognized by your application. This action might include logging an incoming connection or disconnection, or parsing the client message and triggering application logic.

Step 3. Transmit messages to the client

The application should transmit messages with the same four-byte header format used by incoming events.

To transmit a message to the client

1. Write the header with the following properties:

    a. a : Client id byte. If your message is in response to a client message, it should reuse the same client id as the incoming client message, to avoid race conditions such as delivering a response from an old client connection to a newly reconnected client. If your application is sending an unsolicited message to the client, it should set the client id to match the most recent "client connection" event (event type 0).

    b. b : The event type of outgoing messages must always be 2. The client ignores messages with other event types.

    c. cc : Length of the message, in bytes.

2. Write the message bytes.

The message delivers to the specified client, unless the client disconnects. When a disconnected client reconnects, a new client ID is assigned via a client connected event. Any undelivered messages for the old client ID are discarded.

Example

The following pseudo-code demonstrates the logic to communicate messages in the application-side. For a complete example using Winsock, refer to Complete Winsock Client Code in the Windows Sockets 2 documentation.

```
connection = connect_to_tcp_socket("localhost:40712")
loop:
    while has_pending_bytes(connection):
        client_id = read_unsigned_byte(connection)
        event_type = read_unsigned_byte(connection)
        event_length = 256 * read_unsigned_byte(connection)
        event_length = event_length + read_unsigned_byte(connection)
        event_data = read_raw_bytes(connection, event_length)
        if message_type == 0:
            app_process_client_connected(client_id)
        else if message_type == 1:
            app_process_client_disconnected(client_id)
        else if message_type == 2:
            app_process_client_message(client_id, event_data)
        else:
            log("ignoring unrecognized event type")
    while app_has_outgoing_messages():
        target_client_id, message_bytes = app_next_outgoing_message()
        message_length = length(message_bytes)
        write_unsigned_byte(connection, target_client_id)
        write_unsigned_byte(connection, 2)
        write_unsigned_byte(connection, message_length / 256)
        write_unsigned_byte(connection, message_length mod 256)
        write_raw_bytes(connection, message_bytes)
```

Amazon GameLift Streams launch checklist

Preparing for a successful launch on Amazon GameLift Streams involves careful planning and coordination. Follow this detailed checklist to ensure a smooth experience in the weeks leading up to your event.

Notify the Amazon GameLift Streams team

Action: At least 6-8 weeks in advance, inform the Amazon GameLift Streams team and your technical account manager about your launch timeline and expected peak concurrent streams.

Reason: Understanding the scale helps us ensure that your service limits are adequate and adjust them if necessary. We also provide guidance on capacity availability and recommendations for the launch.

Compatibility and performance testing

Action: Test your application across multiple Amazon GameLift Streams stream classes and runtimes to confirm it streams well. Amazon GameLift Streams offers NVIDIA based stream classes with different levels of performance and runtimes supported.

Reason: Thorough testing helps identify and resolve any potential compatibility issues before the launch. Keep in mind the following about stream classes:

- The "high" stream classes support multi-tenancy, allowing two applications to run concurrently on a single instance. If you're using the "high" stream class, test with at least 2 concurrent streams to see how your application performs with shared resources, such as the CPU, GPU, and memory.

Capacity reservation

Action: At least 6-8 weeks before launch, reach out to your technical account manager to reserve capacity, especially if you anticipate a critical, large-scale need. Decide on the stream class(es) and the AWS region(s) based on your compatibility testing, performance requirements, and budget. Provide the start/end times and the required capacity.

Reason: Amazon GameLift Streams operates on a first-come, first-serve basis using on-demand capacity. Reservations are essential to guarantee the necessary capacity.

Performance testing at scale

Action: Conduct thorough load testing of your APIs and your Amazon GameLift Streams configurations to observe the performance (latency, resolution, and frame rate).

Reason: Load testing reveals how your application and Amazon GameLift Streams configurations will perform under stress before the launch. This is crucial to ensure smooth performance at scale.

Pre-launch setup

Action: At least 2-3 days before launch, create your final application resources and stream groups. Validate streaming performance and scale up capacity as needed.

Reason: This ensures that all components are working as expected, minimizing the risk of unexpected issues and allowing for easier diagnosis and recovery during the event.

Additional tips

- **Consistency is key**: Using the same existing stream groups throughout a launch event maintains consistency in the Amazon GameLift Streams backend, simplifying troubleshooting.
- **Monitor closely**: Closely monitor performance and user feedback to quickly address any issues.

Need Further Assistance?

If you have any questions or require further support, don't hesitate to reach out to us at Amazon GameLift Streams support. We're here to help ensure your launch is successful and seamless.

Security in Amazon GameLift Streams

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon GameLift Streams, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. Amazon GameLift Streams is designed to run programs that you provide, and that you are solely responsible for the content and security of those programs. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon GameLift Streams. The following topics show you how to configure Amazon GameLift Streams to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon GameLift Streams resources.

Topics

Data protection in Amazon GameLift Streams

The AWS shared responsibility model applies to data protection in Amazon GameLift Streams. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon GameLift Streams or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Amazon GameLift Streams handles service-specific data as follows:

- **Customer-supplied applications** – Amazon GameLift Streams stores customer data, if provided, in internal service-managed Amazon S3 buckets and on NVME storage drives attached to Amazon EC2 instances. All data is stored with service-managed encryption at rest. There is no direct customer access to this copy of the data. To delete an application, use the Amazon GameLift Streams console or the service API.
- **Customer-supplied metadata** – Customers may provide metadata to Amazon GameLift Streams APIs including descriptions, connection information, and opaque identifiers such as customer IDs. This metadata is always associated with specific customer resources.
- **Customer-generated data** – If an application writes new data as part of its normal operation, this customer-generated data is retained until the end of the user session. At the end of the session, generated data can optionally be exported to an Amazon S3 bucket destination of the customer's choice. Customer-generated data otherwise does not leave the Amazon EC2 instance where it was generated. For more information about data handling, refer to the topics on Session isolation.
- **Metrics and event data** – Amazon GameLift Streams metric and event data, which can be accessed through the Amazon GameLift Streams console or by calls to the service API. Data is available on applications, stream groups, and stream sessions. Authorized users can also access this data through Amazon CloudWatch and CloudWatch Events.

> **Important**
>
> If you provide customer IDs or other identifiers to Amazon GameLift Streams, it is expected that these values are anonymized references and do not contain any sensitive or personal information. Amazon GameLift Streams does not redact any metadata fields.

For more information about data protection, see the AWS shared responsibility model and GDPR blog post on the *AWS Security Blog*.

Encryption at rest

At-rest encryption of Amazon GameLift Streams-specific data is handled as follows:

- Application content is stored in service-managed encrypted Amazon S3 buckets and additionally on hardware-encrypted NVME drives attached to service-managed Amazon EC2 instances.

Encryption in transit

Calls to the Amazon GameLift Streams APIs are made over a secure (SSL) connection and authenticated using AWS Signature Version 4 (when connecting through the AWS CLI or AWS SDK, signing is handled automatically). Calling entities use security credentials, which are authenticated by applying the IAM access policies that are defined for Amazon GameLift Streams resources.

Direct communication between stream clients and stream servers hosted by Amazon GameLift Streams is as follows:

- Stream clients connect directly to Amazon GameLift Streams-hosted stream sessions. Encryption of this direct communication is the responsibility of the customer.
- In the context of multi-location stream groups, in order to stream an application from any location in the stream group that has been allocated streaming capacity, Amazon GameLift Streams securely replicates applications to those locations.

  Similarly, Amazon GameLift Streams will save log data and session files, when requested, to customer-named Amazon S3 buckets at the end of a session. If the bucket is not in the same location as the session, Amazon GameLift Streams will transfer the files securely to the AWS Region where the bucket is located.

Session isolation in Linux stream classes

On Linux stream classes (Ubuntu and Proton runtimes), Amazon GameLift Streams uses *container isolation*. Every session runs in a new Linux container which is discarded after use. This means each new session runs in a fresh environment, isolated from other users sharing the compute resource (if running in a shared-resource stream class). No data from prior sessions exists when a new session starts up.

