Amazon AppStream 2.0

Developer Guide

Amazon AppStream 2.0: Developer Guide

Table of Contents

What Is Amazon AppStream 2.0?

Amazon AppStream 2.0 is a fully managed, secure, application streaming service that allows you to stream desktop applications from AWS to any device running a web browser, without rewriting them. AppStream 2.0 provides users instant-on access to the applications they need, and a responsive, fluid user experience on the device of their choice.

With AppStream 2.0, you can easily add your existing desktop applications to AWS and instantly start streaming them to an HTML5 compatible browser. You can maintain a single version of each of your apps, which makes application management easier. Your users always access the latest versions of their applications. Your applications run on AWS compute resources, and data is never stored on users' devices, which means they always get a high performance, secure experience.

Unlike traditional on-premises solutions for desktop application streaming, AppStream 2.0 offers pay-as-you-go pricing, with no upfront investment and no infrastructure to maintain. You can scale instantly and globally, ensuring that your users always have the best possible experience.

For more information, see the AppStream 2.0 detail page, AppStream 2.0 Pricing or Amazon AppStream 2.0 FAQs.

The AppStream 2.0 API provides programmatic control over all streaming actions as an alternative to using the AWS Management Console. For more information, see Amazon AppStream 2.0 API Reference.

Topics

Features

Using Amazon AppStream 2.0 leverages the following advantages:

Run desktop applications on any device

With AppStream 2.0, your desktop applications can run securely in an HTML5 web browser on Windows and Linux PCs, Macs, and Chromebooks. You can add your applications without rewriting them, maintain a single version for all your users, and provide easy access to your users from anywhere.

Instant-on access

AppStream 2.0 provides users instant-on access to their desktop applications in a browser on the desktop device of their choice. There are no delays, no large files to download, and no time-consuming installations. Users get a responsive, fluid experience that is indistinguishable from natively installed apps.

Secure applications and data

With AppStream 2.0, applications and data remain on AWS — only encrypted pixels are streamed to end users. Applications run on an AppStream 2.0 instance dedicated to each user so that compute resources are not shared. Applications can run inside your own VPC, and you can use Amazon VPC security features to control access. This allows you to isolate your applications and deliver them in a secure way.

Easily integrate with your IT environment

AppStream 2.0 can integrate with your existing AWS services, and your on-premises environments. By running applications inside your VPCs, your users can access data and other resources that you're running on AWS, reducing the movement of data between AWS and your location and providing a faster user experience. AppStream 2.0 supports identity federation, which allows your users to access their applications using their corporate credentials. You can also allow authenticated access to your IT resources from applications running on AppStream 2.0.

Fully managed service

With AppStream 2.0, you don't need to plan, deploy, manage, or upgrade any application streaming infrastructure. AppStream 2.0 manages the AWS resources required to host and run your applications, scales automatically, and provides access to your end users on demand.

Consistent, scalable performance

AppStream 2.0 runs on AWS with access to compute capabilities not available on local devices, which means that your applications run with consistently high performance. You can instantly scale locally and globally, and ensure that your users always get a low-latency experience. Unlike on-premises solutions, you can quickly deploy your applications to a new AWS region that is closest to your users, and start streaming with no incremental capital investment.

Key Concepts

To get the most out of AppStream 2.0, be familiar with the following concepts:

stack

Set up an AppStream 2.0 stack to start streaming apps to user browsers. An AppStream 2.0 stack consists of a fleet of streaming instances, user access policies, and storage configurations.

fleet

The fleet in an AppStream 2.0 stack consists of streaming instances that can scale automatically based on demand. You can set the desired number of streaming instances in a fleet. The fleet runs the image that you specify.

image

An AppStream 2.0 image contains applications to be streamed to users accessing an AppStream 2.0 stack. The image is used to launch streaming instances that are part of an AppStream 2.0 fleet. Use an image builder to create or modify an image.

image builder

Install your apps and create an image by using an AppStream 2.0 image builder. You can launch and connect to an AppStream 2.0 image builder from the AWS Management Console. After you are connected to an image builder, you can install, add, and test your apps, and then use the image builder to publish an AppStream 2.0 image.

How to Get Started

If you are using AppStream 2.0 for the first time, follow the Getting Started with Amazon AppStream 2.0 (p. 8) tutorial in the console. When you go through the getting started experience for the first time, AppStream 2.0 creates an IAM role to create and manage AppStream 2.0 resources on your behalf.

To use the Try It Now feature

You can go directly to Try it now or follow these steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Try it now**.
3. Sign in using your AWS account credentials, if requested.
4. Read the terms and conditions and choose **Agree and Continue**.
5. From the list of applications shown, select one to try.

To run the interactive tutorial

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Get Started**.
3. Select the option to learn more about AppStream 2.0 resources.

Related Services

AppStream 2.0 is used in conjunction with the following AWS service:

AWS Identity and Access Management

IAM is a web service that helps you securely control access to AWS resources for your users. The AppStream 2.0 service uses IAM service roles to create and manage AppStream 2.0 resources on your behalf. You can use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization). For more information, see Using Identity Based Policies (p. 50).

Accessing AppStream 2.0

You can work with AppStream 2.0 in any of the following ways:

AWS Management Console

The console is a browser-based interface to manage AppStream 2.0 resources. For more information about using the console, see Getting Started with Amazon AppStream 2.0 (p. 8).

AWS command line tools

AWS provides two sets of command line tools: the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell. For more information, see the AWS Command Line Interface User Guide and AWS Tools for Windows PowerShell User Guide.

You can use the AWS command line tools to issue commands at your system's command line to perform AppStream 2.0 and AWS tasks. To use the AWS CLI to run AppStream 2.0 commands, see Amazon AppStream 2.0 Command Line Reference.

AWS SDKs

You can access AppStream 2.0 from a variety of programming languages. The SDKs automatically take care of tasks such as the following:

- Setting up an AppStream 2.0 stack or fleet
- Getting an application streaming URL to your stack
- Describing your resources

For more information about available SDKs, see Tools for Amazon Web Services.

Setting Up for Amazon AppStream 2.0

Complete the following tasks to get set up for Amazon AppStream 2.0.

Topics

Sign Up for AWS

When you sign up for AWS, your AWS account is automatically signed up for all services, including AppStream 2.0. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Note your AWS account number, because you need it for the next task.

Create an IAM User

Services in AWS, such as AppStream 2.0, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. The console requires your password. You can create access keys for your AWS account to access the command line interface or API.

We recommend that you use IAM to access AWS instead of the credentials for your AWS account. Create an IAM user, and then add the user to an IAM group with administrative permissions or and grant this user administrative permissions. You can then access AWS using a special URL and the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

To create an IAM user for yourself and add the user to an Administrators group

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**, and then choose **Add user**.
3. For **User name**, type a user name, such as `Administrator`. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 64 characters in length.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to select a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, type the name for the new group. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 128 characters in length.
9. For **Filter**, choose **Job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
12. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users, and to give your users access to your AWS account resources. To learn about using policies to restrict users' permissions to specific AWS resources, go to Access Management and Example Policies for Administering AWS Resources.

To sign in as this new IAM user, sign out of the AWS console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name @ your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, choose **Create Account Alias** and enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **IAM users sign-in link** on the dashboard.

For more information about IAM, see the AWS Identity and Access Management User Guide.

(Optional) Install the AWS CLI

Note
This step is not required to use the first-run wizard and create your first stack.

The AWS Command Line Interface provides high-level commands to simplify creating, updating, and monitoring application streaming from a local development environment. For information about installing the AWS CLI or upgrading it to the latest version, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Getting Started with Amazon AppStream 2.0

To stream your applications, Amazon AppStream 2.0 requires an environment consisting of a stack and at least one application image. This topic walks through the steps needed to understand how to put together a sample AppStream 2.0 environment for application streaming and give users access to that stream.

Topics

Step 1: Set Up a Sample Stack

Before you can stream your applications, you need to create a stack. In this step, you create a new stack from the sample stack template to simplify the creation.

To set up the AppStream 2.0 sample stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Get Started** if you are visiting the service console for the first time, or **Quick Links** from the left navigation menu of the service console.
3. Choose **Set up with sample apps**, choose **Next** to use the default sample stack name and details, or enter your own details and then choose **Next**.
4. Review the sample apps image details and choose **Next** to use the default image. The sample apps image contains a few pre-installed open source applications for evaluation purposes.
5. Provide the details for your fleet. Note that most values are pre-populated for you.

   Instance Type — Choose an instance type that matches the performance requirements of your applications. All streaming instances in your fleet launch with the instance type that you select.

   Network Access — Provide two subnets that have access to the network resources with which your applications need to interact. If you don't have any subnets, create them using the help link provided and then refresh the subnets list. You can choose existing network settings or create new settings for this fleet. For more information, see Network Settings for Fleet and Image Builder Instances (p. 25).

Disconnect Timeout — Select the time that a streaming instance should remain active after users disconnect. If users try to reconnect to the streaming session after a disconnection or network interruption within this time interval, they are connected to the previous session. If users try to connect after this timeout interval, a session launches with a new instance.

Minimum Capacity — Choose a minimum capacity for your fleet based on the minimum number of users that are expected to be connected at the same time. Capacity is defined in terms of number of instances within a fleet, and every unique user session is served by an instance. For example, to have your stack support 100 concurrent users during low demand, enter the minimum capacity as 100. This ensures that 100 instances are running even if there are fewer than 100 users. If you are unsure about minimum capacity, accept the default value.

Maximum Capacity — Choose a maximum capacity for your fleet based on the maximum number of users that are expected to be connected at the same time. Capacity is defined in terms of number of instances within a fleet, and every unique user session is served by an instance. For example, to have your stack support 500 concurrent users during high demand, enter the maximum capacity as 500. This ensures that up to 500 instances can be created on demand. If you are unsure about maximum capacity, accept the default value.

Scaling Details (Advanced) — This section contains default scaling policies that can increase and decrease the capacity of your fleet under specific conditions. Expand this section to change the default scaling policy values. Regardless of scaling policy, your fleet size is always in the range of values specified by **Minimum Capacity** and **Maximum Capacity**.

We recommend that you accept the default values and choose **Next**. You can change these values after fleet creation. For more information, see Fleet Auto Scaling for Amazon AppStream 2.0 (p. 38).

6. Home Folders offer persistent storage for AppStream 2.0 streaming sessions. For more information, see Persistent Storage with AppStream 2.0 Home Folders (p. 19). We recommend that you enable this option and choose **Review**.
7. Review the details for the sample stack, choose **Edit** for any section to change, and then choose **Create**.

After the service sets up some resources, the **Stacks** dashboard appears. Your new stack is listed as **Active** when it is available to work with from the console, but cannot be used for streaming sessions until the stack fleet is in **Running** status.

Step 2: Provide Access to Users

After you create a stack, each user needs an active URL for access. This step automatically creates a streaming URL that you can share with a user for access to apps.

To provide access to users

1. In the navigation pane, choose **Stacks**, select the stack to use, check that the fleet status is Running, and then choose **Actions**, **Create Streaming URL**.
2. For **UserID**, type the user ID and select an expiration time. This time determines how long the generated URL is valid.
3. To view the user ID and URL, choose **Get URL**.
4. To copy the link to the clipboard, choose **Copy Link**.
5. Choose **Exit**.

Next Steps

You can now create URLs and send them to users. At this point, you can monitor your stack and make decisions about managing it. For more information, see the following topics.

- Learn how to use the AppStream 2.0 image builder to add your own apps and create new images you can stream. For more information, see Tutorial: Using an AppStream 2.0 Image Builder (p. 11).
- Provide persistent storage for your session users with AppStream 2.0 Home Folders. For more information, see Persistent Storage with AppStream 2.0 Home Folders (p. 19).
- Manage your AppStream 2.0 resources to optimize your streaming performance, automatic scaling, and cost structure. For more information, see Managing Amazon AppStream 2.0 Resources (p. 35).
- Control who has access to your AppStream 2.0 streaming instances. For more information, see Controlling Access to Amazon AppStream 2.0 (p. 49) and Enabling Single Sign-on Access to AppStream 2.0 Using SAML 2.0 (p. 30).
- Monitor your AppStream 2.0 resources using Amazon CloudWatch. For more information, see Monitoring Amazon AppStream 2.0 with Amazon CloudWatch (p. 45).
- Troubleshoot your AppStream 2.0 streaming experience. For more information, see Troubleshooting (p. 56).

Tutorial: Using an AppStream 2.0 Image Builder

Before you can stream your applications, Amazon AppStream 2.0 requires at least one image that you create using an image builder. This tutorial walks through the steps to create images using an image builder.

Important
After you create an image builder and it is running, your account may incur nominal charges. For more information, see AppStream 2.0 Pricing.

Topics

Step 1: Create an Image Builder

Create a new image builder so you can add apps and create images for streaming.

To create an image builder for adding applications

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. You may see the welcome screen showing two choices: **Try it now** and **Get started**. Choose **Get started**, **Custom set up**. If you do not see the welcome screen, choose **Quick links** in the left navigation pane, then **Custom set up**.
3. Select a base image. If you are launching the image builder for the first time, you can use the sample base image provided by AWS (seelcted by default). If you have created images before or to update applications in an existing image, you can choose one of your existing images. Choose **Next**.
4. Configure the image builder by accepting the default values or by providing your own inputs for the following fields:

Name

Provide a unique name identifier for the image builder.

Instance Type

Select the instance type for the image builder. Choose a type that matches the performance requirements of the applications that you plan to install.

Network Access

Choose a VPC subnet in which your image builder should be launched. Your image builder will have access to any of the network resources that are accessible from within this VPC subnet. For Internet access on the image builder, choose **Default Internet Access**, select a VPC that has public subnets on your default VPC, and then select one of the public subnets listed for **Subnet**. If you are controlling Internet access using a NAT gateway, leave **Default Internet Access** unselected and use the VPC with the NAT gateway. For more information, see Network Settings for Fleet and Image Builder Instances (p. 25).

After you have configured your image builder, choose **Review**.

5. Review the details for the image builder, choose **Edit** for any section to change, and choose **Launch**.

After the service sets up some resources, the image builder instance list appears. Your new image builder is listed as **Running** when it is ready to use (choose **Refresh** to update the status).

Step 2: Installing Applications to an Image

In this step, connect to the image builder that you created and launched, then install the applications to be included in the image.

To install applications

1. On the left navigation pane, choose **Images**, **Image Builder**.
2. Select the image builder to use, check to be sure it has a Running status, and choose **Actions**, **Connect**. For this step to work, you may need to configure your browser to allow pop-ups from https://stream.<aws-region>.amazonappstream.com/.
3. When you connect to your image builder, the service requests that you press "Ctrl + Alt + Delete" for first time sign-in. On the top right corner of the image builder session toolbar, choose **Admin Commands**, **Send Ctrl + Alt + Delete**.
4. Sign in by selecting one of thee following options:

   ImageBuilderAdmin

   This mode has full administrator privileges on the image builder instance. Use this mode to install your applications, add applications to the image, and create an image.

   ImageBuilderTest

   This mode has the same limited privileges as your end users have on their streaming instances. Use this mode to test applications for proper function as an end user.

   At any point after logging in, you can switch between admin and test modes by selecting the appropriate option from the **Admin Commands** menu.
5. If the image builder session requests you to enter a password, choose **Admin Commands**, **Log me in**.

6. Install apps by browsing to an application website or other download source and beginning the installation process as you normally would on a local physical computer. Complete the application's own installation process before moving to the next step.

Step 3: Adding Applications to an Image

In this step, you can add applications (*.exe*), batch scripts (*.bat*), and application shortcuts (*.lnk*) to the image.

To add applications

1. From the image builder desktop, launch the application named **Image Assistant**.
2. Choose **Add Application** and navigate to the location of the application, script, or shortcut to add. Choose **Open**.
3. In the **Application Properties** dialog box, enter a display name to be shown to the users in the catalog, change the icon, and enter launch parameters (additional arguments passed to the application when it is launched). Repeat for each application to add to the image.
4. When you are finished adding apps, choose **Next**.

To test your applications

- Verify that the apps you've added launch correctly by starting a Windows session as a test user.

  a. On the web toolbar in the top right of your session, choose **Admin Commands**, **Switch to Image Builder Test**, **ImageBuilderTest**.
  b. If a password is requested, choose **Admin Commands**, **Log me in**.
  c. Use the **Image Assistant** to launch and test your apps.
  d. To return to the admin mode, choose **Admin Commands**, **Switch to Image Builder Admin**.

  Note
  Do not exit the **Image Assistant** application, you need to use it in the next section.

Step 4: Optimizing Apps

In this step, you optimize your apps and create the image. The image builder optimizes your applications for start-up performance. This is a mandatory step that is performed on all applications in the list. All applications must be launched prior to optimization

To optimize your applications

1. Choose **Launch** and the service automatically launches the first application in your list. When the app is completely started, choose **Continue**.
2. Provide any interactions or inputs that may be required by the application launched to bring it to a usable state. For example, a web browser may prompt you to import settings before it is completely up and running.

3. After you have brought the application to a usable state, choose **Continue** in the small dialog box. The app helper launches the next application automatically.
4. Repeat the previous step until all applications are launched, and leave them running. In the **Image Assistant** helper app, choose **Next**.

Step 5: Creating an Image

In this step, you choose a name and create the image.

To create the image

1. Enter a unique image name and image display name, with an optional description, and choose **Next**. The name you choose cannot begin with "Amazon", "AWS", or "AppStream".
2. Review the details and choose **Disconnect and Create Image**. Your new image is created and the session is disconnected. You can now close the session window.
3. Return to the console and navigate to **Images**, **Image Registry**. Verify your new image appears in the list.

The new image first appears with status **Pending** in the image registry of your console. After the image is successfully created, the status of the image changes to **Available**, which means you can now use the image to launch a stack and stream your applications. To continue to work with creating images, you can start the image builder and connect to it from the console, or create a new image builder. There is a limit of five image builders per account.

Step 7: Clean Up

Finally, stop your running image builders to free up resources and avoid unintended charges to your account. We recommend stopping any unused, running image builders. For more information, see AppStream 2.0 Pricing.

To stop a running image builder

1. In the navigation pane, choose **Images**, **Image Builders**, and select the running image builder instance.
2. Choose **Actions**, **Stop**.

Set Up AppStream 2.0 Stacks and Fleets

To stream your applications, Amazon AppStream 2.0 requires an environment consisting of a stack, an associated fleet and at least one application image. This topic walks through the steps needed to understand how to set up a stack and a fleet, and how to give users access to the stack. If you haven't already done so, we recommend that you go through the procedures in Getting Started with Amazon AppStream 2.0 (p. 8) before using this topic.

Topics

Set Up a Fleet

Set up and create a fleet from which user applications are executed and streamed.

To set up and create a fleet

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. You may see the welcome screen showing two choices: **Try it now** and **Get started**. Choose **Get started**, **Skip**. If you do not see a welcome screen, move on to the next step.
3. In the left navigation pane, choose **Fleets**.
4. Choose **Create Fleet** and provide a fleet name, optional display name, and optional description. Choose **Next**.

5. Choose an image with the applications to stream and choose **Next**. If you don't have an image to use, see Tutorial: Using an AppStream 2.0 Image Builder (p. 11).
6. Provide details for your fleet by providing inputs for the following fields:

   Instance Type

   Choose an instance type that matches the performance requirements of your applications. All streaming instances in your fleet launch with the instance type that you select.

   Network Access

   Select a VPC and two subnets that have access to the network resources with which your applications need to interact. If you don’t have any subnets, create them using the help link provided and then refresh the subnets list. You can choose existing network settings or create new settings for this fleet. For Internet access on the fleet using your default VPC or with a VPC with a public subnet, choose **Default Internet Access**. For **VPC**, select your default VPC or VPC with a public subnet. For **Subnet**, select one or two public subnets. If you are controlling Internet access using a NAT gateway, leave **Default Internet Access** unselected and use the VPC with the NAT gateway. For more information, see Network Settings for Fleet and Image Builder Instances (p. 25).

   Disconnect Timeout

   Select the time that a streaming instance should remain active after users disconnect. If users try to reconnect to the streaming session after a disconnection or network interruption within this time interval, they are connected to the session instance they were disconnected from. If users try to connect after this timeout interval, a session launches with a new instance.

   Minimum Capacity

   Choose a minimum capacity for your fleet based on the minimum number of users that are expected to be connected at the same time. Capacity is defined in terms of number of instances within a fleet, and every unique user session is served by an instance. For example, to have your stack support 100 concurrent users during low demand, enter the minimum capacity as 100. This ensures that 100 instances are running even if there are fewer than 100 users. If you are unsure about minimum capacity, accept the default value.

   Maximum Capacity

   Choose a maximum capacity for your fleet based on the maximum number of users that are expected to be connected at the same time. Capacity is defined in terms of number of instances within a fleet, and every unique user session is served by an instance. For example, to have your stack support 500 concurrent users during high demand, enter the maximum capacity as 500. This ensures that up to 500 instances can be created on demand. If you are unsure about maximum capacity, accept the default value.

   Scaling Details (Advanced)

   This section contains default scaling policies that can increase and decrease the capacity of your fleet under specific conditions. Expand this section to change the default scaling policy values. Regardless of scaling policy, your fleet size is always in the range of values specified by **Minimum Capacity** and **Maximum Capacity**. We recommend that you accept the default values and choose **Review**. You can change these values after fleet creation. For more information, see Fleet Auto Scaling for Amazon AppStream 2.0 (p. 38).
7. Review the details for the fleet, choose **Edit** for any section to change, and choose **Create**.

Upon completion of the previous steps, the initial status of your new fleet is listed as **Starting** in the **Fleets** dashboard. The fleet needs to be in **Running** status to be associated with a stack and used for streaming sessions. Over the next few minutes, the service sets up some resources and the fleet moves to **Running** status. Wait for the fleet to be in **Running** status before attempting to use it for streaming sessions.

Set Up a Stack

Set up and create a stack to control access to your fleet.

To set up and create a stack

1. On the left navigation pane, choose **Stacks**, **Create Stack**.
2. Provide a stack name, optional display name and description. For **Fleet**, select the fleet to associate with your stack. Choose **Next**.
3. To enable or disable persistent storage for the stack users, select or clear the **Enable Home Folders** check box For more information, see Persistent Storage with AppStream 2.0 Home Folders (p. 19).

   Choose **Review**.
4. Review the details for the stack, choose **Edit** for any section to change, and choose **Create**.

Upon completion of the previous steps, the status of your new stack is listed as **Active** in the **Stacks** dashboard. This signifies that the stack is available to work with from the console, but it cannot be used for streaming sessions until the associated fleet is in **Running** status.

Provide Access to Users

After you create a stack with an associated fleet, each user needs an active URL to access it. This procedure automatically creates a streaming URL that you can share with a user for access to apps.

To provide access to users

1. On the left navigation pane, choose **Stacks**, select a stack with a running fleet, and choose **Actions**, **Create streaming URL**.
2. For **UserID**, specify the user ID. Select an expiration time, which determines how long the generated URL is valid.
3. Choose **Get URL**. This displays a window with the URL. To copy the link to your clipboard, choose **Copy Link**.
4. When you are finished viewing and copying the generated URL, choose **Exit**.

Clean Up Resources

You can stop your running fleet and delete your active stack to free up resources and to avoid unintended charges to your account. We recommend stopping any unused, running fleets. For more information, see AppStream 2.0 Pricing.

To clean up your resources

1. In the navigation pane, choose **Stacks** and select the active stack.
2. Choose **Actions**, **Disassociate Fleet**.
3. From **Stack Details**, open the **Associated Fleet** link.
4. The associated fleet is automatically selected in the new window. Choose **Actions**, **Stop**. It usually takes about 5 minutes for a fleet to stop completely. Use the refresh button to update the status.
5. When the fleet has a **Stopped** status, choose **Actions**, **Delete**.