Session isolation in Windows stream classes

On Windows stream classes (Microsoft Windows Server runtimes), Amazon GameLift Streams uses *software isolation*. The service relies on a software agent to reset critical system state between sessions. Some folders are preserved across multiple sessions to allow for performance optimizations, such as on-host disk caching. The software agent automatically removes any files that were generated in the user's profile directory during the prior stream session. However, the agent does not remove any files that existed prior to the application running and were

modified while the application was running. Nor does it remove any Windows registry keys that the application had added. Customers should be aware that it is their responsibility to avoid damaging the integrity of the overall operating system. Applications are executed as the Administrator user, which may permit modification to critical system-level files, including changes that persist across multiple sessions. It is the responsibility of the customer to secure their applications and guard against creating unsafe or unstable operating system modifications.

Customers are responsible for cleaning up those modified files and added registry keys from previous sessions when the application launches. This is an important step to protect confidential or sensitive information that the application writes to the user's profile directory. To do this, customers can write their own custom script that performs the following actions:

- Restore any files outside of the `%USERPROFILE%` directory that were modified by the application.
- Clean up any sensitive or user-specific registry keys that the application added.

Key management

The service uses AWS-managed keys. Each region uses a separate KMS key. Customer-managed keys are not supported.

Application files provided to Amazon GameLift Streams cannot be republished or exported from the service. The customer can use the service console or APIs to delete applications. Drives which previously held these application files can be completely purged by deleting the associated stream groups.

Inter-network traffic privacy

Amazon GameLift Streams uses public-facing networks to host stream sessions. Each stream group consists of one or more service-managed VPC networks which are isolated from other stream groups and from other customers. Inbound network connections are denied except for authenticated, service-brokered WebRTC stream connections. Customer applications may connect out from these VPCs to other public addresses without restriction.

Additionally, there is no way for a customer to make a stream or their application data publicly-accessible using service API calls or settings alone. All service interactions are gated by AWS-authenticated API calls. If the customer wishes to make a stream accessible to the public they must create their own client web application which makes the authenticated calls to start and display a stream.

Identity and Access Management for Amazon GameLift Streams

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon GameLift Streams resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon GameLift Streams.

Service user – If you use the Amazon GameLift Streams service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon GameLift Streams features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon GameLift Streams, see Troubleshooting Amazon GameLift Streams identity and access.

Service administrator – If you're in charge of Amazon GameLift Streams resources at your company, you probably have full access to Amazon GameLift Streams. It's your job to determine which Amazon GameLift Streams features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon GameLift Streams, see How Amazon GameLift Streams works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon GameLift Streams. To view example Amazon

GameLift Streams identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon GameLift Streams.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a

role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can

perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user

or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon GameLift Streams works with IAM

Before you use IAM to manage access to Amazon GameLift Streams, learn what IAM features are available to use with Amazon GameLift Streams.

IAM features you can use with Amazon GameLift Streams

IAM feature	Amazon GameLift Streams support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial. ABAC is only supported for applications and stream groups.
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	No

To get a high-level view of how Amazon GameLift Streams and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon GameLift Streams

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an

identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon GameLift Streams

To view examples of Amazon GameLift Streams identity-based policies, see Identity-based policy examples for Amazon GameLift Streams.

Resource-based policies within Amazon GameLift Streams

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Amazon GameLift Streams

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation.

There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon GameLift Streams use the following prefix before the action:

```
gameliftstreams
```

To specify multiple actions in a single statement, separate them with commas.

Example

```
"Action": [
            "gameliftstreams:action1",
            "gameliftstreams:action2"
               ]
```

To view examples of Amazon GameLift Streams identity-based policies, see Identity-based policy examples for Amazon GameLift Streams.

Policy resources for Amazon GameLift Streams

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To view examples of Amazon GameLift Streams identity-based policies, see Identity-based policy examples for Amazon GameLift Streams.

Policy condition keys for Amazon GameLift Streams

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To view examples of Amazon GameLift Streams identity-based policies, see Identity-based policy examples for Amazon GameLift Streams.

ACLs in Amazon GameLift Streams

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Amazon GameLift Streams

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Amazon GameLift Streams

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Amazon GameLift Streams

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

When creating new application resources, Amazon GameLift Streams uses the permissions of the calling principal to access the Amazon S3 bucket that contains the customer's application files. Amazon GameLift Streams also examines the calling principal to verify opt-in eligibility for certain cross-region functionality, such as multi-location stream groups.

Service roles for Amazon GameLift Streams

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon GameLift Streams functionality. Edit service roles only when Amazon GameLift Streams provides guidance to do so.

Service-linked roles for Amazon GameLift Streams

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon GameLift Streams

By default, users and roles don't have permission to create or modify Amazon GameLift Streams resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon GameLift Streams, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon GameLift Streams in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon GameLift Streams resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies*

that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Amazon GameLift Streams console

To access the Amazon GameLift Streams console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon GameLift Streams resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "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": "*"
        }
    ]
}
```

Troubleshooting Amazon GameLift Streams identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon GameLift Streams and IAM.

Topics

I am not authorized to perform an action in Amazon GameLift Streams

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `gameliftstreams:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 gameliftstreams:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `gameliftstreams:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon GameLift Streams resources

This is not possible with Amazon GameLift Streams. All API access is restricted to the account which owns the resources. Instead, customers who wish to share content externally are responsible for using their account to initiate new stream sessions on behalf of other users using Amazon GameLift Streams APIs, and forwarding the appropriate connection information to those external users' web browsers.

Compliance validation for Amazon GameLift Streams

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon GameLift Streams

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the data redundancy provided by the AWS global infrastructure, Amazon GameLift Streams is built with a resilient multi-Availability Zone infrastructure. In the case of an Availability Zone outage, individual existing sessions might be affected, but the service will continue to load-balance new sessions across healthy Availability Zones.

Infrastructure Security in Amazon GameLift Streams

As a managed service, Amazon GameLift Streams is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon GameLift Streams through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Shared tenancy in Amazon GameLift Streams

Some Amazon GameLift Streams stream groups rely on internal resource sharing. This approach is different from dedicated resource allocation, as seen in other AWS services like Amazon EC2. Amazon GameLift Streams provisions stream groups uniquely within your AWS account, and resources are never shared or mixed between different stream groups. A single stream group might share underlying compute and storage resources across multiple streams. You define the degree of sharing by your stream class selection. The Windows-based classes and the Linux-based "ultra" classes never share resources, while the Linux-based "high" classes use shared resources to provide two streams from a single GPU.

This shared tenancy model in the Linux-based "high" stream classes is unique to Amazon GameLift Streams and comes with important security and performance implications. You are responsible for maintaining the security of your provided applications. The security posture of a dedicated-resource group is equivalent to running one application per physical server, which is the highest possible level of isolation. The security posture of a shared-tenancy group is equivalent to hosting multiple application containers on a single physical server. This posture isn't inherently insecure, but it might amplify the impact of existing security vulnerabilities in your applications.

Amazon GameLift Streams makes efforts to ensure that each application instance is self-contained and isolated regardless of resource sharing. However, if an application consumes CPU or GPU resources beyond the limits of the provisioned stream class, it can have an impact on other streams that use the shared resources (in a "high" stream group, up to one other stream). Your application should regulate its own resource consumption. If your application can't regulate its resource usage and your use case has zero tolerance for a potential "noisy neighbor" problem, a dedicated-resource stream class, such as `gen4n_win2022`, `gen5n_win2022`, `gen4n_ultra` , or `gen5n_ultra` is recommended.

Interface VPC endpoints in Amazon GameLift Streams

You can improve the security posture of your VPC by configuring Amazon GameLift Streams to use an interface VPC endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that allows you to privately access Amazon GameLift Streams APIs by using private IP addresses. AWS PrivateLink restricts all network traffic between your VPC and Amazon GameLift Streams to the Amazon network. You don't need an internet gateway, a NAT device, or a virtual private gateway.

For more information about AWS PrivateLink and VPC endpoints, see VPC endpoints in the *Amazon VPC User Guide*.

> **Note**
>
> AWS PrivateLink is only applicable to API endpoints. Amazon GameLift Streams managed stream sessions always use public network addresses.

Creating the VPC endpoints for Amazon GameLift Streams

To create the VPC endpoint for the Amazon GameLift Streams service, use the Access an AWS service using an interface VPC endpoint procedure in the *Amazon VPC User Guide* to create the following endpoint:

- `com.amazonaws.region.gameliftstreams`

> **Note**
>
> *region* represents the Region identifier for an AWS Region supported by Amazon GameLift Streams, such as `us-east-2` for the US East (Ohio) Region.

Creating a VPC endpoint policy for Amazon GameLift Streams

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon GameLift Streams. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Control access to VPC endpoints using endpoint policies in the *Amazon VPC User Guide*.

Example Example: VPC endpoint policy for Amazon GameLift Streams

The following is an example of an endpoint policy for Amazon GameLift Streams. When attached to an endpoint, this policy grants permission to create and list stream groups.