6. In the navigation pane, choose **Stacks** and select the active stack that you chose above.
7. Choose **Actions**, **Delete**.

Next Steps

For more information, see the following topics:

- Learn how to use the AppStream 2.0 image builder to add your own apps and create new images that you can stream. For more information, see Tutorial: Using an AppStream 2.0 Image Builder (p. 11).
- Manage your AppStream 2.0 Home Folders. For more information, see Persistent Storage with AppStream 2.0 Home Folders (p. 19).
- Manage your AppStream 2.0 resources to optimize your streaming performance, automatic scaling, and cost structure. For more information, see Managing Amazon AppStream 2.0 Resources (p. 35).
- Control who has access to your AppStream 2.0 streaming instances. For more information, see Controlling Access to Amazon AppStream 2.0 (p. 49).
- Monitor your AppStream 2.0 resources using Amazon CloudWatch. For more information, see Monitoring Amazon AppStream 2.0 with Amazon CloudWatch (p. 45).
- Troubleshoot your AppStream 2.0 streaming experience. For more information, see Troubleshooting (p. 56).

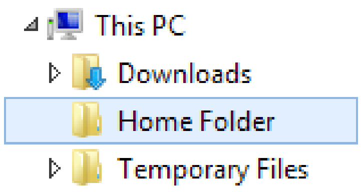
Persistent Storage with AppStream 2.0 Home Folders

AppStream 2.0 offers persistent storage support for your end users with Home Folders. When this option is enabled for an AppStream 2.0 stack, end users of the stack are presented with a persistent storage folder in their AppStream 2.0 sessions with no further configuration required on the end user's part. Data stored by the user in this folder is automatically backed up to an Amazon S3 bucket in your AWS account and is made available in subsequent sessions for that user.

Home Folder End User Experience

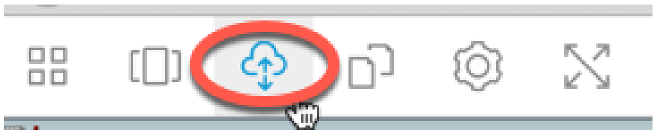
For end users, the Home folder behaves like a normal Windows folder. Users experience the following Home Folder access features within their streaming session.

- Users can save their documents and project files within their Home folder. AppStream 2.0 continuously checks for the most recently modified files and back them up to each user's folder in persistent storage.
- Data content in each user's Home folder is specific to that user, and cannot be accessed by other users.
- Navigation to the Home folder can be accomplished as simply as choosing **This PC**, **Home Folder** from the left navigation menu of Windows Explorer.



- As with Windows Explorer navigation, File Explorer dialogs that are displayed by applications provide seamless access the user's Home folder, for example when the user chooses **File Open** or **File Save** from the application interface.

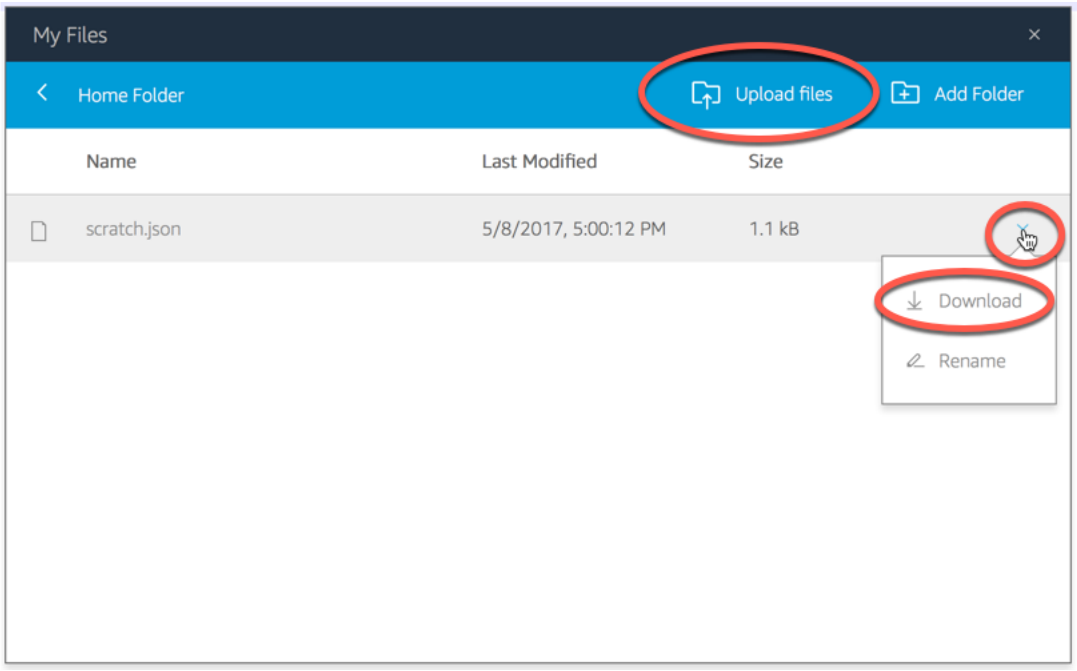
- A web view is available to access the Home folder by choosing **My Files** from the web view session toolbar.



Using the web view, users can navigate to Home folder files, create new folders within their Home folder, upload files from their local computer to their Home folder, download files from their Home folder to their local computer, and rename any of the existing files/folders within their Home folder.

To upload and download files between your local computer and your Home folder

1. In the AppStream 2.0 web view session, choose the **My Files** icon at the top left of your browser.
2. Navigate to an existing folder, or choose **Add Folder** to create a new folder.
3. When the desired folder is displayed, choose **Upload** to upload a local file to the currently displayed location in the session's Home folder. To download a specific file, choose the down arrow on the right of the file name and choose **Download**.



My Files

Home Folder | Upload files | Add Folder

Name	Last Modified	Size
scratch.json	5/8/2017, 5:00:12 PM	1.1 kB

Home Folder Administration

AppStream 2.0 administrators can enable or disable Home Folders for a stack using the AWS Management Console for AppStream 2.0, AWS SDK, or AWS CLI. For each region, Home Folders are backed by an S3 bucket. Choose an administration task from the following list:

Topics

- Amazon S3 Bucket Storage (p. 22)
- Home Folder Formats (p. 23)
- Using the AWS Command Line Interface or AWS SDKs (p. 23)

Before Enabling Home Folders

Refer to the following guidelines before enabling Home Folders for a stack:

- Check that you have the correct IAM permissions for Amazon S3 actions. For more information, see IAM Policies and the Amazon S3 Bucket for Home Folders (p. 52).
- Use an image that was created from an AWS base image released on or after May 18, 2017. For a current list of released AWS images, see Amazon AppStream 2.0 Windows Image Version History (p. 61).
- Enable network connectivity to Amazon S3 from your VPC by configuring Internet access or an VPC endpoint for Amazon S3. For more information, see Network Settings for Fleet and Image Builder Instances (p. 25) and Home Folders and VPC Endpoints (p. 29).

Enabling Home Folders

An AppStream 2.0 administrator can enable or disable Home Folders while creating a stack (see Set Up a Stack (p. 17)), or after the stack is created (see Stacks (p. 36)).

When enabling Home Folders for the first time for an AppStream 2.0 stack in an AWS region, the service creates an S3 bucket in your account in that same region. The same bucket is used to store content of Home folders for all users and all stacks in that region. For more information, see Amazon S3 Bucket Storage (p. 22).

To enable Home Folders while creating a stack

- Follow the instructions at Set Up a Stack (p. 17), and ensure that the **Enable Home Folders** check box is checked.

To enable Home Folders for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. On the left navigation pane, choose **Stacks**, and select the stack for which to enable Home Folders.
3. Below the stack list, choose **Storage** and select the **Enable Home Folders** check box.
4. Confirm the enable action in the resulting dialog box by choosing **Enable**.

Disabling Home Folders

You can disable Home Folders for a stack without losing user content already stored in Home folders. Disabling Home Folders for a stack has the following effects:

- If the stack has sessions that are in use, an error message is shown to any users currently using the session, and these users can no longer store content to Home folders in the session.
- Any new sessions using the stack with Home Folders disabled do not present Home folders.
- Other stacks that have Home Folders enabled are not affected by this stack's disable operation, and continue to provide users access to their data within those stacks' streaming sessions. Only the specific stack for which Home folders is disabled is affected.

- Even if Home folders are disabled for all stacks, AppStream 2.0 does not delete the user content.

To restore access to Home folders for the stack, enable Home folders again as shown in the previous section.

To disable Home folders while creating a stack

- Follow the instructions at Set Up a Stack (p. 17), ensure the **Enable Home Folders** check box is cleared.

To disable Home Folders for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. On the left navigation pane, choose **Stacks**, and select the stack for which to disable Home Folders.
3. Below the stack list, choose **Storage** and clear the **Enable Home Folders** check box.
4. Confirm the disable action in the resulting dialog box by typing CONFIRM (case sensitive) and choosing **Disable**.

Amazon S3 Bucket Storage

AppStream 2.0 manages user content stored in Home folders using S3 buckets created in your account. For every region, AppStream 2.0 creates a bucket in your account and stores all user content generated from streaming sessions of stacks in that region in that bucket. The buckets are fully managed by the service without any admin inputs or configuration. The buckets are named in a specific format as follows:

```
appstream2-36fb080bb8-region-code-account-id-without-hyphens
```

Where *region-code* is the AWS region code in which the stack is created and *account-id-without-hyphens* is your AWS account ID. The first part of the bucket name, `appstream2-36fb080bb8-`, does not change across accounts or regions.

For example, if you enable Home folders for stacks in region us-west-2 on account number 123456789012, the service creates an S3 bucket in the us-west-2 region with the name shown. This bucket name cannot change or be deleted without manual modification by an administrator.

```
appstream2-36fb080bb8-us-west-2-123456789012
```

As previously stated, disabling Home folders for stacks does not delete any user content stored in the S3 bucket. To permanently delete user content, an administrator with adequate access must do so from the Amazon S3 console. AppStream 2.0 adds a bucket policy that prevents accidental deletion of the bucket. For more information, see IAM Policies and the Amazon S3 Bucket for Home Folders (p. 52).

Additional Resources

To learn more about managing S3 buckets and best practices, see the following topics in the *Amazon Simple Storage Service Developer Guide*:

- You can provide offline access to user data for your users with Amazon S3 policies. For more information, see Allow Users to Access a Personal "Home Directory" in Amazon S3.
- You can enable file versioning for content stored in S3 buckets used by AppStream 2.0. For more information, see Using Versioning.

Home Folder Formats

When Home folders are enabled, every user gets a unique folder in which to store their content. The folder is created and maintained as a unique Amazon S3 object within the bucket for that region. The hierarchy of a user folder depends how users launch a streaming session.

AWS SDKs and AWS CLI

For sessions created using `CreateStreamingURL` or `create-streaming-url` the user folder structure is as follows:

```
bucket-name/user/custom/user-id-SHA-256-hash/
```

Where *`bucket-name`* is in the format shown in Amazon S3 Bucket Storage (p. 22) and *`user-id-SHA-256-hash`* is the user-specific folder name created using a lower case SHA-256 hash hexadecimal string generated from the `UserId` value passed to the CreateStreamingURL API operation or `create-streaming-url` command. For more information, see CreateStreamingURL in the *Amazon AppStream 2.0 API Reference* and create-streaming-url in the *AWS Command Line Interface Reference*.

The following example folder structure applies to session access using the API or CLI with a `UserId` testuser@mydomain.com, account id 123456789012 in region us-west-2:

```
appstream2-36fb080bb8-us-west-2-123456789012/user/custom/
a0bcb1da11f480d9b5b3e90f91243143eac04cfccfbdc777e740fab628a1cd13/
```

Administrators can identify the folder for a user by generating the lower case SHA-256 hash value of the `UserId` using websites or open source coding libraries available online.