```
{
```

```
  "Statement":[
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "gameliftstreams:CreateStreamGroup",
        "gameliftstreams:ListStreamGroups"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configuration and vulnerability analysis in Amazon GameLift Streams

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model. AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resource: Amazon Web Services: Overview of security processes (whitepaper).

The following security best practices also address configuration and vulnerability analysis in Amazon GameLift Streams:

- Customers are responsible for the management of software deployed to Amazon GameLift Streams stream groups for stream hosting. Specifically:
  - Customer-provided application content and software should be maintained, including updates and security patches. To update, create a new Amazon GameLift Streams application and deploy it to new stream groups.
  - At this time, the operating system and runtime environment for a stream group is updated only when you create a new stream group. To patch, update, and secure the operating system and other applications that are part of the runtime environment, we recommend that you recycle stream groups every two to four weeks, regardless of application updates.

- Customers should consider regularly updating their games with the latest SDK versions, including the AWS SDK and the Amazon GameLift Streams Web Client SDK.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Amazon GameLift Streams gives another service to the resource. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, `arn:aws:gameliftstreams:*:123456789012:*`.

If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Amazon GameLift Streams to prevent the confused deputy problem.

Example

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Sid": "ConfusedDeputyPreventionExamplePolicy",
        "Effect": "Allow",
        "Principal": {
          "Service": "gameliftstreams.amazonaws.com"
        },
        "Action": "gameliftstreams:ActionName",
        "Resource": [
          "arn:aws:gameliftstreams:::ResourceName/*"
        ],
        "Condition": {
          "ArnLike": {
            "aws:SourceArn": "arn:aws:gameliftstreams:*:123456789012:*"
          },
          "StringEquals": {
            "aws:SourceAccount": "123456789012"
          }
        }
      }
    }
```

Security best practices for Amazon GameLift Streams

Amazon GameLift Streams provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

- At this time, the operating system and runtime environment for a stream group is updated only when you create a new stream group. To patch, update, and secure the operating system and other applications that are part of the runtime environment, we recommend that you recycle stream groups every two to four weeks, regardless of application updates.
- Best practices for security, identity, and compliance

Monitoring Amazon GameLift Streams

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon GameLift Streams and your other AWS solutions. AWS provides the following monitoring tools to watch Amazon GameLift Streams, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For more information, see the Amazon CloudWatch User Guide.
- With *Amazon CloudWatch Logs* you can monitor, store, and access your log files from services like Amazon Elastic Compute Cloud, AWS CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when your services meet certain thresholds. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon Simple Storage Service bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Monitor Amazon GameLift Streams with Amazon CloudWatch

You can monitor Amazon GameLift Streams using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

Amazon GameLift Streams provides metrics to help customers monitor the following:

- Stream group capacity and usage.
- Stream performance and resource usage.
- Stream status to resolve issues and support users.
- Customer engagement across content offerings.

- Data channel usage.

The following tables list the dimensions and metrics for Amazon GameLift Streams.

Stream group capacity and usage

Use these metrics to help scale resources to meet demand. These metrics are published every minute.

> **Important**
>
> Due to an issue with CloudWatch's data retention policy, accurate capacity metrics are only available for the last 15 days. For capacity metrics that are older than 15 days, no data will be visible when the period is 1 minute, and the data shown will be inaccurate when the period is 5 minutes or more.
> In the meantime, you can add `SUM(METRICS())/5` math (for example, when using a 5-minute period) to a sum-type statistic in your CloudWatch graph as a workaround to see accurate capacity counts beyond the 15 day, 1-minute metrics retention limitation.

Metric	Description	Dimension	Unit
ActiveCapacity	The number of compute resources that are provisioned and ready to stream. It includes resources that are currently streaming and resources that are idle and ready to respond to new stream requests.	(StreamGroupId, Location)	Count
IdleCapacity	The numerical portion of active capacity that is not currently streaming. It represents the availability of compute resources to respond to new stream requests.	(StreamGroupId, Location)	Count

Stream group performance and resource utilization

These metrics are published every minute.

Metric	Description	Dimension	Unit
MemoryUtilization	% of available memory used by the stream.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Percentage
CPUUtilization	% of available CPU used by the stream.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Percentage
FrameCaptureRate	Rate at which frames are captured from the application.	(StreamGroupId, Location), (ApplicationId, StreamClass)	None
AudioCaptureRate	Rate at which audio samples are captured from the application.	(StreamGroupId, Location), (ApplicationId, StreamClass)	None
RoundTripTime	Round trip time between client and server.	(StreamGroupId, Location), (Applicat	ms

Metric	Description	Dimension	Unit
		ionId, StreamClass)	

Stream status

These metrics are published at the end of a stream session.

Metric	Description	Dimension	Unit
TerminatedStreamSessions	Number of sessions ended in state `TERMINATED`	(StreamGroupId, Location), (ApplicationId, StreamClass)	Count
ErroredStreamSessions	Number of sessions ended in state `ERROR`	(StreamGroupId, Location), (ApplicationId, StreamClass)	Count

Customer engagement

These metrics are published at the end of a stream session..

Metric	Description	Dimension	Unit
Session Length	Stream session duration	(StreamGroupId, Location), (Applicat	Seconds

Metric	Description	Dimension	Unit
		ionId, StreamClass)	

Data channels

These metrics are published at the end of a stream session.

Metric	Description	Dimension	Unit
DataChannel-ApplicationConnected	Number of times your application connects to the data channel port. This number is at most 1 per stream session.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Count
DataChannel-ApplicationMessage	Number of messages your application has sent to your client.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Count
DataChannel-ApplicationMessageBytes	Total bytes of messages your application has sent to your client.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Bytes
DataChannel-ClientMessage	Number of messages your client has sent to your application.	(StreamGroupId, Location), (Applicat	Count

Metric	Description	Dimension	Unit
		ionId, StreamClass)	
DataChannel-Client MessageBytes	Total bytes of messages your client has sent to your application.	(StreamGroupId, Location), (ApplicationId, StreamClass)	Bytes

Log Amazon GameLift Streams API calls using AWS CloudTrail

Amazon GameLift Streams is integrated with AWS CloudTrail. CloudTrail provides a record of actions that a user, role, or AWS service takes in Amazon GameLift Streams. CloudTrail captures all API calls for Amazon GameLift Streams as events. The calls captured include calls from the Amazon GameLift Streams console and code calls to the Amazon GameLift Streams API operations. If you create a *trail*, you can turn on continuous delivery of CloudTrail events to an Amazon Simple Storage Service (Amazon S3) bucket. This includes events for Amazon GameLift Streams. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. You can use the information collected by CloudTrail to find information on each request to Amazon GameLift Streams. This information includes the IP address where the request originated, who made the request, and when it was made.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon GameLift Streams information in CloudTrail

When you create your AWS account, CloudTrail is automatically turned on for the account. When activity occurs with any AWS service events, including Amazon GameLift Streams events, that activity is recorded in a CloudTrail event in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon GameLift Streams, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs

events from all Regions in the AWS partition and delivers the log files to the S3 bucket that you specify. You can also configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions
- Receiving CloudTrail log files from multiple accounts

CloudTrail logs all Amazon GameLift Streams API actions. These are documented in the Amazon GameLift Streams API Reference. For example, calls to the StartStreamSession and ListStreamGroups actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request used root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request used temporary security credentials for a role or federated user.
- Whether another AWS service made the request.

For more information, see the CloudTrail userIdentity element.

Understanding Amazon GameLift Streams log file entries

A trail is a configuration that enables delivery of events as log files to an S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the operation CreateApplication.

Example : CloudTrail log entry

```
{
      "eventVersion": "1.08",
```

```
      "userIdentity": {
         "type": "AssumedRole",
         "principalId": "AIDACKCEVSQ6C2EXAMPLE:assume-temporary-gameliftstreams-access-
role",
         "arn": "arn:aws:sts::111122223333:assumed-role/GameLiftStreamsTestRole/assume-
temporary-gameliftstreams-access-role",
         "accountId": "111122223333",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "sessionContext": {
               "sessionIssuer": {
                  "type": "Role",
                  "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                  "arn": "arn:aws:iam::111122223333:role/GameLiftStreamsTestRole",
                  "accountId": "111122223333",
                  "userName": "GameLiftStreamsTestRole"
               },
               "webIdFederationData": {},
               "attributes": {
                  "creationDate": "2022-11-01T18:05:32Z",
                  "mfaAuthenticated": "false"
               }
         }
      },
      "eventTime": "2022-11-01T18:05:34Z",
      "eventSource": "gameliftstreams.amazonaws.com",
      "eventName": "CreateApplication",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "aws-sdk-javascript/2.0.0 Linux/4.14.291-218.527.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/11.0.17+9-LTS Java/11.0.17 vendor/Amazon.com_Inc. exec-env/
AWS_ECS_FARGATE io/sync http/Apache cfg/retry-mode/legacy",
      "requestParameters": {
         "ApplicationSourceUri": "s3://amzn-s3-demo-bucket/MyGame",
         "Description": "Canary-CustomerS3Location-1667325931",
         "RuntimeEnvironment": {
               "Type": "WINE-STAGING",
               "Version": "20220721"
         },
         "ClientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
         "ExecutablePath": "MyGame100.exe"
      },
      "responseElements": {
         "Status": "PROCESSING",
         "ApplicationSourceUri": "s3://amzn-s3-demo-bucket/MyGame",
```

```
        "Description": "MyGame-High",
        "RuntimeEnvironment": {
              "Type": "WINE-STAGING",
              "Version": "20220721"
        },
        "LastUpdatedAt": 1667325933.972,
        "CreatedAt": 1667325933.972,
        "Id": "EXAmplE11",
        "Arn": "arn:aws:gameliftstreams:us-west-2:111122223333:EXAmplE11",
        "ExecutablePath": "MyGame100.exe"
      },
      "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
      "readOnly": false,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "recipientAccountId": "111122223333",
      "eventCategory": "Management"
        }
```

Troubleshooting Amazon GameLift Streams

Topics

Connectivity issues

When you set up your Amazon GameLift Streams backend service, check the following:

- Choose the closest AWS Region possible to the end users. High latency from your clients to the Region hosting your stream can impact stream quality. You can ping AWS console endpoints in the Region to get an approximate latency measurement.
- Verify your stream group has capacity for new streams.
- Verify that `connectionTimeout` is reasonably set. Increase it to allow more time for customers to connect before timing out.

Advise your customers to check the following:

- Open the UDP ports 33435-33465 to allow streaming from Amazon GameLift Streams. If Amazon GameLift Streams can't reach these ports, it can lead to streaming issues, such as a black or grey screen.
- Verify that your internet connection can sustain a connection speed of at least 10 Mbps for a 1080p stream. If you detect network issues while playing on Amazon GameLift Streams, your internet speed might be fluctuating and you might not be getting at least 10 Mbps consistently. Run an internet speed test and continue through the troubleshooting steps.
- Use a wired network if possible. When using Wi-Fi, move your device close to your router for stronger signal strength.

- If you're using a Wi-Fi router with both 2.4 GHz and 5 GHz bands, try connecting to a different band. If you're unsure how to switch your router to a different band, visit the support pages of the manufacturer or provider of your Wi-Fi router. You can also contact their customer service.
- Identify if others on the same network (especially when on home Wi-Fi) are running high-bandwidth applications like video streaming, downloading, online gaming, or backups.
- Close other applications on your device that take up bandwidth.
- Don't use a VPN or proxy while streaming. They can cause higher latencies and block the gameplay.
- Verify you're using Wi-Fi instead of cellular networks when playing on an iPad or iPhone. Using a cellular network can result in connectivity issues.
- MacOS users should disable Location Services as it will cause the Wi-Fi to pause from time to time, which will lead to a poor streaming experience.

Performance issues

If your game is experiencing high latency or poor optimization, it can result in a disappointing gaming experience. Make sure to check that your game streams smoothly with Amazon GameLift Streams.

Check the connection

If your game runs well on your own machine but struggles when you stream it on Amazon GameLift Streams, consider the following:

- Your machine might have more powerful hardware than Amazon GameLift Streams. Make sure to test the application on a machine with similar performance as the hardware that Amazon GameLift Streams uses, which is similar to a laptop with NVIDIA 2070 GPU.
- The problem might be due to your network connection or Amazon GameLift Streams's settings. Try the troubleshooting tips in the Connectivity issues section.

Optimize your game

If your game is slow even when running locally, you'll need to optimize its performance. The best optimization methods will depend on the specific engine or framework you're using.

- For Unreal Engine games, refer to Profiling Unreal Engine performance.

Application issues

Preliminary checks

- Run your application on a different machine to verify that it's correctly packaged. This confirms that your application content doesn't contain any hardcoded paths, missing assets, libraries, or binaries that might not work on other devices.
- (Optional) Run your application on a machine with a GPU that's comparable to your Amazon GameLift Streams stream class. This verifies that your application's rendering settings are compatible with the GPU and that the performance meets your expectations.
- Open the UDP ports 33435-33465 to allow streaming from Amazon GameLift Streams. If Amazon GameLift Streams can't reach these ports, it can lead to streaming issues, such as a black or grey screen.

Application doesn't work with Amazon GameLift Streams on Proton

- **Verify that your application is compatible with Proton.** Test your application on a local environment without the Amazon GameLift Streams server to verify that it's compatible with Proton. For instructions, see Troubleshoot compatibility with Proton for Amazon GameLift Streams.

Application issues due to screen resolution

Applications may freeze, crash, or render incorrectly if you attempt to use full-screen resolutions that is not 1920x1080. We recommend that you use a "borderless full-screen" window to run your application and do not attempt to change the resolution.

Unreal Engine application crashes or requires additional dependencies

If your Unreal Engine application crashes, stalls, or requires you to install additional dependencies, such as Microsoft Visual C++ Runtime, try the following.

- **Use the correct executable.** For your application to work correctly with Amazon GameLift Streams, set the application path to the full executable that's located in the `Binaries/Win64/` subfolder, or similar. Unreal Engine produces two executables: a small executable (a shortcut) at the root of the folder, and a full executable in the `Binaries/Win64/` subfolder. If the full

executable is missing, the application might have not been built correctly. For example, see the following folder structure for a sample Unreal application:

```
BuildApp
|-> MyUnrealApp.exe
|-> MyUnrealApp
            |-> Binaries
                    |-> Win64
                            |-> MyUnrealApp.exe
```

- **Turn off Unreal Engine Asserts.** Disable Check, Verify, and Ensure macros. This can prevent the application from creating crash dumps, which otherwise causes the application to stall on Amazon GameLift Streams. If Asserts are enabled, you should expect a delay. For more information, see Asserts in Unreal Engine documentation.
  - Set `USE_CHECKS_IN_SHIPPING=0` to disable Check and Verify macros.
  - Set `handleensurepercent=0` to disable Ensure macros.

Windows application terminates at launch

If your Windows application terminates at launch, your application may be missing required DLLs. If your application is a debug build, then it specifically requires the debug version of the Visual C++ library DLLs.

To resolve this, we recommend that you package your build and DLLs side-by-side. For instructions, refer to Prepare a test machine to run a debug executable by Microsoft.

With the packaged build and DLLs, test your application on a clean machine, such as an Amazon EC2 instance. When you're ready to try it on Amazon GameLift Streams, create a new application using this package. Be sure to choose the correct executable that will run the build with the included DLLs.

In general, we recommend that you test your build on a clean machine first, before trying on Amazon GameLift Streams. For instructions about testing on an Amazon EC2 instance, refer to Set up a remote machine.

Access denied when making a request to Amazon GameLift Streams service

If you encounter an "access denied" exception when trying to perform an Amazon GameLift Streams action or use resources, your AWS Identity and Access Management (IAM) role may have insufficient permissions. This is caused by making requests to the Amazon GameLift Streams service, such as a call to StartStreamSession.

Make sure that the affected IAM role's policy has proper permissions for Amazon GameLift Streams. Check the following:

- If the IAM role has an explicit "deny-all" policy, you must explicitly list Amazon GameLift Streams as an exception to that policy by adding `"gameliftstreams:*"` to the NotAction element. For example:

```
{
    "Sid": "DenyAllExceptListedIfNoMFA",
    "Effect": "Deny",
    "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice",
        "sts:GetSessionToken",
        "gameliftstreams:*"                    // Add this
    ],
    "Resource": "*",
    "Condition": {
        "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
    }
}
```

- For additional troubleshooting, review the Troubleshooting access denied error messages in the *IAM User Guide*.

Troubleshoot compatibility with Proton for Amazon GameLift Streams

If your Amazon GameLift Streams application runs on a Proton runtime environment, this section can help you troubleshoot compatibility issues between your application and the Proton layer. These instructions include a set of scripts that installs Proton to your own machine, simulating the environment that Amazon GameLift Streams would use. By troubleshooting without the Amazon GameLift Streams server, you can focus on troubleshooting issues specific to your application and the runtime environment.

Proton is a compatibility layer that allows Windows applications to run on Linux. As such, you must complete these troubleshooting steps by using an Ubuntu machine.

Topics

High-level steps

1. Acquire an Ubuntu 22.04 machine. You can use either your local machine or an Amazon EC2 cloud-based desktop. For instructions, refer to either Set up a local machine or Set up a remote machine.
2. Install the Proton runtime environment and debug your application. For instructions, refer to Troubleshoot on Proton.

Set up a local machine to troubleshoot Proton

In this step, you will set up your local Ubuntu machine, which you will use to troubleshoot your application's compatibility with Proton for Amazon GameLift Streams.

If you don't have an Ubuntu machine, you can set up a remote machine using Amazon EC2. Follow the steps in Set up a remote machine instead.

Prerequisites

- Ubuntu 22.04
- NVIDIA GPU

Install GPU drivers

Installing the latest GPU drivers can prevent your application from poor performance and crashes.

To check what GPU driver your system uses

1. Run the following command in a terminal:

```
lshw -C display | grep driver
```

2. If the correct drivers are installed, you should see the following output, or similar, where *<gpu>* is `nvidia` for NVIDIA: `configuration: driver=<gpu> latency=0`

To install the latest GPU drivers, complete the following steps for your GPU.

To install the latest NVIDIA GPU drivers

Follow the instructions in NVIDIA GPU installation for Ubuntu.

Verify GPU drivers

Verify that GPU drivers are installed and working correctly. One way to verify this is by running the vkcube application in a terminal.

1. Install the `vulkan-tools` apt package using the following command.

```
sudo apt install -y vulkan-tools
```

2. Run `vkcube`.
3. Review the output.
   - If your system is properly using the correct GPU, you will see output similar to the following, with the name of your GPU: `Selected GPU 0: AMD Radeon Pro V520 (RADV NAVI12), type: 2`

- If your application isn&t able to use the GPU correctly, you might see different output similar to the following: `Selected GPU 0: llvmpipe (LLVM 15.0.7, 256 bits), type: 4`

  In this case, check the GPU drivers and re-install if needed.

Next step

With your local Ubuntu machine ready, the next step is to set up Proton. For instructions, refer to Troubleshoot on Proton.

Set up a remote Amazon EC2 machine to troubleshoot Proton

If you don't have a local Ubuntu machine, follow these instructions to set up a remote machine instead.

In this step, you will set up your remote Ubuntu machine using Amazon Elastic Compute Cloud (Amazon EC2), which you will use to troubleshoot your application's compatibility with Proton for Amazon GameLift Streams. This topic describes how to set up an Amazon EC2 instance with Ubuntu 22 LTS, necessary GPU drivers, and the Amazon DCV Server for a visual remote desktop.

Launch an Amazon EC2 Instance with Ubuntu 22.04 LTS AMI

1. Navigate to Amazon EC2 in the AWS Management Console.
2. Select **Launch Instances**.
3. Enter "Amazon GameLift Streams Testing" for **Name**.
4. Select **Ubuntu Server 22.04 LTS (HVM)** for **Application and OS Images (Amazon Machine Image)**.
5. Select **g4dn.2xlarge** for **Instance Type**.
6. For **Key pair (login)**, choose a key pair if you want to use SSH to access the instance. We recommend using an instance profile with the `AmazonSSMManagedInstanceCore` policy to connect to your instances using AWS Systems Manager Session Manager. For more details, follow Adding Session Manager permissions to an existing IAM role.
7. For **Network settings**, create a new security group:
8. For **Security Group Name**, enter **DCV.**
9. Add **Inbound Security Group Rules** with **Type** `Custom TCP`, **Port Range** `8443`, and **Source Type** `Anywhere` to allow access using Amazon DCV.

10. Increase storage to at least **256GB** and choose **gp3** as the storage type.
11. Choose **Launch Instance**.

Your instance should now be launched.

Follow the instructions in Connect to your Linux instance to connect to the instance using SSH or AWS Systems Manager Session Manager.

Install GPU drivers

G4dn - NVIDIA GPU

Install additional modules and Linux firmware by running the following commands:

```
sudo apt install linux-modules-extra-aws linux-firmware

# Install the AWS CLI required for NVIDIA driver installation
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
```

Follow the instructions on the NVIDIA GRID drivers for Ubuntu and Debian in Install NVIDIA drivers on Linux.

Set up user environment

Set up your user environment so it can use the GPU by running the following commands. This does the following things:

- Add you to the `video` groups to give you access to a video device, and the `render` group to give you access to a rendering device.
- Install the AWS CLI, which is required for NVIDIA drivers and for downloading your applications or games from Amazon S3.

```
sudo adduser user

# Add the current user to the video and render group
sudo usermod -a -G video user
sudo usermod -a -G render user
sudo adduser user sudo