SAML

For sessions created using SAML federation, the user folder structure is as follows:

```
bucket-name/user/federated/user-id-SHA-256-hash/
```

In this case, *`user-id-SHA-256-hash`* is the folder name created using a lower case SHA-256 hash hexadecimal string generated from the `NameID` SAML attribute value passed in the SAML federation request. To differentiate users with the same name belonging to two different domains, send the SAML request with `NameID` in the format `domainname\username`. For more information, see Enabling Single Sign-on Access to AppStream 2.0 Using SAML 2.0 (p. 30).

The following example folder structure applies to session access using SAML federation with a `NameID` SAMPLEDOMAIN\testuser, account ID 123456789012 in region us-west-2:

```
appstream2-36fb080bb8-us-west-2-123456789012/user/
federated/8dd9a642f511609454d344d53cb861a71190e44fed2b8af9fde0c507012a9901
```

Administrators can identify the folder for a user by generating the lower case SHA-256 hash value of the `NameID` using websites or open source coding libraries available online.

Using the AWS Command Line Interface or AWS SDKs

You can enable and disable Home Folders for a stack using the AWS CLI or AWS SDKs.

The following CLI command can be used to enable Home Folders while creating a new stack named *`ExampleStack`*:

```
aws appstream create-stack --name ExampleStack --storage-connectors type=HOMEFOLDERS
```

The following CLI command can be used to enable Home Folders for an existing stack named *ExistingStack*:

```
aws appstream update-stack --name ExistingStack --storage-connectors type=HOMEFOLDERS
```

The following CLI command can be used to disable Home Folders for an existing stack named *ExistingStack*. This command does not delete any user data.

```
aws appstream update-stack -name ExistingStack --delete-storage-connectors
```

For more information, see AWS Command Line Interfaceand Amazon AppStream 2.0 Command Line Reference. For more information about the AWS SDKs, see Tools for Amazon Web Services. For more information about the AppStream 2.0 API, see Amazon AppStream 2.0 API Reference.

Network Settings for Fleet and Image Builder Instances

The following sections contain information about configuring your AppStream 2.0 fleets and image builders to access network resources and the Internet.

When creating an AppStream 2.0 fleet or image builder, you can provide Amazon VPC subnets. AppStream 2.0 sets up elastic network interfaces (ENI) to the subnets provided so that AppStream 2.0 instances have access to your network resources or have access to public Internet through your VPC. For more information about Amazon VPC, see VPC and Subnet Basics.

Contents

Network Setup Guidelines

There are some network setup guidelines to consider for fleets and image builders. If your fleets and image builders require Internet access, you can use either the Default Internet Access feature, or manually control Internet access using an advanced networking configuration such as a VPC with NAT gateways. For more information, see Enabling Internet Access Using a Public Subnet (p. 25) and Enabling Internet Access Using a NAT Gateway (p. 25).

Fleets

You can provide subnets to establish network connections from your fleet instances to your VPC. We recommend that you specify two private subnets from different Availability Zones for high availability and fault tolerance. Also, ensure that the network resources for your applications are accessible through both of the specified private subnets.

AppStream 2.0 creates as many elastic network interfaces as the maximum desired capacity of your fleet. The following guidelines will help you set up a VPC to support scaling behavior for your fleet.

- Make sure that your AWS account has sufficient elastic network interface capacity to support the scaling requirements of your fleet. If you are planning to launch a large fleet of streaming instances, contact AWS Support and request a higher ENI limit to match the maximum number of instances that you plan to launch.
- Specify subnets with a sufficient number of elastic IP addresses to match the maximum desired capacity of your fleet.

Image Builders

You can choose one subnet while launching an image builder. Ensure the subnet accessibility of the network resources,with which your applications may interact. The typical resources required for successful execution of your apps may include licensing servers, database servers, file servers, and so on.

Enabling Internet Access Using a Public Subnet

AppStream 2.0 can provide your fleets with a default Internet connection using your Amazon VPC public subnet that has a route to the Internet through an Internet gateway.

AppStream 2.0 enables Internet connectivity by associating an Elastic IP address to the network interface that is attached from the streaming instance to your VPC public subnet. You can have a VPC with a public subnet in several ways:

Default VPC

Your AWS account, if it was created after 2013-12-04, has a default VPC that has public subnets. You can use this default VPC to enable Internet access from your streaming instances. For more information, see Default VPC and Default Subnets in the *Amazon VPC User Guide*.

New VPC

If your AWS account was created before 2013-12-04 or to manage a new VPC, you can create a new VPC with a public subnet using the VPC creation wizard. For more information, see Implementation of VPC with a single public subnet in the *Amazon VPC User Guide*.

Existing VPC

To use an existing VPC that does not have a public subnet, you can add a new public subnet using the following steps.

To add a new public subnet to an existing VPC

1. Follow the steps given in the Creating a Subnet section in the *Amazon VPC User Guide*, using the existing VPC you intend to use with AppStream 2.0.
2. To add an Internet gateway to your VPC, follow the steps in the Attaching an Internet Gateway section in the *Amazon VPC User Guide*.

3. To configure your subnets to route Internet Traffic through the Internet Gateway ,follow the steps in the Creating a Custom Route Table section in the *Amazon VPC User Guide*, specify IPv4 format (`0.0.0.0/0`) for **Destination**.

Enabling Internet Access for a Fleet

After you have a public subnet available on a VPC, you can enable Internet access for your fleet. This can be performed either when you create it, or by editing the fleet details after creation.

To enable Internet access at fleet creation

1. Follow the instructions at Set Up a Fleet (p. 15) up to the **Network access** section.
2. Choose **Default Internet Access**.
3. If the subnet fields are empty, select a subnet for **Subnet 1** and, if desired, **Subnet 2**.
4. Continue with the instructions at Set Up a Fleet (p. 15).

To enable Internet access after fleet creation

1. In the navigation pane, choose **Fleets**.
2. Select a fleet and check that the state is **Stopped**.
3. Choose **Fleet Details**, **Edit**, **Default Internet Access**.
4. Choose a subnet for **Subnet 1** and, if desired, **Subnet 2**. Choose **Update**.

You can test Internet connectivity by starting your fleet, creating a stack, associating the fleet to a stack, and browsing Internet within a streaming session for stack. For more information, see Set Up AppStream 2.0 Stacks and Fleets (p. 15).

Enabling Internet Access for an Image Builder

After you have a public subnet available on a VPC, and can enable Internet access for your image builder. This can be performed when you create the image builder.

To enable Internet access for an image builder

1. Follow the instructions at Step 1: Create an Image Builder (p. 11) up to the **Network Access** section.
2. Choose **Default Internet Access**.
3. If **Subnet** is empty, select a subnet.
4. Continue with the instructions at Step 1: Create an Image Builder (p. 11).

Enabling Internet Access Using a NAT Gateway

You can control Internet access for your users using an advanced networking configuration such as NAT gateways. To manage your own VPC and VPC NAT gateway, launch your AppStream 2.0 image builders and fleets in private VPC subnets that provide a route to the Internet. Use the instructions below to quickly create a network setup for enabling Internet access. For more information, see NAT Gateways and VPC with Public and Private Subnets (NAT) in the *Amazon VPC User Guide*.

To create and configure a new VPC to use with a VPC NAT gateway

1. Navigate to Implementing VPC with Public and Private Subnets (NAT) in the *Amazon VPC User Guide*, and follow the steps given in the section **To create a VPC**, leaving out the optional IPv6 step.

2. For **Availability Zone**, leave the public subnet zone as the default, and select a specific zone for the private subnet. Make a note of the zones you chose.
3. For **Elastic IP Allocation ID**, choose an existing Elastic IP address. If you don't have one, create an Elastic IP address from the **Elastic IPs** section on the Amazon VPC console.
4. Leave the other fields as their default values, making a note of the value for **Private subnet's IPv4 CIDR**, and then choose **Create VPC**. This may take some time to complete.
5. If you want to add another private subnet to your VPC, perform the following steps.

    a. In the left navigation pane, choose **Subnets**, **Create Subnet**. Be sure to choose a different name than the ones specified in step 3.
    b. For **VPC**, enter the VPC that you created earlier. For **Availability Zone**, enter a different value than the one noted earlier.
    c. For **IPv4 CIDR block**, provide a unique for the new subnet. For example, if you noted the first subnet has a IPv4 CIDR block range of `10.0.1.0/24`, the new subnet could have a valid CIDR block range of `10.0.2.0/24`.
6. Choose **Yes, Create**.

To add a NAT gateway to an existing VPC

1. Follow the instructions in Creating a NAT Gateway in the *Amazon VPC User Guide* to add a new NAT gateway to the VPC to use with AppStream 2.0.
2. Follow the instructions in Updating Your Route Table in the *Amazon VPC User Guide* to update the route tables of your private subnets and route Internet traffic through the NAT gateway.
3. Check your VPC to be sure it has at least one private subnet and, if needed, create a new private subnet. For more information, see Creating a Subnet in the *Amazon VPC User Guide*.

Enabling Internet Access for a Fleet Using a NAT Gateway

After you have a NAT gateway available on a VPC, you can enable Internet access for your fleet. This can be performed either when you create it, or by editing the fleet details after creation.

To enable Internet access at fleet creation using a NAT gateway

1. Follow the instructions at Set Up a Fleet (p. 15) up to the **Network access** section.
2. Choose a VPC with a NAT gateway.
3. If the subnet fields are empty, select a private subnet for **Subnet 1** and, if desired, another private subnet for **Subnet 2**. You may need to create a second private subnet if one is not already present for your VPC.
4. Continue with the instructions at Set Up a Fleet (p. 15).

To enable Internet access after fleet creation using a NAT gateway

1. In the navigation pane, choose **Fleets**.
2. Select a fleet and check that the state is **Stopped**.
3. Choose **Fleet Details**, **Edit**, and choose a VPC with a NAT gateway.
4. Choose a private subnet for **Subnet 1** and, if desired, another private **Subnet 2**. You may need to create a second private subnet if one is not already present for your VPC.
5. Choose **Update**.

You can test your Internet connectivity by starting your fleet, and then connecting to your streaming instance and browsing to the Internet.

Enabling Internet Access for an Image Builder Using a NAT Gateway

After you have a NAT gateway available on a VPC, and can enable Internet access for your image builder. This can be performed when you create the image builder.

To enable Internet access for an image builder using a NAT gateway

1. Follow the instructions at Step 1: Create an Image Builder (p. 11) up to the **Network Access** section.
2. Choose the VPC with a NAT gateway.
3. If **Subnet** is empty, select a subnet.
4. Continue with the instructions at Step 1: Create an Image Builder (p. 11).

Home Folders and VPC Endpoints

To support Home Folders on a private network, AppStream 2.0 needs access permissions to the VPC endpoint. To enable AppStream 2.0 access to your private Amazon S3 endpoint, attach a custom policy, as defined below, to your VPC endpoint for Amazon S3. For more information about private Amazon S3 endpoints, see VPC Endpoints and Endpoints for Amazon S3 in the *Amazon VPC User Guide*.