# Install the AWS CLI
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install

sudo reboot
```

Installation and configuration of Amazon DCV

Reconnect to the instance using SSH or AWS Systems Manager Session Manager and follow the instructions from Installing the Amazon DCV Server on Linux for Ubuntu.

- Verify that the server is correctly configured as described in the documentation.
- Follow the steps in Install and configure NVIDIA drivers for NVIDIA GPU.
- Add the Amazon DCV user to video group, as explained in step 7 of the Installing the Server guide.

There is no need to install any optional parts of the Amazon DCV Server.

When you're done, run the following command to start the Amazon DCV Server:

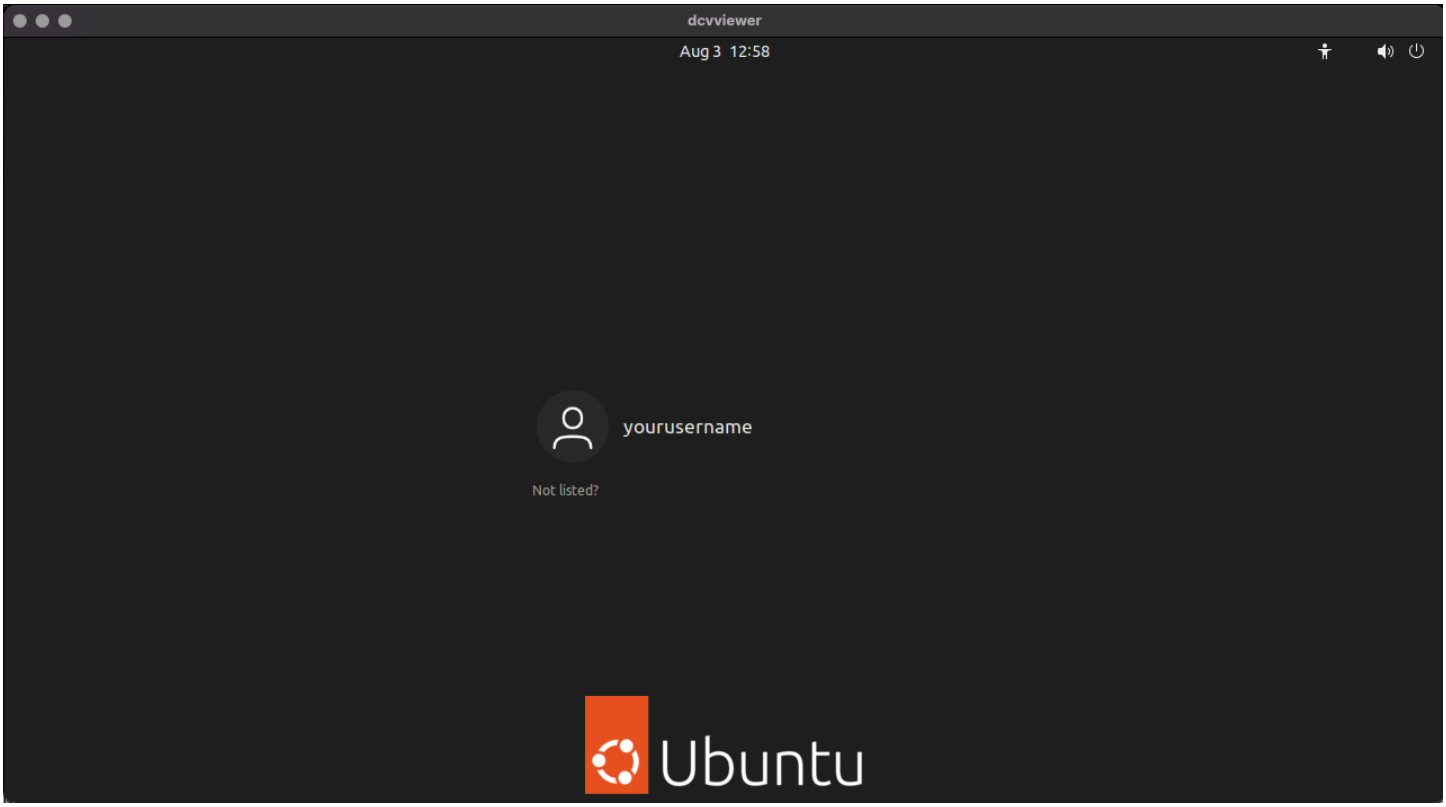
```
sudo systemctl start dcvserver sudo systemctl enable dcvserver
```

Connecting to the Ubuntu Server using the Amazon DCV client

Reconnect to your Ubuntu instance and create a session for a user by running:

```
sudo dcv create-session --owner user --user user my-session --type console
```

You can now use the Amazon DCV Client to access your Ubuntu instance using its public IP address. When you launch a Amazon DCV client, a window appears, allowing you to access your Ubuntu instance through a visual display.



Verify GPU drivers

Verify that GPU drivers are installed and working correctly. One way to verify this is by running the vkcube application in a terminal.

1. Install the `vulkan-tools` apt package using the following command.

   ```
   sudo apt install -y vulkan-tools
   ```

2. Run `vkcube`.
3. Review the output.
   - If your system is properly using the correct GPU, you will see output similar to the following, with the name of your GPU: `Selected GPU 0: AMD Radeon Pro V520 (RADV NAVI12), type: 2`
   - If your application isn&t able to use the GPU correctly, you might see different output similar to the following: `Selected GPU 0: llvmpipe (LLVM 15.0.7, 256 bits), type: 4`

     In this case, check the GPU drivers and re-install if needed.

Set up Podman (Proton only)

If you're using a Proton runtime, you must install Podman, a container that used by Proton's build process. Complete the following steps by using a terminal.

1. Install Podman, a container that Proton's build process uses.

```
sudo apt install podman
```

2. In the files `/etc/subgid` and `/etc/subgid`

   a. Verify that the files list your Linux machine user name and ID. You can either open the files or use the `cat` command to see what's in the files. Format example: `test:165536:65536`, where `test` corresponds to your user name.

   b. If they're not listed, add them in. Format example: `test:165536:65536`, where `test` corresponds to your user name.

```
$ cat /etc/subuid
               ceadmin:100000:65536
               test:165536:65536

               $ cat /etc/subgid
               ceadmin:100000:65536
               test:165536:65536
```

   For more information, refer to Basic Setup and Use of Podman in a Rootless environment in Podman's documentation.

Next step

You now have an Amazon EC2 instance and environment setup to troubleshoot compatibility issues with Amazon GameLift Streams. The next step is to set up Proton. For instructions, refer to Troubleshoot on Proton.

Troubleshoot compatibility on Proton

In this step, you will set up Proton on your own machine, so you can troubleshoot compatibility issues between your Amazon GameLift Streams application and Proton. Running your application

in a simulated environment without the Amazon GameLift Streams server can help you identify issues specific to your application and runtime environment.

Prerequisites

- Ubuntu 22.04 LTS. For instructions, refer to Set up a local machine or Set up a remote machine.

Install Proton

To install Proton on your Ubuntu 22.04 LTS machine, use the following script to clone, build, and configure the version of Proton you want to test from the Proton GitHub repository.

1. Copy and paste the following code into a file called `proton-setup.sh` on your Ubuntu 22.04 LTS machine.

```
#!/bin/bash
# This is a script to build Proton. The default build is a tag from the
# experimental_8.0 branch of Proton, but can be changed as a parameter to this
 script.
#
# Usage: ./proton-setup.sh [optional proton_branch_name {default:
 experimental-8.0-20240205}]
set -e

sudo apt install -y podman make git

# clone proton from github, recurse submodules
# if no proton git link is supplied, use a default tag from the experimental_8.0
 branch
PROTON_BRANCH=${1:-"experimental-8.0-20240205"}
PROTON_BUILD_DIR=protonBuild
PROTON_DIR=$(pwd)/proton
if git clone https://github.com/ValveSoftware/Proton.git --recurse-submodules --
branch $PROTON_BRANCH proton;
then
 echo "Successfully cloned Proton and its submodules."
else
 echo "Warning: a proton directory/repository already exists. It is recommended to
 delete this folder and re-run this script unless it is a valid repository with
 initialized submodules."
fi
```

```
if [ -d $PROTON_BUILD_DIR ];
then
 echo "Error: protonBuild directory already exists. Delete this folder first to
 create a fresh build of Proton before re-running this script."
 exit 1
fi
mkdir $PROTON_BUILD_DIR
cd $PROTON_BUILD_DIR
$PROTON_DIR/configure.sh --enable-ccache --container-engine=podman

# build proton
echo "Building Proton"
make
echo "Done building Proton!"

# prepare proton for execution
cd dist
mkdir compatdata
if [ -e ./dist ]; then
  PROTON_FILES=dist
elif [ -e ./files ]; then
  PROTON_FILES=files
fi
cp version $PROTON_FILES/
echo "Finished installing proton. Proton binary location: $(pwd)/proton"
echo "STEAM_COMPAT_DATA_PATH: $(pwd)/compatdata"
echo "STEAM_COMPAT_CLIENT_INSTALL_PATH: anything"
```

2. In this step you will run the Proton setup script to clone and install Proton and additional dependencies. The script accepts as an argument the tag or branch name for the Proton version you want to install. To simulate one of the custom builds of Proton that Amazon GameLift Streams provides, use the instructions for that version, below.

> **Note**
>
> Expect the cloning from GitHub to take some time. There are many submodules to download, totalling several GB.

In your terminal, run the `proton-setup.sh` script and specify the Proton version branch:

- **Built-in Proton versions**

- For Proton 8.0-5 (PROTON-20241007), use experimental-8.0-20240205.