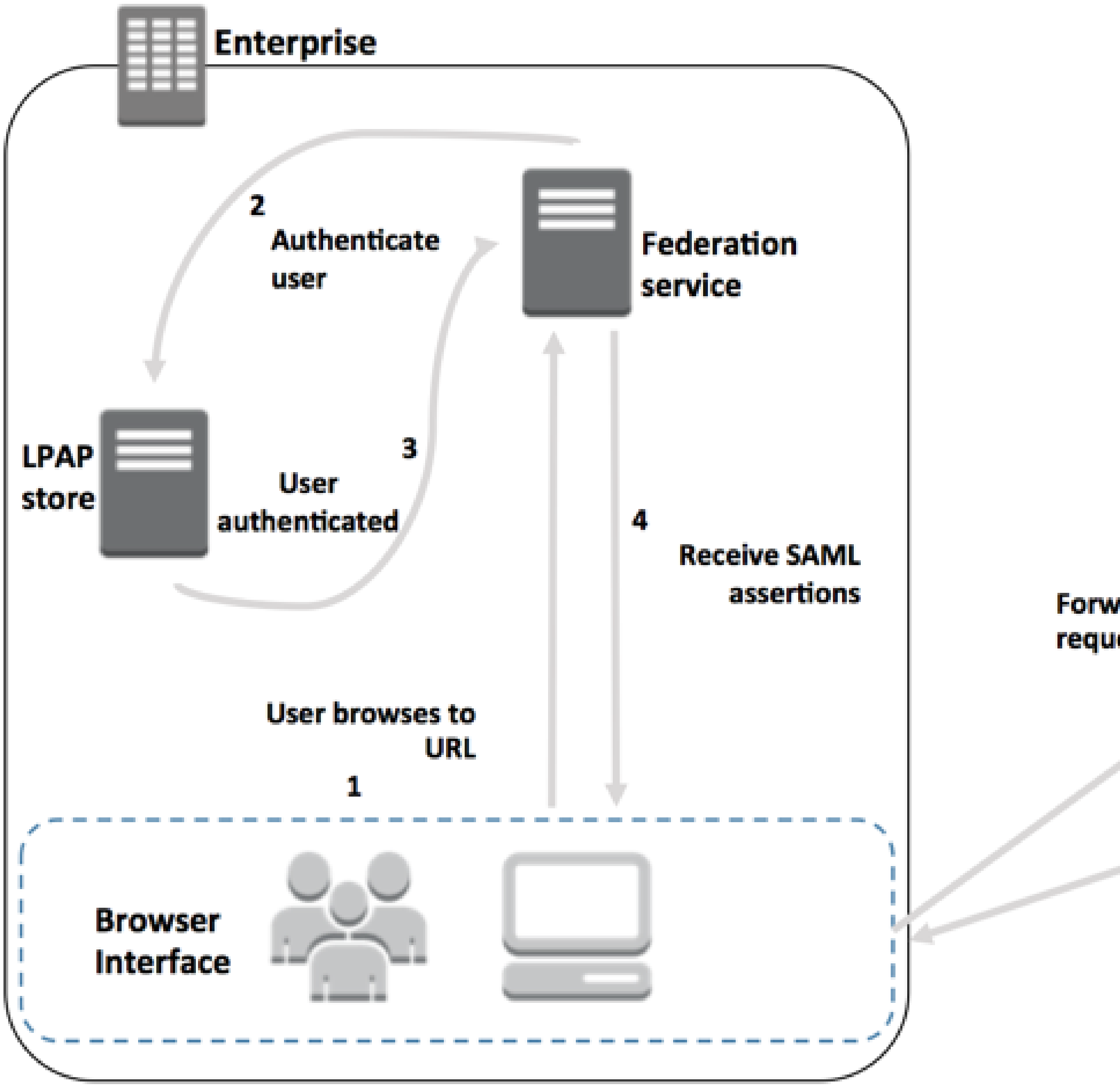
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow-AppStream-to-access-specific-bucket",
      "Effect": "Allow",
      "Principal": {
        "Service": "appstream.amazonaws.com"
      },
      "Action": [
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:GetObjectVersion",
        "s3:DeleteObjectVersion",
        "s3:PutBucketPolicy"
      ],
      "Resource": "arn:aws:s3:::appstream2-36fb080bb8-*"
    }
  ]
}
```

Enabling Single Sign-on Access to AppStream 2.0 Using SAML 2.0

Amazon AppStream 2.0 supports identity federation to AppStream 2.0 stacks through Security Assertion Markup Language 2.0 (SAML 2.0). You can use an identity provider that supports SAML 2.0—such as Active Directory Federation Service, Ping One Federation Server, or Okta—to provide a simple onboarding flow for your AppStream 2.0 users. This feature offers your users the convenience of one-click access to their AppStream 2.0 applications using their existing identity credentials. You also have the security benefit of identity authentication by your identity provider. You can control which users have access to a particular AppStream 2.0 stack, using your existing identity provider.

Example Authentication Workflow

The following diagram illustrates the authentication flow between AppStream 2.0 and a third-party identity provider. In this example, the administrator has set up a sign-in page to access AppStream 2.0, called `applications.exampleco.com`. The web page uses a SAML 2.0 compliant federation service to trigger a sign-on request. The administrator has also set up a user to allow access to AppStream 2.0.



1. The user browses to `https://applications.exampleco.com`. The sign-on page requests authentication for the user.
2. The federation service requests authentication from the organization's identity store.
3. The identity store authenticates the user and returns the authentication response to the federation service.
4. On successful authentication, the federation service posts the SAML assertion to the user's browser.
5. The user's browser posts the SAML assertion to the AWS Sign-In SAML endpoint (`https://signin.aws.amazon.com/saml`). AWS Sign-In receives the SAML request, processes the request, authenticates the user, and forwards the authentication token to the AppStream 2.0 service.
6. Using the authentication token from AWS, AppStream 2.0 authorizes the user and presents applications to the browser.

From the user's perspective, the process happens transparently: The user starts at your organization's internal portal and lands at an AppStream 2.0 application portal, without ever having to supply any AWS credentials.

Setting Up SAML

You can use an IAM role and a relay state URL to configure your SAML 2.0-compliant IdP and enable AWS to permit your federated users to access an AppStream 2.0 stack. The role grants the user permissions to access the stack. The relay state is the stack portal to which the user is forwarded after successful authentication by AWS.

Topics

Prerequisites

Here is a summary of the prerequisite steps required for configuring your SAML 2.0 connection.

- Configure your IdP to establish a trust relationship with AWS.
  - Inside your organization's network, configure your identity store, such as Windows Active Directory, to work with a SAML-based identity provider (IdP), such as Microsoft Windows Active Directory Federation Services, Shibboleth, and so on.
  - Using your IdP, generate a metadata document that describes your organization as an identity provider.
  - For more information, see AppStream 2.0 Integration with SAML 2.0 (p. 34).
- Create an AppStream 2.0 stack and note the name of the stack for use in IAM policy and IdP configuration.
  - You can create an AppStream 2.0 stack using the AppStream 2.0 management console, AWS CLI, or AppStream 2.0 API.

Step 1: Create a SAML Provider in AWS

This identity provider defines your organization's IdP to AWS using the metadata document you previously generated using your IdP. Here are the steps to create a SAML provider in AWS:

- Sign in to the AWS Identity and Access Management (IAM) console.
- Create a new SAML provider, which is an entity in IAM that holds information about your organization's identity provider.
- As part of this process, upload the metadata document produced by the IdP software in your organization noted in the previous section. For more information, see Creating SAML Identity Providers in the *IAM User Guide*.

Step 2: Configure Permissions in AWS for Your Federated Users

The next step is to create an IAM role that establishes a trust relationship between IAM and your organization's IdP that identities your IdP as a principal (trusted entity) for the purposes of federation. The role also defines which users authenticated by your organization's IdP are allowed to access an AppStream 2.0 stack. For more information about creating a role for a SAML IdP, see Creating a Role for SAML 2.0 Federation in the *IAM User Guide*.

After you have created the role, you can limit the role to have permissions only to one or more AppStream 2.0 stacks by attaching an inline policy to the role. The following sample policy document provides access to a single AppStream 2.0 stack:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "appstream:Stream",
      "Resource": "arn:aws:appstream:REGION-CODE:ACCOUNT-ID-WITHOUT-HYPHENS:stack/STACK-
NAME",
      "Condition": {
          "StringEquals": {
              "appstream:userId": "${saml:sub}",
              "saml:sub_type": "persistent"
          }
        }
    }
  ]
}
```

Choose a value for *REGION-CODE* that corresponds to the region where your AppStream 2.0 stack exists, and replace *STACK-NAME* with the name of the stack. You can view stack details in the **Stacks** dashboard of the AppStream 2.0 management console.

Step 3: Configure the SAML IdP

After you create the role, update your SAML IdP about AWS as a service provider by installing the `saml-metadata.xml` file found at https://signin.aws.amazon.com/static/saml-metadata.xml. Review instructions provided by your IdP for updating the metadata. Some providers give you the option to type the URL, whereupon the IdP gets and installs the file for you. Others require you to download the file from the URL and then provide it as a local file. For more information, see your IdP documentation or AppStream 2.0 Integration with SAML 2.0 (p. 34).

Step 4: Create Assertions for the SAML Authentication Response

Next, configure the information that the IdP passes as SAML attributes to AWS as part of the authentication response. For more information, see Configuring SAML Assertions for the Authentication Response in the *IAM User Guide*.

Step 5: Configure the Relay State of Your Federation

You also need to configure the relay state of your federation to point to the AppStream 2.0 stack relay state URL. After successful authentication by AWS, the user is directed to the AppStream 2.0 stack portal, defined as the relay state in the SAML authentication response.

The format of the relay state URL is as follows:

```
https://relay-state-region-endoint?stack=stackname&accountId=aws-account-id-without-hyphens
```

Construct your relay state URL from your AWS account ID, stack name, and the relay state endpoint associated with the region in which your stack is located.

Region	Relay state endpoint
us-east-1 (N.Virginia)	`https://appstream2.us-east-1.aws.amazon.com/saml`
us-west-2 (Oregon)	`https://appstream2.us-west-2.aws.amazon.com/saml`
eu-west-1 (Ireland)	`https://appstream2.eu-west-1.aws.amazon.com/saml`
ap-northeast-1 (Tokyo)	`https://appstream2.ap-northeast-1.aws.amazon.com/saml`

AppStream 2.0 Integration with SAML 2.0

For more information about additional supported SAML providers, see Integrating Third-Party SAML Solution Providers with AWS in the *IAM User Guide*.

The following links help you configure third-party SAML 2.0 identity provider solutions to work with AppStream 2.0.

Identity provider solution	More information
Ping Identity	Configuring an SSO connection to Amazon AppStream 2.0 — This page on the Ping Identity website describes how to set up single sign- on (SSO) to AppStream 2.0
Okta	How to Configure SAML 2.0 for Amazon AppStream 2.0 — This article on the Okta site explains how to use Okta to set up SAML federation to AppStream 2.0.
Microsoft Active Directory Federation Services (ADFS)	Enabling Federation to AWS Using Windows Active Directory, ADFS, and SAML 2.0 — This post on the AWS Security Blog shows how to set up ADFS on an EC2 instance and enable SAML federation to a specific console, using the `RelayState` parameter. You can follow this tutorial and replace the relay state in the example with the relay state of the AppStream 2.0 stack.
Shibboleth	How to Use Shibboleth for Single Sign-On to the AWS Management Console — This AWS Security Blog post talks about setting up federation to the AWS Management Console using AD and Shibboleth. After you have created the setup to federate to the AWS Management Console as outlined in the tutorial, you can edit the relay state provided in the tutorial with the relay state of your AppStream 2.0 stack.

Managing Amazon AppStream 2.0 Resources

You can use the Amazon AppStream 2.0 console or API to manage stacks, fleets and images.

Topics

Image Builders

AppStream 2.0 provides virtual machines, or instances, that are used to install and add applications into and create your image. These instances are called *image builders*. You can launch an image builder from a base image provided by AWS, or from an image that you create. After your image builder instance is available (running), you can connect to the image builder to start a desktop session, install your applications, add your applications to an image, and create an image.

While launching a new image builder, you can choose from different instance types with various compute, memory, and graphics configurations. You also provide a VPC subnet so that AppStream 2.0 can establish a network interface to the image builder. This connection provides your image builder with access to resources that might be needed while you install and add applications; for example, file servers, licensing servers, database servers, and so on. For more information, see Tutorial: Using an AppStream 2.0 Image Builder (p. 11).

Actions

The following actions can be performed on an image builder, depending on the current state (status) of the image builder instance.

Delete

Permanently delete an image builder.

The instance must be in a **Stopped** state.

Connect

Connect to a running image builder. This action starts a desktop streaming session with the image builder to install and add applications to the image, and create an image.

The instance must be in a **Running** state.

Start

Start a stopped image builder. A running instance is billed to your account.

The instance must be in a **Stopped** state.

Stop

Stop a running image builder. A stopped instance is not billed to your account.

The instance must be in a **Running** state.

None of these actions can be performed on an instance in any of the following intermediate states:

- **Pending**
- **Snapshotting**
- **Stopping**
- **Starting**
- **Deleting**

Images

An Amazon AppStream 2.0 image contains applications that can be streamed to users. The image is used to launch streaming instances that are part of an AppStream 2.0 fleet. All images available to you are listed under the **Image Registry** section in the AWS Management Console. There are two types of images listed in your image registry that are differentiated by these visibility attributes:

- **Public Images** — Base images that are made available by AWS to help you create images with your own applications.
- **Private Images** — Images that are created and owned by you.

You can use either public or private images to launch an image builder and set up your AppStream 2.0 fleet. For more information, see Tutorial: Using an AppStream 2.0 Image Builder (p. 11).

You can also delete your private images. Note that a private image cannot be deleted if there are active fleets using it. You must stop all associated fleets before deleting the image.

Stacks and Fleets

Amazon AppStream 2.0 uses stacks and fleets to stream instances, and manage user access policies and storage configurations. You create stacks and fleets as part of the streaming configuration process. For more information, see Getting Started with Amazon AppStream 2.0 (p. 8) and Set Up AppStream 2.0 Stacks and Fleets (p. 15).

Stacks

An Amazon AppStream 2.0 stack consists of user access policies that control access to the fleet associated with the stack. You can create a new stack by choosing **Stacks**, **Create Stack**.

The following actions can be performed on a stack in the AWS console.

Create streaming URL

Create a URL for a user to stream applications. This action can only be performed if a running fleet is associated with the stack.

Associate fleet

Associate an existing fleet with the stack. The stack needs to have an associated fleet in order to stream user applications. This action cannot be performed on a stack that already has an associated fleet. First, disassociate the associated fleet and then associate the desired fleet with the new stack.

Disassociate fleet

Disassociate the associated fleet from the stack.

Delete

Delete the stack. This action cannot be performed on a stack that has an associated fleet. First, disassociate the fleet from the stack and then delete the stack. Note that you must stop the disassociated fleet to avoid being charged for resources running as a part of that fleet.

In addition to the **Actions** button, you can select the desired tab below the stack list to view Stack Details, or Storage to enable/disable Home Folders for the stack. For more information, see Persistent Storage with AppStream 2.0 Home Folders (p. 19).

Fleets

An AppStream 2.0 fleet is a group of streaming instances from which user applications are executed and streamed. You can create a new fleet by choosing **Fleets**, **Create Fleet**.

The following actions can be performed on a fleet in the AWS console.

Delete

Delete all resources in a fleet. This action cannot be performed on a running, stopping, or starting fleet.

Edit

Edit the fleet parameters. The parameters available for editing depends on the current status of the fleet.

Fleet Status	Editable Fields
STARTING	None
RUNNING	Display Name, Desired Capacity, Image, Disconnect Timeout
STOPPING	None
STOPPED	Display Name, Desired Capacity, VPC, Subnets, Image, Disconnect Timeout, Instance Type

Start

Start the desired number of instances in the fleet.

Stop

Stop all instances in the fleet.

Passing Command Line Parameters to Applications Using Session Context

You can pass command line parameters dynamically by launching your streaming session with the AWS CLI. This parameter is passed via session context to the streaming URL.

To pass a parameter to a batch script

1. Connect to your image builder in ImageBuilderAdmin mode. For this example, install the Google Chrome application on the image builder.
2. Create a new folder under `C:\`. For this example, name the folder "Scripts".
3. Create a batch script under the new folder. In this example, use a script that launches Chrome and then waits for keyboard input before closing the script window. Chrome launches with a URL passed as a command line parameter to the batch script.

   Example `session-context-test.bat` script:

   ```
   chrome.exe %1
   pause
   ```
4. In Image Assistant, add `session-context-test.bat` and change the working directory parameter to `C:\Program Files (x86)\Google\Chrome\Application`.
5. Create an image, fleet, and stack. For this example, the fleet and stack names are `session-context-test-fleet` and `session-context-test-stack`, respectively.
6. After the fleet is running, you can create a streaming URL using the AWS CLI `session-context` parameter. You can modify the example below to match your specific requirements.

   ```
   aws appstream create-streaming-url -fleet-name session-context-test-fleet -stack-
   name session-context-test-stack -user-id username -validity 10000 -application-
   id chrome -session-context "www.google.com"
   ```

If everything is set up correctly, browsing to the streaming URL results in the batch script launching Chrome and loading `http://www.google.com`.

Fleet Auto Scaling for Amazon AppStream 2.0

Fleet Auto Scaling allows you to automatically change the size of your AppStream 2.0 fleet to match the supply of running instances to user demand. Because each running instance in a fleet can be used by only one user at a time, the size of your fleet determines the number of users who can stream concurrently. You can define scaling policies that adjust the size of your fleet automatically based on a variety of utilization metrics, and optimize the number of running instances to match user demand. You can also choose to turn off automatic scaling and make the fleet run at a fixed size.

AppStream 2.0 scaling is provided by Application Auto Scaling. For more information, see the Application Auto Scaling User Guide.

Before you can use Fleet Auto Scaling, Application Auto Scaling needs permissions to access Amazon CloudWatch alarms and AppStream 2.0 fleets. For more information, see Application Auto Scaling IAM Role (p. 51) and Application Auto Scaling Required IAM Permissions (p. 52).

For a walk-through of AppStream 2.0 scaling, see Scaling Your Desktop Application Streams with Amazon AppStream 2.0 in the *AWS Compute Blog*.

Scaling Concepts

To use Application Auto Scaling effectively, there are a few terms and concepts that you should be familiar with and understand.

Minimum Capacity

> The minimum size of the fleet. Scaling policies do not scale your fleet below this value. For example, if you specify 2, your fleet will never have less than 2 instances running. Note that if **Desired Capacity** (set by editing **Fleet Details** and not **Scaling Policies**) is set below the value of **Minimum Capacity** and a scale-out activity is triggered, Application Auto Scaling scales the **Desired Capacity** value up to the value of **Minimum Capacity** and then continues to scale out as required, based on the scaling policy. However, in this example, a scale-in activity does not adjust **Desired Capacity**, because it is already below the **Minimum Capacity** value.

Maximum Capacity

> The maximum size of the fleet. Scaling policies do not scale your fleet above this value. For example, if you specify 10, your fleet will never have more than 10 instances running. Note that if **Desired Capacity** (set by editing **Fleet Details** and not **Scaling Policies**) is set above the value of **Maximum Capacity** and a scale-in activity is triggered, Application Auto Scaling scales **Desired Capacity** down to the value of **Maximum Capacity** and then continues to scale in as required, based on the scaling policy. However, in this example, a scale-out activity does not adjust **Desired Capacity**, because it is already above the **Maximum Capacity** value.

Scaling Policy Action

> The action that scaling policies perform on your fleet when the **Scaling Policy Condition** is met. You can choose an action based on **% capacity** or **number of instance(s)**. For example, if **Desired Capacity** is 4 and **Scaling Policy Action** is set to "Add 25% capacity", **Desired Capacity** is increased by 25% to 5 when **Scaling Policy Condition** is met.

Scaling Policy Condition

> The condition that triggers the action set in **Scaling Policy Action**. This condition includes a scaling policy metric, a comparison operator, and a threshold. For example, to scale a fleet if the utilization of the fleet is greater than 50%, your scaling policy condition should be "If Capacity Utilization > 50%".

Scaling Policy Metric

> This is the metric on which your scaling policy is based. The following metrics are available for scaling policies:
>
> **Capacity Utilization**
>
> > Percentage of instances in a fleet that are being used. You can use this metric to scale your fleet based on usage of the fleet. For example, **Scaling Policy Condition**: "If Capacity Utilization < 25%" perform **Scaling Policy Action**: "Remove 25 % capacity".
>
> **Available Capacity**
>
> > Number of instances in your fleet that are available for user sessions. You can use this metric to maintain a buffer in your capacity available for users to start streaming sessions. For example, **Scaling Policy Condition**: "If Available Capacity < 5" perform **Scaling Policy Action**: "Add 5 instance(s)".
>
> **Insufficient Capacity Error**
>
> > Number of session requests rejected due to lack of capacity. You can use this metric to provision new instances for users that are unable to get sessions because of lack of capacity. For example, **Scaling Policy Condition**: "If Insufficient Capacity Error > 0" perform **Scaling Policy Action**: "Add 1 instance(s)".

Managing Fleet Scaling Using the Console

You can set up and manage fleet scaling using the AWS Management Console in two ways: during fleet creation, or anytime using the **Fleets** tab. Two default scaling policies are associated with newly created fleets after launch and can be edited via the console from the **Scaling Policies** tab. For more information, see Set Up a Fleet (p. 15).

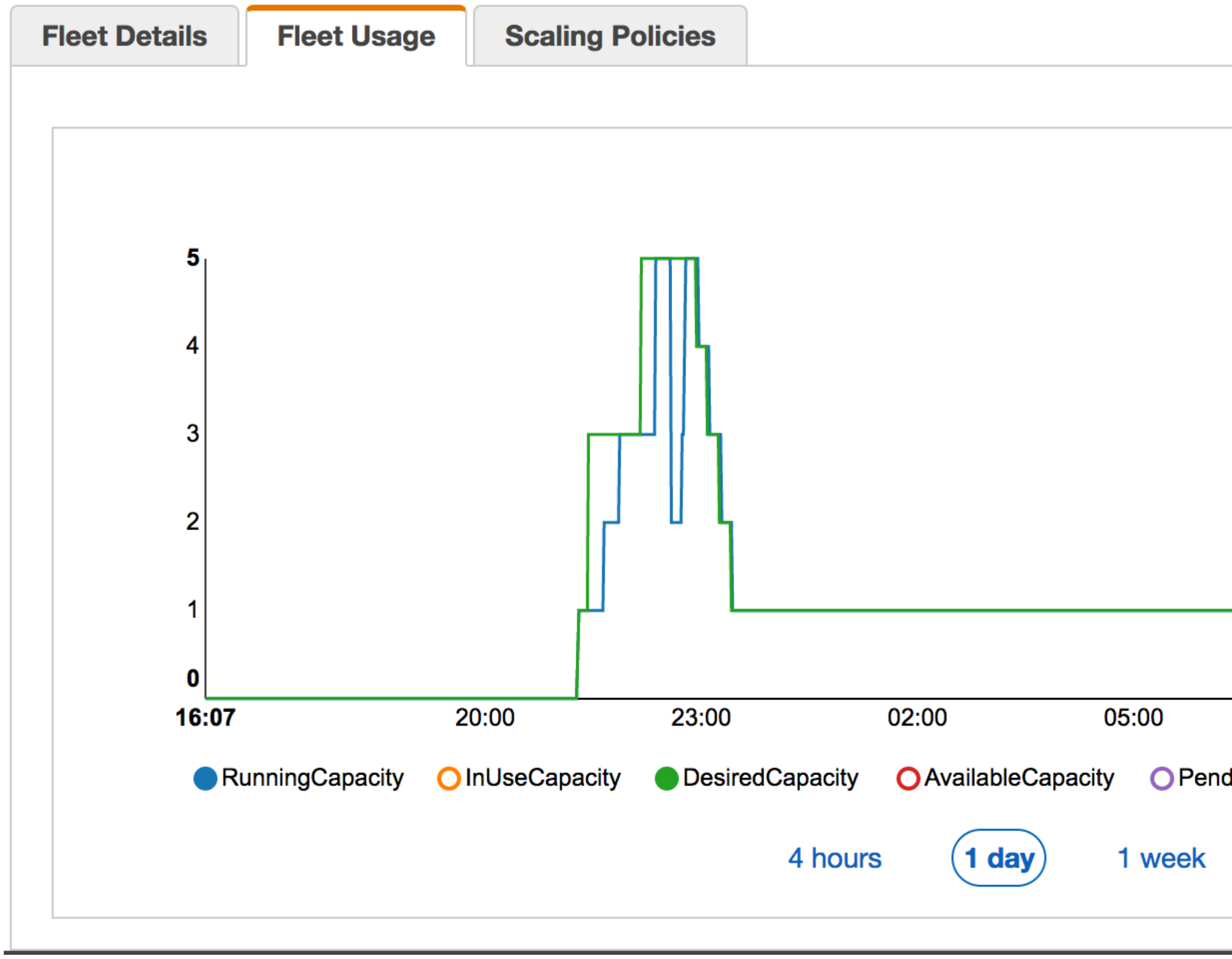
For user environments that vary in number, define scaling policies to control how scaling responds to demand. If you expect a fixed number of users or have other reasons for disabling scaling, you can set your fleet with a fixed number of instances.