```
proton-setup.sh experimental-8.0-20240205
```

Typically, no additional source code is needed. However, if you run into problems with Electra Media Player, (an Unreal Engine plugin) we recommend using the fixes found in https://github.com/ValveSoftware/wine/pull/257.

> **Note**
>
> For Proton 8.0-2c (PROTON-20230704), Amazon GameLift Streams uses a proprietary build, which is not available to buld locally.

- **Recommended custom Proton version**

  For a custom Proton version, we recommend using the Proton experimental_8.0 branch.

```
proton-setup.sh experimental_8.0
```

- **Other custom Proton versions**

  For other Proton versions, use an exact branch or tag name listed in Proton releases.

```
proton-setup.sh branch-or-tag-name
```

If your installation is successful, the output in your terminal should be similar to the following:

```
...
Done building Proton!
Finished preparing proton. Proton binary location: /home/test/protonBuild/dist/
proton
STEAM_COMPAT_DATA_PATH: /home/test/protonBuild/dist/compatdata
STEAM_COMPAT_CLIENT_INSTALL_PATH: anything
```

Take note of the following variables from the output because you will need them to run Proton in the next step:

- Proton binary location

- STEAM_COMPAT_DATA_PATH
- STEAM_COMPAT_CLIENT_INSTALL_PATH

Run your application on Proton

The following steps assume that the application executable is located in path/myapplication/bin/application.exe. Replace it with the path and file name for your application.

- In a terminal, navigate to the folder where your application executable is located.

```
cd path/myapplication/bin/application.exe
```

- Run your application on Proton. Use the Proton binary location and the environment variables that you got in the previous step.

```
STEAM_COMPAT_DATA_PATH=/home/test/protonBuild/dist/compatdata
 STEAM_COMPAT_CLIENT_INSTALL_PATH=anything /home/test/protonBuild/dist/proton run
 application.exe
```

The application should now attempt to start. If the application starts locally, but not on Amazon GameLift Streams, it may be due to a configuration issue when calling Amazon GameLift Streams APIs. Verify that the API call parameters are correct. Otherwise, continue to the next step for debugging.

Debug the application through log files

If your application has issues running on the local Proton environment, check the output log. The log contains output from your application and runtime environment. Trace where your application is failing to discover issues on the application side.

To dump the log output into a text file, such as proton.log, use the following command:

```
STEAM_COMPAT_DATA_PATH=/home/test/protonBuild/dist/compatdata
 STEAM_COMPAT_CLIENT_INSTALL_PATH=anything /home/test/protonBuild/dist/proton run
 application.exe &>proton.log
```

Proton also indicates if the issue is due to a Wine plugin, unimplemented function, missing dlls, and so on. For more information, see Wine HQ's Debugging Wine guide. If you find a Proton or

Wine error in the logs that you can't fix on the application side, reach out to your AWS Account Manager or post a question in AWS re:Post for help with further debugging.

Profiling Unreal Engine performance

In this section, learn how to analyze your Unreal Engine game or application performance. This can help you identify area's to optimize, leading to smoother streaming in Amazon GameLift Streams.

You can use Unreal Engine's console and its built-in stat commands to get a detailed look at your game's performance. You can access the console in a non-shippable build or the **Editor**. A non-shippable build refers to a project that was built using a debug or development configuration.

To access the console

In non-shippable builds and the Play In Editor mode, press the tilde (**~**) key to open the console. Double-press the tilde key to expand the console.

Here are some tips for using the console:

- Type a keyword to list all possible commands containing that keyword. Scroll through the list using the arrow keys.
- Scroll through the history by using the arrow keys or Page up and Page down keys.
- Logs are saved in a `.txt` file in your project's `Saved/Logs` directory

To profile your game's performance

1. Start by running the `stat fps` and `stat unit` commands. This will give you an overview of where your game struggles with performance.

   - `stat fps`: Shows the current frames per second.
   - `stat unit`: Breaks down the frame into several subsections.
     - **Frame**: Total wall-clock time starting from when the simulation of the frame starts to when the presentation of the frame is on the screen.
     - **Game**: Total CPU time taken by the game simulation thread per frame.
     - **Draw**: Total CPU time for the rendering threads to translate the scene to commands for the GPU and submit them to the GPU.
     - **GPU**: Total time for the GPU to process all commands.

- **Draws**: Total number of draws submitted for the frame.
- **Prims**: Total number of triangles drawn.

2. Play through the game and identify areas with low performance, indicated by decreased FPS and increased time in **Game**, **Draw**, or **GPU**.
3. Run `stat game` to see how time is spent for the various gameplay groups.
4. Refine the stats for specific gameplay factors like AI, animation, physics, gameplay, scripting, and so on. Here are a few examples:

   - `stat ai`: Time to compute AI behavior.
   - `stat anim`: Time to compute skinned meshes.
   - `stat physics`: Time to compute physics simulations.

5. Run `stat drawcount` to see which render areas generate the most draws. The list shows the render passes that emit draws, and the number of draws emitted each frame. You can get more information by analyzing the GPU stats in the next step.
6. Run `stat gpu` to see which render types consume the most GPU time.
7. Refine the rendering types into broad groups, such as lights, shadows, lumen (lighting), hair, post processing, and so on. Here are a few common examples:

   - `stat lightrendering`: GPU time to render lights and shadows.
   - `stat shadowrendering`: GPU time to update the various shadows.
   - `stat scenerendering`: GPU time to render the scene.

This section covers only a subset of available commands. Depending on your game's features, look into stats for areas such as asset streaming, virtual texturing, CPU task workload distribution, threading, sound, particles, and so on. For more information, refer to Stat commands.

Regions, quotas, and limitations

Amazon GameLift Streams is available across multiple AWS Regions, offering dual-stack service endpoints that support both IPv4 and IPv6 connectivity. The service operates from primary locations including US East (Ohio), US West (Oregon), Asia Pacific (Tokyo), and Europe (Frankfurt), with the ability to manage additional AWS Regions and locations, collectively referred to as *remote locations*, for optimized latency and stream quality.

The service infrastructure is governed by three main categories of constraints:

- Service quotas
- API rate limits
- Fixed service limitations

These include restrictions on application sizes, number of applications per region, file management capacities, and GPU allocations across different stream classes and regions. The service implements specific API rate limits for various operations, ranging from 1 to 20 requests per second, ensuring stable service performance. Additionally, there are fixed service limitations concerning stream group configurations, GPU deployments, and application associations that apply uniformly across all customers.

AWS Regions and remote locations supported by Amazon GameLift Streams

An AWS Region is a collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the other Regions. For general information about AWS Regions, see Managing AWS Regions in the *AWS General Reference*.

The following table lists the AWS Regions where the Amazon GameLift Streams service is available and the endpoints for each Region. You create all Amazon GameLift Streams application and stream group resources in a specified Region, whether you work in the Amazon GameLift Streams console, use the AWS Command Line Interface (AWS CLI), or make programmatic calls. The Region where you create these resources is known as the *primary location*. Use your primary location's endpoint to connect to the Amazon GameLift Streams service programmatically.

Service endpoints

Amazon GameLift Streams supports dual-stack service endpoints, allowing clients and resources to interact with the service using IPv6 or IPv4.

Region Name	Region	Endpoint	Protocol	
US East (Ohio)	us-east-2	gameliftstreams.us-east-2.api.aws	HTTPS	
US West (Oregon)	us-west-2	gameliftstreams.us-west-2.api.aws	HTTPS	
Asia Pacific (Tokyo)	ap-northeast-1	gameliftstreams.ap-northeast-1.api.aws	HTTPS	
Europe (Frankfurt)	eu-central-1	gameliftstreams.eu-central-1.api.aws	HTTPS	

Remote locations

Amazon GameLift Streams can extend coverage to remote locations, enabling you to host your application and stream sessions in more locations. The remote locations available to you depend on your primary location. We recommend that you choose locations that are geographically close to your users to optimize latency and stream quality.

Primary location	Remote locations		
US East (Ohio) – `us-east-2`	• US East (N. Virginia) – `us-east-1` • US West (Oregon) – `us-west-2`		

Primary location	Remote locations		
	• Europe (Ireland) – `eu-west-1` • Europe (Frankfurt) – `eu-central-1` • Asia Pacific (Tokyo) – `ap-northeast-1`		
US West (Oregon) – `us-west-2`	• US East (N. Virginia) – `us-east-1` • US East (Ohio) – `us-east-2` • Europe (Ireland) – `eu-west-1` • Europe (Frankfurt) – `eu-central-1` • Asia Pacific (Tokyo) – `ap-northeast-1`		
Asia Pacific (Tokyo) – `ap-northeast-1`	• US East (N. Virginia) – `us-east-1` • US West (Oregon) – `us-west-2` • US East (Ohio) – `us-east-2` • Europe (Ireland) – `eu-west-1` • Europe (Frankfurt) – `eu-central-1`		

Primary location	Remote locations		
Europe (Frankfurt) – `eu-central-1`	• US East (N. Virginia) – `us-east-1` • US West (Oregon) – `us-west-2` • US East (Ohio) – `us-east-2` • Europe (Ireland) – `eu-west-1` • Asia Pacific (Tokyo) – `ap-northeast-1`		

Amazon GameLift Streams service quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. For more information, see AWS service quotas in the *AWS General Reference*.

Service quotas

In the following table, the GPU quotas are all 0 by default. However, your account's applied quotas might be different. To check, sign in to the AWS Management Console and open the Service Quotas console to Amazon GameLift Streams, where you can review your current quotas in the **Applied account-level quota value** column and submit a request to increase these values.

Name	Default	Adjustable	Description
Application size (GiB)	Each supported Region: 100	Yes	The maximum total size (in GiB) of an application, in this account. Note that

Name	Default	Adjustable	Description
			a gibibyte (GiB) equals 1024*1024*1024 bytes.
Applications	Each supported Region: 5	Yes	The maximum number of applications that you can create in this account, per AWS Region.
Files per application	Each supported Region: 30,000	Yes	The maximum number of files that you can have in an application, in this account.
Gen4n GPUs, ap-northeast-1	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the ap-northeast-1 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".
Gen4n GPUs, eu-central-1	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the eu-central-1 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".

Name	Default	Adjustable	Description
Gen4n GPUs, eu-west-1	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the eu-west-1 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".
Gen4n GPUs, us-east-1	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the us-east-1 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".
Gen4n GPUs, us-east-2	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the us-east-2 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".

Name	Default	Adjustable	Description
Gen4n GPUs, us-west-2	Each supported Region: 0	Yes	The maximum number of Gen4n GPUs you can allocate for streaming in this account in the us-west-2 Region. Some stream classes support running more than one application per GPU, such as "Gen4n_high".
Gen5n GPUs, ap-northeast-1	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the ap-northeast-1 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".
Gen5n GPUs, eu-central-1	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the eu-central-1 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".

Name	Default	Adjustable	Description
Gen5n GPUs, eu-west-1	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the eu-west-1 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".
Gen5n GPUs, us-east-1	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the us-east-1 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".
Gen5n GPUs, us-east-2	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the us-east-2 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".

Name	Default	Adjustable	Description
Gen5n GPUs, us-west-2	Each supported Region: 0	Yes	The maximum number of Gen5n GPUs you can allocate for streaming in this account in the us-west-2 Region. Some stream classes support running more than one application per GPU, such as "Gen5n_high".
Stream groups	Each supported Region: 2	Yes	The maximum number of stream groups that you can create in this account, per AWS Region. A stream group is a collection of compute resources that stream your application to end users.

Amazon GameLift Streams API rate limits

These limits reflect the maximum rate of requests per second from your AWS account to the Amazon GameLift Streams service in an AWS Region.

API operation	Requests per second
AddStreamGroupLocations	5
AssociateApplications	5
CreateApplication	5
CreateStreamGroup	1

API operation	Requests per second
CreateStreamSessionConnection	20
DeleteApplication	5
DeleteStreamGroup	5
DisassociateApplications	5
ExportStreamSessionFiles	20
GetApplication	10
GetStreamGroup	10
GetStreamSession	20
ListApplications	10
ListStreamGroups	10
ListStreamSessions	20
ListStreamSessionsByAccount	20
ListTagsForResource	10
RemoveStreamGroupLocations	5
StartStreamSession	20
TagResource	10
TerminateStreamSession	20
UntagResource	10
UpdateApplication	5
UpdateStreamGroup	5

Other Amazon GameLift Streams limitations

This page lists other limitations to be aware of as you create your streaming solution. These limits are fixed within the service for all customers.

Name	Limitation	Description
Applications in a stream group	100	The maximum number of Amazon GameLift Streams applications that can be associated to a stream group.
GPUs in a stream group	2500	The maximum number of GPUs in a stream group across all Regions and remote locations.
Single file size (GiB)	80 GiB	The maximum size (in GiB) of a single file in an application. Note that a gibibyte (GiB) equals 1024*1024*1024 bytes.
Stream group associations per application	100	The maximum number of stream groups that an Amazon GameLift Streams application can be associated to.

Managing usage and bills for Amazon GameLift Streams

This topic covers ways you can review and manage your Amazon GameLift Streams usage and bills.

Also see the Amazon GameLift Streams Pricing page for the following information:

- **Cost breakdown**: Understand what AWS charges you for when you use Amazon GameLift Streams.
- **Amazon GameLift Streams rates**: See how much Amazon GameLift Streams costs and compare different options.
- **Stream capacity reservation**: Plan ahead and ensure that you have enough stream capacity to meet your customer demands.

Review your Amazon GameLift Streams bills and usage

You can review your Amazon GameLift Streams bills and usage by using the AWS Billing and Cost Management tools in the AWS Console or AWS CLI.

To view your bill through the AWS Console, refer to Viewing your bill in the AWS Billing User Guide.

To view your bill through the AWS CLI, call `GetCostAndUsage` using the Billing and Cost Management API. For example, use the following command to retrieve a monthly bill for Amazon GameLift Streams, and replace the dates with ones relevant to you.

Example : Use `GetCostAndUsage` API to view bill

```
aws ce get-cost-and-usage /
    --time-period Start=2023-01-01,End=2023-01-31 /
    --granularity MONTHLY /
    --metrics BlendedCost /
    --filter Amazon GameLift Streams-bill-filter.json
```

where the filter, such as `Amazon GameLift Streams-bill-filter.json`, specifies the Amazon GameLift Streams service as follows:

```
{
    "Dimensions": {
        "Key": "SERVICE",
        "Values": ["Amazon Amazon GameLift Streams"]
    }
}
```

Best practices to manage Amazon GameLift Streams costs

We strongly recommend that you use the following tools and techniques to manage your Amazon GameLift Streams costs to avoid unexpected costs.

Create billing alerts to monitor usage

Set up billing alerts using AWS Budgets, which enables you to track your costs and usage, and respond quickly to alerts to avoid unexpected costs. You can also configure the billing alert to trigger actions that help you stay within budget. By default, budgets include all of your AWS services. To specify a budget for Amazon GameLift Streams only, add a budget filter.

For more information, see the following topics:

- Creating a budget
- Best practices for AWS Budgets

Scale stream groups to zero capacity

Allocated stream capacity continues to incur costs even when they're not currently hosting stream sessions. Scale stream groups to zero capacity when not in use to avoid unnecessary cost. This prevents your stream group from allocating resources. When you set always-on and on-demand stream capacity to zero, all connected streams end. When you're ready, you can reuse your stream group by scaling capacity back up.

For instructions, refer to Edit capacity.

> **Warning**
>
> Avoid deleting a stream group, unless you don't plan to use the stream group again. If you delete a stream group, you cannot restore the original stream group and must create a new one.

Delete original application files

To optimize storage cost, you can delete the original application files that you uploaded to an Amazon S3 bucket. It's safe to delete the files if the application is in **Ready** status. At that point, Amazon GameLift Streams has a snapshot of the application files and no longer accesses your original files.