To set a fleet scaling policy using the console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Fleets**.
3. Select the fleet to set scaling policies and then choose **Scaling Policies**.
4. Edit existing policies by choosing the edit icon next to each value. Set the desired values in the edit field and choose **Update**. The policy changes go into effect within a few minutes.
5. Add (create) new policies using the **+Add Policy** link. Set the desired values in the edit field and choose **Create**. The new policy goes into effect within a few minutes.

You can use the **Fleet Usage** tab to monitor the effects of your scaling policy changes. The following is an example usage graph of scaling activity when five users connect to the fleet and then disconnect. This example is from a fleet using the following scaling policy values:

- Minimum Capacity = 1
- Maximum Capacity = 5
- Scale Out = Add 2 instances if Capacity Utilization > 75%
- Scale In = Remove 1 instance if Capacity Utilization < 25%



To set a fixed capacity fleet using the console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Fleets**.
3. Select the fleet to set to fixed capacity.
4. In the **Scaling Policies** section, remove all policies associated with the fleet.
5. In the **Fleet Details** section, edit the fleet to set the **Desired Capacity**.

The fixed fleet has constant capacity based on the value that you specified as **Desired Capacity**. Note that a fixed fleet has the desired number of instances running at all times and the fleet must be stopped to stop billing costs for that fleet.

Managing Fleet Scaling Using the AWS CLI

You can set up and manage fleet scaling using the AWS Command Line Interface.

CLI Examples

Before running scaling policy commands, you must register your fleet as a scalable target. Use the following command:

```
aws application-autoscaling register-scalable-target
    --service-namespace appstream
    --resource-id fleet/fleetname
    --scalable-dimension appstream:fleet:DesiredCapacity
    --min-capacity 1
    --max-capacity 5
    --role-arn arn:aws:iam::account-number-without-hyphens:role/service-role/
ApplicationAutoScalingForAmazonAppStreamAccess
```

Example 1: Applying a Scaling Policy Based on Capacity Utilization

This CLI example sets up a scaling policy that scales out a fleet by 25% if Utilization >= 75%.

The following command defines a utilization-based scaling policy:

```
aws application-autoscaling put-scaling-policy --cli-input-json file://scale-out-
utilization.json
```

The contents of the file `scale-out-utilization.json` are as follows:

```
{
    "PolicyName": "policyname",
    "ServiceNamespace": "appstream",
    "ResourceId": "fleet/fleetname",
    "ScalableDimension": "appstream:fleet:DesiredCapacity",
    "PolicyType": "StepScaling",
    "StepScalingPolicyConfiguration": {
        "AdjustmentType": "PercentChangeInCapacity",
        "StepAdjustments": [
            {
                "MetricIntervalLowerBound": 0,
                "ScalingAdjustment": 25
            }
        ],
        "Cooldown": 1500
    }
}
```

If the command is successful, the output looks something like the following, although some details are unique to your account and region. In this example, the policy identifier is `e3425d21-16f0-d701-89fb-12f98dac64af`.

```
{"PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:e3425d21-16f0-
d701-89fb-12f98dac64af:resource/appstream/fleet/SampleFleetName:policyName/
SamplePolicyName"}
```

Now, set up a CloudWatch alarm for this policy. Use the names, region, account number, and policy identifier from your information. You can use the policy ARN returned by the previous command for the `--alarm-actions` parameter.

```
aws cloudwatch put-metric-alarm
--alarm-name alarmname
--alarm-description "Alarm when Capacity Utilization exceeds 75 percent"
```

```
--metric-name CapacityUtilization
--namespace AWS/AppStream
--statistic Average
--period 300
--threshold 75
--comparison-operator GreaterThanThreshold
--dimensions "Name=FleetName,Value=fleetname"
--evaluation-periods 1
--alarm-actions "arn:aws:autoscaling:your-region-code:account-number-without-
hyphens:scalingPolicy:policyid:resource/appstream/fleet/fleetname:policyName/policyname"
--unit Percent
```

Example 2: Applying a Scaling Policy Based on Insufficient Capacity Errors

This CLI example sets up a scaling policy that scales out the fleet by 1 if the fleet throws an `InsufficientCapacityError` error.

The following command defines a insufficient capacity–based scaling policy:

```
aws application-autoscaling put-scaling-policy --cli-input-json file://scale-out-
capacity.json
```

The contents of the file `scale-out-capacity.json` are as follows:

```
{
    "PolicyName": "policyname",
    "ServiceNamespace": "appstream",
    "ResourceId": "fleet/fleetname",
    "ScalableDimension": "appstream:fleet:DesiredCapacity",
    "PolicyType": "StepScaling",
    "StepScalingPolicyConfiguration": {
        "AdjustmentType": "ChangeInCapacity",
        "StepAdjustments": [
            {
                "MetricIntervalLowerBound": 0,
                "ScalingAdjustment": 1
            }
        ],
        "Cooldown": 1500
    }
}
```

If the command is successful, the output looks something like the following, although some details are unique to your account and region. In this example, the policy identifier is `f4495f21-0650-470c-88e6-0f393adb64fc`.

```
{"PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:f4495f21-0650-470c-88e6-0f393adb64fc:resource/appstream/
fleet/SampleFleetName:policyName/SamplePolicyName"}
```

Now, set up a CloudWatch alarm for this policy. Use the names, region, account number, and policy identifier from your information. You can use the policy ARN returned by the previous command for the `--alarm-actions` parameter.

```
aws cloudwatch put-metric-alarm
--alarm-name alarmname
--alarm-description "Alarm when out of capacity is > 0"
--metric-name InsufficientCapacityError
--namespace AWS/AppStream
```

```
--statistic Maximum
--period 300
--threshold 0
--comparison-operator GreaterThanThreshold
--dimensions "Name=FleetName,Value=fleetname"
--evaluation-periods 1
--alarm-actions "arn:aws:autoscaling:your-region-code:account-number-without-
hyphens:scalingPolicy:policyid:resource/appstream/fleet/fleetname:policyName/policyname"
--unit Count
```

Monitoring Amazon AppStream 2.0 Resources

You can monitor Amazon AppStream 2.0 using the following features:

- AppStream 2.0 console (p. 45)— You can quickly and easily view your fleet resource usage on the AppStream 2.0 console.
- CloudWatch metrics (p. 45)— AppStream 2.0 publishes metrics to Amazon CloudWatch for detailed tracking and deep dive analysis.

These statistics are recorded for an extended period so you can access historical information and gain a better perspective on how your fleets are performing.

Viewing Fleet Usage with the Console

Fleet metrics are available in the AppStream 2.0 console.

To view fleet usage in the console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left pane, choose **Fleets** and select a fleet.
3. In the **Fleet Usage** section, a graph is displayed for the default period, showing `RunningCapacity`, `InUseCapacity` and `CapacityUtilization`. You can select other metrics to plot on the graph. To view a different period, select one of the buttons below the graph.

For a list of all AppStream 2.0 metrics reported by CloudWatch, see Monitoring Amazon AppStream 2.0 with Amazon CloudWatch (p. 45).

Monitoring Amazon AppStream 2.0 with Amazon CloudWatch

You can monitor your Amazon AppStream 2.0 fleets using Amazon CloudWatch, which automatically collects and processes raw data from AppStream 2.0 into readable, near real-time metrics. For more information, see Amazon CloudWatch User Guide.

Topics

AppStream 2.0 Metrics and Dimensions

Amazon AppStream 2.0 sends the following metrics and dimension information to Amazon CloudWatch. For more information about the statistics gathered from CloudWatch metrics, see Amazon CloudWatch Statistics in the *Amazon CloudWatch User Guide*.

Amazon AppStream 2.0 Metrics

AppStream 2.0 sends metrics to CloudWatch one time every minute. The `AWS/AppStream` namespace includes the following metrics.

Metric	Description
`RunningCapacity`	Total number of instances currently running. Represents the number of concurrent streaming sessions that can be supported by the fleet in its current state. Units: Count Valid statistics: Average, Minimum, Maximum
`InUseCapacity`	Number of instances currently being used for streaming sessions. One `InUseCapacity` count represents one streaming session. Units: Count Valid statistics: Average, Minimum, Maximum
`PendingCapacity`	Number of instances being provisioned by AppStream 2.0. Represents the additional number of streaming sessions the fleet can support after provisioning is complete. When provisioning starts, it usually takes 10-20 minutes for an instance to become available for streaming. Units: Count Valid statistics: Average, Minimum, Maximum
`AvailableCapacity`	Number of idle instances currently available for user sessions. `AvailableCapacity = RunningCapacity - InUseCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
`DesiredCapacity`	Total number of instances that are either running or pending. This represents the total number of concurrent streaming sessions your fleet can support in a steady state. `DesiredCapacity = RunningCapacity + PendingCapacity`

Metric	Description
	Units: Count Valid statistics: Average, Minimum, Maximum

`CapacityUtilization`	The percentage of instances in a fleet that are being used, using the following formula.
	
	`                            InUseCapacity`
	`CapacityUtilization = -------------------- * 100`
	`                           RunningCapacity`
	
	Monitoring this metric helps with decisions about increasing or decreasing the value of a fleet's desired capacity.
	
	Units: Percent
	
	Valid statistics: Average, Minimum, Maximum

`InsufficientCapacityError`	Number of session requests rejected due to lack of capacity. One `InsufficientCapacityError` count represents one session rejection due to insufficient capacity. You can set alarms to use this metric to be notified of users waiting for streaming sessions. Units: Count Valid statistics: Average, Minimum, Maximum, Sum

Dimensions for Amazon AppStream 2.0 Metrics

Amazon AppStream 2.0 provides metrics for the following dimension.

Dimension	Description
`Fleet`	The name of the AppStream 2.0 fleet. All available statistics are filtered by `Fleet`.

Creating CloudWatch Alarms

After you have enabled CloudWatch metrics for your fleets, and the metrics are visible in the CloudWatch console, you can set alarms on the metrics. For more information, see Creating Amazon CloudWatch Alarms in the *Amazon CloudWatch User Guide*.

Custom CloudWatch Metrics

You can set up your own AppStream 2.0 CloudWatch metrics using custom metrics. For more information, see Custom CloudWatch metrics in the *Amazon CloudWatch User Guide*.

CloudWatch Pricing

You can view metrics in the AppStream 2.0 and CloudWatch consoles free of charge. Additional charges may apply when setting up alarms and dashboards, and using the CloudWatch API. For more information, see Amazon CloudWatch Pricing.

Controlling Access to Amazon AppStream 2.0

By default, IAM users don't have permission to create or modify AppStream 2.0 resources, use Fleet Auto Scaling, or perform tasks using the AppStream 2.0 API. To allow IAM users to create or modify resources and perform tasks, you must create IAM policies that grant permissions on specific resources and API actions, and then attach those policies to the IAM users or groups that require those permissions.

While creating AppStream 2.0 resources, AppStream 2.0 makes API calls to other AWS services on behalf of the user. This authentication is accomplished by the service assuming specific IAM roles available in the user's account. These IAM roles are created by the service when the user gets started with the service in an AWS region.

Topics

Using the IAM Service Role

The following service role is available for AppStream 2.0.

AmazonAppStreamServiceAccess

Allows AppStream 2.0 to create and manage AppStream 2.0 resources on the user's behalf. This role provides the permissions listed.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeAvailabilityZones",
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeSubnets",
        "ec2:AssociateAddress",
        "ec2:DisassociateAddress",
        "ec2:DescribeRouteTables"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:GetObjectVersion",
        "s3:DeleteObjectVersion",
        "s3:PutBucketPolicy"
      ],
      "Resource": "arn:aws:s3:::appstream2-36fb080bb8-*"
    }
  ]
}

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

Using Identity Based Policies

AppStream 2.0 provides managed policies that you can attach to your IAM users and allow users to control your AppStream 2.0 resources or perform API actions through the AWS CLI, SDK, or console.

AmazonAppStreamFullAccess

Provides full administrator access to AppStream 2.0.

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

AmazonAppStreamReadOnlyAccess

Provides your IAM users with read-only access to AppStream 2.0 resources.

```
{
"Version": "2012-10-17",
"Statement": [
   {
      "Effect": "Allow",
         "Action": [
            "appstream:Describe*",
            "appstream:List*",
            "appstream:Get*"
         ],
         "Resource": "*"
   }
 ]
}
```

Application Auto Scaling IAM Role

Before you can use AppStream 2.0 Fleet Auto Scaling, Application Auto Scaling needs permissions to access Amazon CloudWatch alarms and AppStream 2.0 fleets. The service role **ApplicationAutoscalingForAmazonAppStreamAccess** is automatically added to your IAM roles when you use the AppStream 2.0 console. If you plan to use the AWS CLI instead, you must create the following role.

Permissions

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "appstream:UpdateFleet",
        "appstream:DescribeFleets"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Trust Relationship

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

Application Auto Scaling Required IAM Permissions

To use AppStream 2.0 Fleet Auto Scaling, the IAM user accessing fleet creation and settings must have appropriate permissions for the services that support scaling. AppStream 2.0 requires the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "appstream:",
          "application-autoscaling:",
          "cloudwatch:DeleteAlarms",
          "cloudwatch:DescribeAlarmsForMetric",
          "cloudwatch:DisableAlarmActions",
          "cloudwatch:DescribeAlarms",
          "cloudwatch:EnableAlarmActions",
          "cloudwatch:ListMetrics",
          "cloudwatch:PutMetricAlarm",
          "iam:passrole",
          "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM Policies and the Amazon S3 Bucket for Home Folders

Access to the Amazon S3 bucket for Home folders is managed using IAM permissions and policies.

Topics

Deleting the Amazon S3 Bucket for Home Folders

AppStream 2.0 adds an Amazon S3 bucket policy that prevents accidental deletion of the S3 bucket, shown below. To delete the S3 bucket,delete the S3 bucket policy first, and then delete the S3 bucket.

For more information, see Deleting or Emptying a Bucket in the *Amazon Simple Storage Service Developer Guide*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PreventAccidentalDeletionOfBucket",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:DeleteBucket",
      "Resource": "arn:aws:s3:::appstream2-36fb080bb8-region-code-account-id-without-
hyphens"
    }
  ]
}
```

Restricting Administrator Access to the Amazon S3 Bucket for Home Folders

By default, administrators who can access the Amazon S3 bucket created by AppStream 2.0 can view and modify content that is part of users' Home folders. To restrict administrator access to the S3 bucket containing user files, we recommend applying the S3 bucket access policy based on the following policy template:

```
{
  "Sid": "RestrictedAccess",
  "Effect": "Deny",
  "NotPrincipal":
  {
    "AWS": [
      "arn:aws:iam::account-id-without-hyphens:role/service-role/
AmazonAppStreamServiceAccess",
      "arn:aws:sts::account-id-without-hyphens:assumed-role/AmazonAppStreamServiceAccess/
PhotonSession",
      "arn:aws:iam::account-id-without-hyphens:user/IAM-user-name-to-restrict"
    ]
  },
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::appstream2-36fb080bb8-region-code-account-id-without-hyphens"
  }
 ]
}
```

This policy only allows Home folder S3 bucket access to the users specified and to the AppStream 2.0 service. For every IAM user who should have access, replicate the following line:

```
      "arn:aws:iam::account-id-without-hyphens:user/IAM-user-name-to-restrict"
```

For example, the following policy restricts access to the Home folder S3 bucket for anyone other than IAM users bobsmith and terrypratt, and the AppStream 2.0 service, in region us-west-2 for account ID 123456789012.

```
{
  "Sid": "RestrictedAccess",
  "Effect": "Deny",
  "NotPrincipal":
  {
    "AWS": [
      "arn:aws:iam::123456789012:role/service-role/AmazonAppStreamServiceAccess",
      "arn:aws:sts::123456789012:assumed-role/AmazonAppStreamServiceAccess/PhotonSession",
      "arn:aws:iam::123456789012:user/bobsmith",
      "arn:aws:iam::123456789012:user/terrypratt"
    ]
  },
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::appstream2-36fb080bb8-us-west-2-123456789012"
  }
 ]
}
```

Amazon AppStream 2.0 Service Limits

By default, AWS limits the resources you can create and the number of users that can use the service. The following table lists the limits for each AppStream 2.0 resource.

Default Limits Per Region

Resource	Default Limit
Stacks	5 per account
Fleets	5 per account
Streaming instances	5 per account
Images	5 per account
Image builders	5 per account
Users	5 per account

For more information about limits, including how to increase limits for your account, see AWS Service Limits in *AWS General Reference*.

Troubleshooting

The following are possible problems you may have while trying to use Amazon AppStream 2.0.

Topics

I cannot connect to the Internet from my image builder.

Image builders cannot communicate to the Internet by default. To resolve this issue, launch your image builder in a VPC subnet that has Internet access. You can enable Internet access from your VPC subnet using a NAT gateway. Alternatively, you can configure an Internet gateway in your VPC, and attach an Elastic IP address to your image builder. For more information, see Network Settings for Fleet and Image Builder Instances (p. 25).

When I tried installing my application, I see an error that the operating system version is not supported.

Only applications that can be installed on Windows Server 2012 R2 can be added to an AppStream 2.0 image. Check if your application is supported on Microsoft Windows Server 2012 R2.

When I connect to my image builder, I see a login screen asking me to enter Ctrl + Alt + Delete to log in. However, my local machine intercepts the key strokes.

Your client may intercept certain key combinations locally instead of sending them to the image builder session. To reliably send the **Ctrl + Alt + Delete** key combination to the image builder, choose **Admin Commands**, **Send Ctrl + Alt + Delete**. The **Admin Commands** menu is available on the top right corner of the image builder session toolbar.

When I switched between admin and test modes, I saw a request for a password. I don’t know how to get a password.

AppStream 2.0 usually logs you into the user mode that you choose automatically. On some occasions, the switch may not happen automatically. If a password is requested, choose **Admin Commands**, **Log me in**. This sends a one-time password, securely, to your image builder and pastes it into the **Password** field.

I get an error when I add my installed application.

Check if your application type is supported. You can add applications of the types .exe, .lnk, and .bat.

Check if your application is installed under the C:\Users folder hierarchy. Any application installed under the C:\Users is not supported. Select a different installation folder under C:\ when installing the application.

I accidentally quit a background service on the image builder and got disconnected. I am now unable to connect to that image builder.

Try stopping the image builder, restarting it and connecting to it again. If the problem persists, you need to launch (create) a new image builder. Do not stop any background services running on the image builder instance. Doing so may interrupt your image builder session or interfere with the image creation.

The application fails to launch in test mode.

Check if your application requires elevated user privileges or any special permissions that are usually available only to an administrator. The **Image Builder Test** mode has the same limited permissions on the image builder instance as your end users have on an AppStream 2.0 test fleet. If your applications require elevated permissions, they do not launch in the **Image Builder Test** mode.

The application could not connect to a network resource in my VPC.

Check if the image builder was launched in the correct VPC subnet. You may also need to verify that the route tables in your VPC are configured to allow a connection.

I customized my image builder desktop, but my changes are not available when connecting to a session after launching a fleet from the image I created.

Changes that are saved as part of a local user session, such as wallpaper, are not persisted when creating an image. To persist any local user session changes, add them to the local group policy on the image builder instance.

My application is missing a command line parameter when launching.

You can provide a command line parameter when using image builder to add an application to an image. If the launch parameters for the application do not change for each user, you can enter them while adding an application to the image in the image builder instance.

If the launch parameters are different for every launch, you can pass them programmatically when using the `CreateStreamingURL` API. Set the `sessionContext` and `applicationID` parameters in the API fields. The sessionContext is included as a command line option when launching the application.

If the launch parameters need to be computed on the fly, you can launch your application using a script. You can parse the `sessionContext` parameter within your script before launching your application with a computed parameter.

I am unable to use my image with a fleet after installing an antivirus application.

You can install any tools, including antivirus programs, on your AppStream 2.0 stack by using the image builder before creating an image. However, these programs should not block any network ports or stop any processes that are used by the AppStream 2.0 service. We recommend testing your application in **Image Builder Test** mode before creating an image and attempting to use it with a fleet.

My image creation failed.

Verify that you did not make any changes to AppStream 2.0 services before starting the image creation. Try creating your image again; if it fails, contact AWS Support.

SAML federation is not working. The user is not authorized to view AppStream 2.0 applications.

This may happen because the policy of the IAM role, which is assumed by the federated user who is accessing an AppStream 2.0 stack, does not include permissions to the stack ARN. Edit the role permissions to include the stack ARN. For more information, see Enabling Single Sign-on Access to AppStream 2.0 Using SAML 2.0 (p. 30) and Troubleshooting SAML 2.0 Federation with AWS in the *IAM User Guide*.

I get an invalid redirect URI error.

This error happens due to a malformed or invalid AppStream 2.0 stack relay state URL. Make sure that the relay state configured in your federation setup is the same as the stack relay state available in stack details through the AppStream 2.0 console. If they are the same and the problem still persists, contact AWS Support. For more information, see Enabling Single Sign-on Access to AppStream 2.0 Using SAML 2.0 (p. 30).

My stack's Home Folders aren't working correctly.

Home Folder backup to the S3 bucket can have problems under the following scenarios.

- There is no Internet connectivity from the streaming instance, or there is no access to private S3 VPC endpoint if applicable.
- The administrator deletes the bucket created by the service.
- Network bandwidth consumption is too high. For example, multiple files of large size being downloaded or streamed by the user while the service is trying to back up the Home Folder containing large files to S3.

- The administrator incorrectly edits the S3 permissions of the service role **AmazonAppStreamServiceAccess**.

For more information, see Amazon Simple Storage Service Console User Guide and Amazon Simple Storage Service Developer Guide.

Amazon AppStream 2.0 Windows Image Version History

The following table describes the image versions released for Amazon AppStream 2.0.

- **Latest base image:** Amazon-AppStream2-Image-Builder-05-18-2017
- **Latest sample apps image:** Amazon-AppStream2-Sample-Image-05-18-2017

AWS Images Released in 2017

Image Name	Description
Amazon-AppStream2-Image-Builder-05-18-2017 Amazon-AppStream2-Sample-Image-05-18-2017	• Adds support for Amazon AppStream 2.0 Home Folders • Includes Microsoft Windows updates up to May 16th, 2017 • Includes updates to resolve an intermittent network issue that affects Internet connections from streaming instances • Includes updates to resolve an issue with application tiles not functioning correctly

Document History for Amazon AppStream 2.0

The following table describes the documentation for this release of Amazon AppStream 2.0.

- **API version:** 2016-12-01
- **Latest documentation update:** May 18, 2017

Change	Description	Date
Home Folders	Created Home Folders topic and other updates to support the launch of this feature.	May 18, 2017
Default Internet access	Created internet and networking topics and other updates to support the launch of this feature.	April 21, 2017
Fleet automatic scaling	Created automatic scaling topic and other updates to support the launch of this feature.	March 23, 2017
Fleet management	Created stacks and fleets set-up topic and other updates to support the launch of this feature.	February 22, 2017
SAML 2.0 support	Created SAML 2.0 set-up topic and other updates to support the launch of this feature.	February 15, 2017
Image builder support	Created image builder tutorial and other updates to support the launch of this feature.	January 19, 2017
Initial release	First publication of this guide.	December 01, 2016

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.